IsaGeoCoq: Partial porting of GeoCoq 2.4.0. Case studies: Tarski's postulate of parallels implies the 5th postulate of Euclid, the postulate of Playfair and the original postulate of Euclid.

Roland Coghetto

March 17, 2025

Abstract

The GeoCoq library contains a formalization of geometry using the Coq proof assistant. It contains both proofs about the foundations of geometry [20, 15, 6, 16] and high-level proofs in the same style as in high-school. [1](Code Repository https://github.com/GeoCoq/GeoCoq).

Some theorems also inspired by [20] are also formalized with others ITP(Metamath, Mizar) or ATP [24, 25, 3, 23, 4, 2, 17, 5, 11, 19, 8, 9, 10].

We port a part of the GeoCoq 2.4.0 library within the Isabelle/Hol proof assistant: more precisely, the files Chap02.v to Chap13_3.v, suma.v as well as the associated definitions and some useful files for the demonstration of certain parallel postulates.

While the demonstrations in Coq are written in procedural language [26], the transcript is done in declarative language Isar[18].

The synthetic approach of the demonstrations are directly inspired by those contained in GeoCoq. Some demonstrations are credited to G.E Martin(«lemma bet_le_lt:» in Ch11_angles.thy, proved by Martin as Theorem 18.17 in [14]) or Gupta H.N (Krippen Lemma, proved by Gupta in its PhD in 1965 as Theorem 3.45). (See [12]).

In this work, the proofs are not contructive. The sledeghammer tool being used to find some demonstrations.

The names of the lemmas and theorems used are kept as far as possible as well as the definitions. A different translation has been proposed when the name was already used in Isabel/Hol ("Len" is translated as "TarskiLen") or that characters were not allowed in Isabel/Hol ("anga'" in Ch13_angles.v is translated as "angaP"). For some definitions the highlighting of a variable has changed the order or the position of the variables (Midpoint, Out, Inter,...).

All the lemmas are valid in absolute/neutral space defined with Tarski's axioms.

It should be noted that T.J.M. Makarios [13] has begun some demonstrations of certain proposals mainly those corresponding to SST chapters 2 and 3. It uses a definition that does not quite coincide with the definition used in Geocoq and here. As an example, Makarios introduces the axiom A11 (Axiom of continuity) in the definition of the locale "Tarski_absolute_space".

Furthermore, the definition of the locale "TarskiAbsolute" [22, 21] is not not identical to the one defined in the "Tarski_neutral_dimensionless" class of GeoCoq. Indeed this one does not contain the axiom "upper_dimension". In some cases particular, it is nevertheless to use the axiom "upper_dimension". The addition of the word "_2D" in the file indicates its presence.

In the last part, it is formalized that, in the neutral/absolute space, the axiom of the parallels of the system of Tarski implies the Playfair axiom, the 5th postulate of euclide and the postulate original from Euclid. These proofs, which are not constructive, are directly inspired by [12, 7].

Contents

theory *Tarski-Neutral*

imports
Main

begin

1 Tarski's axiom system for neutral geometry

1.1 Tarski's axiom system for neutral geometry: dimensionless

locale *Tarski-neutral-dimensionless* =
fixes *Bet* :: $'p \Rightarrow 'p \Rightarrow 'p \Rightarrow bool$
fixes *Cong* :: $'p \Rightarrow 'p \Rightarrow 'p \Rightarrow 'p \Rightarrow bool$
assumes *cong-pseudo-reflexivity*: $\forall\ a\ b.$
$Cong\ a\ b\ b\ a$
and *cong-inner-transitivity*: $\forall\ a\ b\ p\ q\ r\ s.$
$Cong\ a\ b\ p\ q\ \wedge$
$Cong\ a\ b\ r\ s$
$\longrightarrow$
$Cong\ p\ q\ r\ s$
and *cong-identity*: $\forall\ a\ b\ c.$
$Cong\ a\ b\ c\ c$
$\longrightarrow$
$a = b$
and *segment-construction*: $\forall\ a\ b\ c\ q.$
$\exists x. (Bet\ q\ a\ x \wedge Cong\ a\ x\ b\ c)$
and *five-segment*: $\forall\ a\ b\ c\ a'\ b'\ c'.$
$a \neq b\ \wedge$
$Bet\ a\ b\ c\ \wedge$
$Bet\ a'\ b'\ c' \wedge$
$Cong\ a\ b\ a'\ b'\ \wedge$
$Cong\ b\ c\ b'\ c'\ \wedge$
$Cong\ a\ d\ a'\ d'\ \wedge$
$Cong\ b\ d\ b'\ d'$
$\longrightarrow$
$Cong\ c\ d\ c'\ d'$
and *between-identity*: $\forall\ a\ b.$
$Bet\ a\ b\ a$
$\longrightarrow$
$a = b$
and *inner-pasch*: $\forall\ a\ b\ c\ p\ q.$
$Bet\ a\ p\ c\ \wedge$
$Bet\ b\ q\ c$
$\longrightarrow$
$(\exists\ x.\ Bet\ p\ x\ b \wedge Bet\ q\ x\ a)$
and *lower-dim*: $\exists\ a\ b\ c.\ (\neg\ Bet\ a\ b\ c \wedge \neg\ Bet\ b\ c\ a \wedge \neg\ Bet\ c\ a\ b)$

1.2 Tarski's axiom system for neutral geometry: 2D

locale *Tarski-2D* = *Tarski-neutral-dimensionless* +
assumes *upper-dim*: $\forall\ a\ b\ c\ p\ q.$
$p \neq q\ \wedge$
$Cong\ a\ p\ a\ q\ \wedge$
$Cong\ b\ p\ b\ q\ \wedge$
$Cong\ c\ p\ c\ q$
$\longrightarrow$
$(Bet\ a\ b\ c \vee Bet\ b\ c\ a \vee Bet\ c\ a\ b)$

2 Definitions

2.1 Tarski's axiom system for neutral geometry: dimensionless

context *Tarski-neutral-dimensionless*
begin

2.1.1 Congruence

definition *OFSC* ::
$['p,'p,'p,'p,'p,'p,'p,'p] \Rightarrow bool$
(‹- - - - *OFSC* - - - -› [*99*,*99*,*99*,*99*,*99*,*99*,*99*,*99*] *50*)
where
$A\ B\ C\ D\ OFSC\ A'\ B'\ C'\ D' \equiv$

$Bet\ A\ B\ C \land$
$Bet\ A'\ B'\ C' \land$
$Cong\ A\ B\ A'\ B' \land$
$Cong\ B\ C\ B'\ C' \land$
$Cong\ A\ D\ A'\ D' \land$
$Cong\ B\ D\ B'\ D'$

definition *Cong3* ::
$['p,'p,'p,'p,'p,'p] \Rightarrow bool$
(‹- - - *Cong3* - - -› [*99*,*99*,*99*,*99*,*99*,*99*] *50*)
where
$A\ B\ C\ Cong3\ A'\ B'\ C' \equiv$

$Cong\ A\ B\ A'\ B' \land$
$Cong\ A\ C\ A'\ C' \land$
$Cong\ B\ C\ B'\ C'$

2.1.2 Betweenness

definition *Col* ::
$['p,'p,'p] \Rightarrow bool$
(‹*Col* - - -› [*99*,*99*,*99*] *50*)
where
$Col\ A\ B\ C \equiv$

$Bet\ A\ B\ C \lor Bet\ B\ C\ A \lor Bet\ C\ A\ B$

definition *Bet4* ::
$['p,'p,'p,'p] \Rightarrow bool$
(‹*Bet4* - - - -› [*99*,*99*,*99*,*99*] *50*)
where
$Bet4\ A1\ A2\ A3\ A4 \equiv$

$Bet\ A1\ A2\ A3 \land$
$Bet\ A2\ A3\ A4 \land$
$Bet\ A1\ A3\ A4 \land$
$Bet\ A1\ A2\ A4$

definition *BetS* ::
$['p,'p,'p] \Rightarrow bool$ (‹*BetS* - - -› [*99*,*99*,*99*] *50*)
where
$BetS\ A\ B\ C \equiv$

$Bet\ A\ B\ C \land$
$A \neq B \land$
$B \neq C$

2.1.3 Collinearity

definition *FSC* ::
$['p,'p,'p,'p,'p,'p,'p,'p] \Rightarrow bool$
(‹- - - - *FSC* - - - -› [*99*,*99*,*99*,*99*,*99*,*99*,*99*,*99*] *50*)

where
$A\ B\ C\ D\ FSC\ A'\ B'\ C'\ D' \equiv$

$Col\ A\ B\ C\ \wedge$
$A\ B\ C\ Cong3\ A'\ B'\ C'\ \wedge$
$Cong\ A\ D\ A'\ D'\ \wedge$
$Cong\ B\ D\ B'\ D'$

2.1.4 Congruence and Betweenness

definition *IFSC* ::
$['p,'p,'p,'p,'p,'p,'p,'p] \Rightarrow bool$
(‹- - - - *IFSC* - - - -› [*99*,*99*,*99*,*99*,*99*,*99*,*99*,*99*] *50*)
where
$A\ B\ C\ D\ IFSC\ A'\ B'\ C'\ D' \equiv$

$Bet\ A\ B\ C\ \wedge$
$Bet\ A'\ B'\ C'\ \wedge$
$Cong\ A\ C\ A'\ C'\ \wedge$
$Cong\ B\ C\ B'\ C'\ \wedge$
$Cong\ A\ D\ A'\ D'\ \wedge$
$Cong\ C\ D\ C'\ D'$

2.1.5 Between transivitity LE

definition *Le* ::
$['p,'p,'p,'p] \Rightarrow bool$ (‹- - *Le* - -› [*99*,*99*,*99*,*99*] *50*)
where $A\ B\ Le\ C\ D \equiv$

$\exists\ E.\ (Bet\ C\ E\ D \wedge Cong\ A\ B\ C\ E)$

definition *Lt* ::
$['p,'p,'p,'p] \Rightarrow bool$ (‹- - *Lt* - -› [*99*,*99*,*99*,*99*] *50*)
where $A\ B\ Lt\ C\ D \equiv$

$A\ B\ Le\ C\ D \wedge \neg\ Cong\ A\ B\ C\ D$

definition *Ge* ::
$['p,'p,'p,'p] \Rightarrow bool$ (‹- -*Ge* - -› [*99*,*99*,*99*,*99*] *50*)
where $A\ B\ Ge\ C\ D \equiv$

$C\ D\ Le\ A\ B$

definition *Gt* ::
$['p,'p,'p,'p] \Rightarrow bool$ (‹- - *Gt* - -› [*99*,*99*,*99*,*99*] *50*)
where $A\ B\ Gt\ C\ D \equiv$

$C\ D\ Lt\ A\ B$

2.1.6 Out lines

definition *Out* ::
$['p,'p,'p] \Rightarrow bool$ (‹- *Out* - -› [*99*,*99*,*99*] *50*)
where $P\ Out\ A\ B \equiv$

$A \neq P\ \wedge$
$B \neq P\ \wedge$
$(Bet\ P\ A\ B \vee Bet\ P\ B\ A)$

2.1.7 Midpoint

definition *Midpoint* ::
$['p,'p,'p] \Rightarrow bool$ (‹- *Midpoint* - -› [*99*,*99*,*99*] *50*)
where $M\ Midpoint\ A\ B \equiv$

$Bet\ A\ M\ B\ \wedge$

$Cong\ A\ M\ M\ B$

2.1.8 Orthogonality

definition *Per* ::
$[{}'p,{}'p,{}'p] \Rightarrow bool$ (‹*Per* - - -› [99,99,99] 50)
where $Per\ A\ B\ C \equiv$

$\exists\ C'::{}'p.\ (B\ Midpoint\ C\ C' \land Cong\ A\ C\ A\ C')$

definition *PerpAt* ::
$[{}'p,{}'p,{}'p,{}'p,{}'p] \Rightarrow bool$ (‹- *PerpAt* - - - - › [99,99,99,99,99] 50)
where $X\ PerpAt\ A\ B\ C\ D \equiv$

$A \neq B \land$
$C \neq D \land$
$Col\ X\ A\ B \land$
$Col\ X\ C\ D \land$
$(\forall\ U\ V.\ ((Col\ U\ A\ B \land Col\ V\ C\ D) \longrightarrow Per\ U\ X\ V))$

definition *Perp* ::
$[{}'p,{}'p,{}'p,{}'p] \Rightarrow bool$ (‹- - *Perp* - -› [99,99,99,99] 50)
where $A\ B\ Perp\ C\ D \equiv$

$\exists\ X::{}'p.\ X\ PerpAt\ A\ B\ C\ D$

2.1.9 Coplanar

definition *Coplanar* ::
$[{}'p,{}'p,{}'p,{}'p] \Rightarrow bool$ (‹*Coplanar* - - - -› [99,99,99,99] 50)
where $Coplanar\ A\ B\ C\ D \equiv$
$\exists\ X.\ (Col\ A\ B\ X \land Col\ C\ D\ X) \lor$
$(Col\ A\ C\ X \land Col\ B\ D\ X) \lor$
$(Col\ A\ D\ X \land Col\ B\ C\ X)$

definition *TS* ::
$[{}'p,{}'p,{}'p,{}'p] \Rightarrow bool$ (‹- - *TS* - -› [99,99,99,99] 50)
where $A\ B\ TS\ P\ Q \equiv$
$\neg\ Col\ P\ A\ B \land \neg\ Col\ Q\ A\ B \land (\exists\ T::{}'p.\ Col\ T\ A\ B \land Bet\ P\ T\ Q)$

definition *ReflectL* ::
$[{}'p,{}'p,{}'p,{}'p] \Rightarrow bool$ (‹- - *ReflectL* - -› [99,99,99,99] 50)
where $P'\ P\ ReflectL\ A\ B \equiv$
$(\exists\ X.\ X\ Midpoint\ P\ P' \land Col\ A\ B\ X) \land (A\ B\ Perp\ P\ P' \lor P = P')$

definition *Reflect* ::
$[{}'p,{}'p,{}'p,{}'p] \Rightarrow bool$ (‹- - *Reflect* - -› [99,99,99,99] 50)
where $P'\ P\ Reflect\ A\ B \equiv$
$(A \neq B \land P'\ P\ ReflectL\ A\ B) \lor (A = B \land A\ Midpoint\ P\ P')$

definition *InAngle* ::
$[{}'p,{}'p,{}'p,{}'p] \Rightarrow bool$ (‹- *InAngle* - - -› [99,99,99,99] 50)
where $P\ InAngle\ A\ B\ C \equiv$
$A \neq B \land C \neq B \land P \neq B \land$
$(\exists\ X.\ Bet\ A\ X\ C \land (X = B \lor B\ Out\ X\ P))$

definition *ParStrict*::
$[{}'p,{}'p,{}'p,{}'p] \Rightarrow bool$ (‹- - *ParStrict* - -› [99,99,99,99] 50)
where $A\ B\ ParStrict\ C\ D \equiv\ Coplanar\ A\ B\ C\ D \land \neg\ (\exists\ X.\ Col\ X\ A\ B \land Col\ X\ C\ D)$

definition *Par*::
$[{}'p,{}'p,{}'p,{}'p] \Rightarrow bool$ (‹- - *Par* - -› [99,99,99,99] 50)
where $A\ B\ Par\ C\ D \equiv$
$A\ B\ ParStrict\ C\ D \lor (A \neq B \land C \neq D \land Col\ A\ C\ D \land Col\ B\ C\ D)$

definition *Plg*::
$[{}'p,{}'p,{}'p,{}'p] \Rightarrow bool$ (‹*Plg* - - - -› [99,99,99,99] 50)

where *Plg* $A\ B\ C\ D \equiv$
$(A \neq C \vee B \neq D) \wedge (\exists\ M.\ M\ Midpoint\ A\ C \wedge M\ Midpoint\ B\ D)$

definition *ParallelogramStrict*::
$['p,'p,'p,'p] \Rightarrow bool$ (‹*ParallelogramStrict* - - - -› [99,99,99,99] 50)
where *ParallelogramStrict* $A\ B\ A'\ B' \equiv$
$A\ A'\ TS\ B\ B' \wedge A\ B\ Par\ A'\ B' \wedge Cong\ A\ B\ A'\ B'$

definition *ParallelogramFlat*::
$['p,'p,'p,'p] \Rightarrow bool$ (‹*ParallelogramFlat* - - - -› [99,99,99,99] 50)
where *ParallelogramFlat* $A\ B\ A'\ B' \equiv$
$Col\ A\ B\ A' \wedge Col\ A\ B\ B' \wedge$
$Cong\ A\ B\ A'\ B' \wedge Cong\ A\ B'\ A'\ B \wedge$
$(A \neq A' \vee B \neq B')$

definition *Parallelogram*::
$['p,'p,'p,'p] \Rightarrow bool$ (‹*Parallelogram* - - - -› [99,99,99,99] 50)
where *Parallelogram* $A\ B\ A'\ B' \equiv$
ParallelogramStrict $A\ B\ A'\ B' \vee$ *ParallelogramFlat* $A\ B\ A'\ B'$

definition *Rhombus*::
$['p,'p,'p,'p] \Rightarrow bool$ (‹*Rhombus* - - - -› [99,99,99,99] 50)
where *Rhombus* $A\ B\ C\ D \equiv Plg\ A\ B\ C\ D \wedge Cong\ A\ B\ B\ C$

definition *Rectangle*::
$['p,'p,'p,'p] \Rightarrow bool$ (‹*Rectangle* - - - -› [99,99,99,99] 50)
where *Rectangle* $A\ B\ C\ D \equiv Plg\ A\ B\ C\ D \wedge Cong\ A\ C\ B\ D$

definition *Square*::
$['p,'p,'p,'p] \Rightarrow bool$ (‹*Square* - - - -› [99,99,99,99] 50)
where *Square* $A\ B\ C\ D \equiv Rectangle\ A\ B\ C\ D \wedge Cong\ A\ B\ B\ C$

definition *Lambert*::
$['p,'p,'p,'p] \Rightarrow bool$ (‹*Lambert* - - - -› [99,99,99,99] 50)
where *Lambert* $A\ B\ C\ D \equiv$
$A \neq B \wedge B \neq C \wedge C \neq D \wedge$
$A \neq D \wedge Per\ B\ A\ D \wedge Per\ A\ D\ C \wedge Per\ A\ B\ C \wedge Coplanar\ A\ B\ C\ D$

2.1.10 Plane

definition *OS* ::
$['p,'p,'p,'p] \Rightarrow bool$ (‹- - *OS* - -› [99,99,99,99] 50)
where $A\ B\ OS\ P\ Q \equiv$
$\exists\ R::'p.\ A\ B\ TS\ P\ R \wedge A\ B\ TS\ Q\ R$

definition *TSP* ::
$['p,'p,'p,'p,'p] \Rightarrow bool$ (‹- - -*TSP* - -› [99,99,99,99,99] 50)
where $A\ B\ C\ TSP\ P\ Q \equiv$
$(\neg\ Coplanar\ A\ B\ C\ P) \wedge (\neg\ Coplanar\ A\ B\ C\ Q) \wedge$
$(\exists\ T.\ Coplanar\ A\ B\ C\ T \wedge Bet\ P\ T\ Q)$

definition *OSP* ::
$['p,'p,'p,'p,'p] \Rightarrow bool$ (‹- - - *OSP* - -› [99,99,99,99,99] 50)
where $A\ B\ C\ OSP\ P\ Q \equiv$
$\exists\ R.\ ((A\ B\ C\ TSP\ P\ R) \wedge (A\ B\ C\ TSP\ Q\ R))$

definition *Saccheri*::
$['p,'p,'p,'p] \Rightarrow bool$ (‹*Saccheri* - - - -› [99,99,99,99] 50)
where *Saccheri* $A\ B\ C\ D \equiv$
$Per\ B\ A\ D \wedge Per\ A\ D\ C \wedge Cong\ A\ B\ C\ D \wedge A\ D\ OS\ B\ C$

2.1.11 Line reflexivity 2D

definition *ReflectLAt* ::
$['p,'p,'p,'p,'p] \Rightarrow bool$ (‹- *ReflectLAt* - - - -› [99,99,99,99,99] 50)
where $M\ ReflectLAt\ P'\ P\ A\ B \equiv$
$(M\ Midpoint\ P\ P' \wedge Col\ A\ B\ M) \wedge (A\ B\ Perp\ P\ P' \vee P = P')$

definition *ReflectAt* ::
[$'p,'p,'p,'p,'p$] $\Rightarrow$ *bool* (‹- *ReflectAt* - - - -› [*99*,*99*,*99*,*99*,*99*] *50*)
where M *ReflectAt* P' P A B $\equiv$
$(A \neq B \wedge M$ *ReflectLAt* P' P A $B) \vee (A = B \wedge A = M \wedge M$ *Midpoint* P $P')$

2.1.12 Line reflexivity

definition *upper-dim-axiom* ::
bool (‹*UpperDimAxiom*› [] *50*)
where
upper-dim-axiom $\equiv$

$\forall$ A B C P Q.
$P \neq Q \wedge$
Cong A P A Q $\wedge$
Cong B P B Q $\wedge$
Cong C P C Q
$\longrightarrow$
(*Bet* A B C $\vee$ *Bet* B C A $\vee$ *Bet* C A B)

definition *all-coplanar-axiom* ::
bool (‹*AllCoplanarAxiom*› [] *50*)
where
AllCoplanarAxiom $\equiv$

$\forall$ A B C P Q.
$P \neq Q \wedge$
Cong A P A Q $\wedge$
Cong B P B Q $\wedge$
Cong C P C Q
$\longrightarrow$
(*Bet* A B C $\vee$ *Bet* B C A $\vee$ *Bet* C A B)

2.1.13 Angles

definition *CongA* ::
[$'p,'p,'p,'p,'p,'p$] $\Rightarrow$ *bool* (‹- - - *CongA* - - -› [*99*,*99*,*99*,*99*,*99*,*99*] *50*)
where A B C *CongA* D E F $\equiv$
$A \neq B \wedge C \neq B \wedge D \neq E \wedge F \neq E \wedge$
($\exists$ A' C' D' F'. *Bet* B A A' $\wedge$ *Cong* A A' E D $\wedge$
Bet B C C' $\wedge$ *Cong* C C' E F $\wedge$
Bet E D D' $\wedge$ *Cong* D D' B A $\wedge$
Bet E F F' $\wedge$ *Cong* F F' B C $\wedge$
Cong A' C' D' F')

definition *LeA* ::
[$'p,'p,'p,'p,'p,'p$] $\Rightarrow$ *bool* (‹- - - *LeA* - - -› [*99*,*99*,*99*,*99*,*99*,*99*] *50*)
where A B C *LeA* D E F $\equiv$
$\exists$ P. (P *InAngle* D E F $\wedge$ A B C *CongA* D E P)

definition *LtA* ::
[$'p,'p,'p,'p,'p,'p$] $\Rightarrow$ *bool* (‹- - - *LtA* - - -› [*99*,*99*,*99*,*99*,*99*,*99*] *50*)
where A B C *LtA* D E F $\equiv$ A B C *LeA* D E F $\wedge$ $\neg$ A B C *CongA* D E F

definition *GtA* ::
[$'p,'p,'p,'p,'p,'p$] $\Rightarrow$ *bool* (‹- - - *GtA* - - -› [*99*,*99*,*99*,*99*,*99*,*99*] *50*)
where A B C *GtA* D E F $\equiv$ D E F *LtA* A B C

definition *Acute* ::
[$'p,'p,'p$] $\Rightarrow$ *bool* (‹*Acute* - - -› [*99*,*99*,*99*] *50*)
where *Acute* A B C $\equiv$
$\exists$ A' B' C'. (*Per* A' B' C' $\wedge$ A B C *LtA* A' B' C')

definition *Obtuse* ::
[$'p,'p,'p$] $\Rightarrow$ *bool* (‹*Obtuse* - - -› [*99*,*99*,*99*] *50*)
where *Obtuse* A B C $\equiv$

∃ A′ B′ C′. (*Per* A′ B′ C′ ∧ A′ B′ C′ *LtA* A B C)

definition *OrthAt* ::
['p,'p,'p,'p,'p,'p] ⇒ *bool* (‹- *OrthAt* - - - - -› [99,99,99,99,99,99] 50)
where X *OrthAt* A B C U V ≡
¬ *Col* A B C ∧ U ≠ V ∧ *Coplanar* A B C X ∧ *Col* U V X ∧
(∀ P Q. (*Coplanar* A B C P ∧ *Col* U V Q) ⟶ *Per* P X Q)

definition *Orth* ::
['p,'p,'p,'p,'p] ⇒ *bool* (‹- - - *Orth* - -› [99,99,99,99,99] 50)
where A B C *Orth* U V ≡ ∃ X. X *OrthAt* A B C U V

definition *SuppA* ::
['p,'p,'p,'p,'p,'p] ⇒ *bool*
(‹- - - *SuppA* - - - › [99,99,99,99,99,99] 50)
where
A B C *SuppA* D E F ≡
A ≠ B ∧ (∃ A′. *Bet* A B A′ ∧ D E F *CongA* C B A′)

2.1.14 Sum of angles

definition *SumA* ::
['p,'p,'p,'p,'p,'p,'p,'p,'p] ⇒ *bool* (‹- - - - - - *SumA* - - -› [99,99,99,99,99,99,99,99,99] 50)
where
A B C D E F *SumA* G H I ≡

∃ J. (C B J *CongA* D E F ∧ ¬ B C *OS* A J ∧ *Coplanar* A B C J ∧ A B J *CongA* G H I)

definition *TriSumA* ::
['p,'p,'p,'p,'p,'p] ⇒ *bool* (‹- - - *TriSumA* - - -› [99,99,99,99,99,99] 50)
where
A B C *TriSumA* D E F ≡

∃ G H I. (A B C B C A *SumA* G H I ∧ G H I C A B *SumA* D E F)

definition *SAMS* ::
['p,'p,'p,'p,'p,'p] ⇒ *bool* (‹*SAMS* - - - - - -› [99,99,99,99,99,99] 50)
where
SAMS A B C D E F ≡

(A ≠ B ∧
(E *Out* D F ∨ ¬ *Bet* A B C)) ∧
(∃ J. (C B J *CongA* D E F ∧ ¬ (B C *OS* A J) ∧ ¬ (A B *TS* C J) ∧ *Coplanar* A B C J))

2.1.15 Parallelism

definition *Inter* ::
['p,'p,'p,'p,'p] ⇒ *bool* (‹- *Inter* - - - -› [99,99,99,99,99] 50)
where X *Inter* A1 A2 B1 B2 ≡

B1 ≠ B2 ∧
(∃ P::'p. (*Col* P B1 B2 ∧ ¬ *Col* P A1 A2)) ∧
Col A1 A2 X ∧ *Col* B1 B2 X

2.1.16 Perpendicularity

definition *Perp2* ::
['p,'p,'p,'p,'p] ⇒ *bool* (‹- *Perp2* - - - -› [99,99,99,99,99] 50)
where
P *Perp2* A B C D ≡

∃ X Y. (*Col* P X Y ∧ X Y *Perp* A B ∧ X Y *Perp* C D)

2.1.17 Lentgh

definition *QCong*::
(['p,'p] ⇒ *bool*) ⇒ *bool* (‹*QCong* -› [99] 50)

where
  $QCong\ l \equiv$

$\exists\ A\ B.\ (\forall\ X\ Y.\ (Cong\ A\ B\ X\ Y \longleftrightarrow l\ X\ Y))$

definition *TarskiLen*::
$['p,'p,(['p,'p] \Rightarrow bool)] \Rightarrow bool$ (‹*TarskiLen* - - -› $[99,99,99]\ 50$)
where
  $TarskiLen\ A\ B\ l \equiv$

$QCong\ l \wedge l\ A\ B$

definition *QCongNull* ::
$(['p,'p] \Rightarrow bool) \Rightarrow bool$ (‹*QCongNull* -› $[99]\ 50$)
where
  $QCongNull\ l \equiv$

$QCong\ l \wedge (\exists\ A.\ l\ A\ A)$

2.1.18 Equivalence Class of Angles

definition *QCongA* ::
$(['p,\ 'p,\ 'p] \Rightarrow bool) \Rightarrow bool$ (‹*QCongA* -› $[99]\ 50$)
where
  $QCongA\ a \equiv$

$\exists\ A\ B\ C.\ (A \neq B \wedge C \neq B \wedge (\forall\ X\ Y\ Z.\ A\ B\ C\ CongA\ X\ Y\ Z \longleftrightarrow a\ X\ Y\ Z))$

definition *Ang* ::
$['p,'p,'p,\ (['p,\ 'p,\ 'p] \Rightarrow bool)\ ] \Rightarrow bool$ (‹- - - *Ang* -› $[99,99,99,99]\ 50$)
where
  $A\ B\ C\ Ang\ a \equiv$

$QCongA\ a \wedge$
$a\ A\ B\ C$

definition *QCongAAcute* ::
$(['p,\ 'p,\ 'p] \Rightarrow bool) \Rightarrow bool$ (‹*QCongAACute* -› $[99]\ 50$)
where
  $QCongAAcute\ a \equiv$

$\exists\ A\ B\ C.\ (Acute\ A\ B\ C \wedge (\forall\ X\ Y\ Z.\ (A\ B\ C\ CongA\ X\ Y\ Z \longleftrightarrow a\ X\ Y\ Z)))$

definition *AngAcute* ::
$['p,'p,'p,\ (['p,'p,'p] \Rightarrow bool)] \Rightarrow bool$ (‹- - - *AngAcute* -› $[99,99,99,99]\ 50$)
where
  $A\ B\ C\ AngAcute\ a \equiv$

$((QCongAAcute\ a) \wedge (a\ A\ B\ C))$

definition *QCongANullAcute* ::
$(['p,'p,'p] \Rightarrow bool) \Rightarrow bool$ (‹*QCongANullAcute* -› $[99]\ 50$)
where
  $QCongANullAcute\ a \equiv$

$QCongAAcute\ a \wedge$
$(\forall\ A\ B\ C.\ (a\ A\ B\ C \longrightarrow B\ Out\ A\ C))$

definition *QCongAnNull* ::
$(['p,'p,'p] \Rightarrow bool) \Rightarrow bool$ (‹*QCongAnNull* -› $[99]\ 50$)
where
  $QCongAnNull\ a \equiv$

$QCongA\ a \wedge$
$(\forall\ A\ B\ C.\ (a\ A\ B\ C \longrightarrow \neg\ B\ Out\ A\ C))$

definition *QCongAnFlat* ::
$([{'p},{'p},{'p}] \Rightarrow bool) \Rightarrow bool$ (‹*QCongAnFlat* -› [*99*] *50*)
where
QCongAnFlat a ≡

$QCongA\ a \wedge$
$(\forall\ A\ B\ C.\ (a\ A\ B\ C \longrightarrow \neg\ Bet\ A\ B\ C))$

definition *IsNullAngaP* ::
$([{'p},{'p},{'p}] \Rightarrow bool) \Rightarrow bool$ (‹*IsNullAngaP* -› [*99*] *50*)
where
IsNullAngaP a≡

$QCongAAcute\ a \wedge$
$(\exists\ A\ B\ C.\ (a\ A\ B\ C \wedge B\ Out\ A\ C))$

definition *QCongANull* ::
$([{'p},{'p},{'p}] \Rightarrow bool) \Rightarrow bool$ (‹*QCongANull* -› [*99*] *50*)
where
QCongANull a ≡

$QCongA\ a \wedge$
$(\forall\ A\ B\ C.\ (a\ A\ B\ C \longrightarrow B\ Out\ A\ C))$

definition *AngFlat* ::
$([{'p},\ {'p},\ {'p}] \Rightarrow bool) \Rightarrow bool$ (‹*AngFlat* -› [*99*] *50*)
where
AngFlat a ≡

$QCongA\ a \wedge$
$(\forall\ A\ B\ C.\ (a\ A\ B\ C \longrightarrow Bet\ A\ B\ C))$

2.2 Parallel's definition Postulate

definition *tarski-s-parallel-postulate* ::
bool
(‹*TarskiSParallelPostulate*›)
where
tarski-s-parallel-postulate ≡
$\forall\ A\ B\ C\ D\ T.\ (Bet\ A\ D\ T \wedge Bet\ B\ D\ C \wedge A \neq D) \longrightarrow$
$(\exists\ X\ Y.\ Bet\ A\ B\ X \wedge Bet\ A\ C\ Y \wedge Bet\ X\ T\ Y)$

definition *euclid-5* ::
bool (‹*Euclid5*›)
where
euclid-5 ≡

$\forall\ P\ Q\ R\ S\ T\ U.$
$(BetS\ P\ T\ Q \wedge$
$BetS\ R\ T\ S \wedge$
$BetS\ Q\ U\ R \wedge$
$\neg\ Col\ P\ Q\ S \wedge$
$Cong\ P\ T\ Q\ T \wedge$
$Cong\ R\ T\ S\ T)$
$\longrightarrow$
$(\exists\ I.\ BetS\ S\ Q\ I \wedge BetS\ P\ U\ I)$

definition *euclid-s-parallel-postulate* ::
bool (‹*EuclidSParallelPostulate*›)
where
euclid-s-parallel-postulate ≡

$\forall\ A\ B\ C\ D\ P\ Q\ R.$
$(B\ C\ OS\ A\ D \wedge$
$SAMS\ A\ B\ C\ B\ C\ D \wedge$
$A\ B\ C\ B\ C\ D\ SumA\ P\ Q\ R \wedge$

¬ *Bet P Q R*)
⟶
(∃ *Y*. *B Out A Y* ∧ *C Out D Y*)

definition *playfair-s-postulate* ::
bool
(‹*PlayfairSPostulate*›)
where
playfair-s-postulate ≡

∀ *A1 A2 B1 B2 C1 C2 P*.
(*A1 A2 Par B1 B2* ∧
Col P B1 B2 ∧
A1 A2 Par C1 C2 ∧
Col P C1 C2)
⟶
(*Col C1 B1 B2* ∧ *Col C2 B1 B2*)

3 Propositions

3.1 Congruence properties

lemma *cong-reflexivity*:
shows *Cong A B A B*
using *cong-inner-transitivity cong-pseudo-reflexivity* **by** *blast*

lemma *cong-symmetry*:
assumes *Cong A B C D*
shows *Cong C D A B*
using *assms cong-inner-transitivity cong-reflexivity* **by** *blast*

lemma *cong-transitivity*:
assumes *Cong A B C D* **and** *Cong C D E F*
shows *Cong A B E F*
by (*meson assms*(*1*) *assms*(*2*) *cong-inner-transitivity cong-pseudo-reflexivity*)

lemma *cong-left-commutativity*:
assumes *Cong A B C D*
shows *Cong B A C D*
using *assms cong-inner-transitivity cong-pseudo-reflexivity* **by** *blast*

lemma *cong-right-commutativity*:
assumes *Cong A B C D*
shows *Cong A B D C*
using *assms cong-left-commutativity cong-symmetry* **by** *blast*

lemma *cong-3421*:
assumes *Cong A B C D*
shows *Cong C D B A*
using *assms cong-left-commutativity cong-symmetry* **by** *blast*

lemma *cong-4312*:
assumes *Cong A B C D*
shows *Cong D C A B*
using *assms cong-left-commutativity cong-symmetry* **by** *blast*

lemma *cong-4321*:
assumes *Cong A B C D*
shows *Cong D C B A*
using *assms cong-3421 cong-left-commutativity* **by** *blast*

lemma *cong-trivial-identity*:
shows *Cong A A B B*
using *cong-identity segment-construction* **by** *blast*

lemma *cong-reverse-identity*:

assumes *Cong A A C D*
shows $C = D$
using *assms cong-3421 cong-identity* **by** *blast*

lemma *cong-commutativity*:
assumes *Cong A B C D*
shows *Cong B A D C*
using *assms cong-3421* **by** *blast*

lemma *not-cong-2134*:
assumes ¬ *Cong A B C D*
shows ¬ *Cong B A C D*
using *assms cong-left-commutativity* **by** *blast*

lemma *not-cong-1243*:
assumes ¬ *Cong A B C D*
shows ¬ *Cong A B D C*
using *assms cong-right-commutativity* **by** *blast*

lemma *not-cong-2143*:
assumes ¬ *Cong A B C D*
shows ¬ *Cong B A D C*
using *assms cong-commutativity* **by** *blast*

lemma *not-cong-3412*:
assumes ¬ *Cong A B C D*
shows ¬ *Cong C D A B*
using *assms cong-symmetry* **by** *blast*

lemma *not-cong-4312*:
assumes ¬ *Cong A B C D*
shows ¬ *Cong D C A B*
using *assms cong-3421* **by** *blast*

lemma *not-cong-3421*:
assumes ¬ *Cong A B C D*
shows ¬ *Cong C D B A*
using *assms cong-4312* **by** *blast*

lemma *not-cong-4321*:
assumes ¬ *Cong A B C D*
shows ¬ *Cong D C B A*
using *assms cong-4321* **by** *blast*

lemma *five-segment-with-def*:
assumes $A\ B\ C\ D$ *OFSC* $A'\ B'\ C'\ D'$ **and** $A \neq B$
shows *Cong* $C\ D\ C'\ D'$
using *assms*(*1*) *assms*(*2*) *OFSC-def five-segment* **by** *blast*

lemma *cong-diff*:
assumes $A \neq B$ **and** *Cong A B C D*
shows $C \neq D$
using *assms*(*1*) *assms*(*2*) *cong-identity* **by** *blast*

lemma *cong-diff-2*:
assumes $B \neq A$ **and** *Cong A B C D*
shows $C \neq D$
using *assms*(*1*) *assms*(*2*) *cong-identity* **by** *blast*

lemma *cong-diff-3*:
assumes $C \neq D$ **and** *Cong A B C D*
shows $A \neq B$
using *assms*(*1*) *assms*(*2*) *cong-reverse-identity* **by** *blast*

lemma *cong-diff-4*:
assumes $D \neq C$ **and** *Cong A B C D*

shows *A* ≠ *B*
using *assms*(*1*) *assms*(*2*) *cong-reverse-identity* **by** *blast*

lemma *cong-3-sym*:
assumes *A B C Cong3 A' B' C'*
shows *A' B' C' Cong3 A B C*
using *assms Cong3-def not-cong-3412* **by** *blast*

lemma *cong-3-swap*:
assumes *A B C Cong3 A' B' C'*
shows *B A C Cong3 B' A' C'*
using *assms Cong3-def cong-commutativity* **by** *blast*

lemma *cong-3-swap-2*:
assumes *A B C Cong3 A' B' C'*
shows *A C B Cong3 A' C' B'*
using *assms Cong3-def cong-commutativity* **by** *blast*

lemma *cong3-transitivity*:
assumes *A0 B0 C0 Cong3 A1 B1 C1* **and**
A1 B1 C1 Cong3 A2 B2 C2
shows *A0 B0 C0 Cong3 A2 B2 C2*
by (*meson assms*(*1*) *assms*(*2*) *Cong3-def cong-inner-transitivity not-cong-3412*)

lemma *eq-dec-points*:
shows *A* = *B* ∨ ¬ *A* = *B*
by *simp*

lemma *distinct*:
assumes *P* ≠ *Q*
shows *R* ≠ *P* ∨ *R* ≠ *Q*
using *assms* **by** *simp*

lemma *l2-11*:
assumes *Bet A B C* **and**
Bet A' B' C' **and**
Cong A B A' B' **and**
Cong B C B' C'
shows *Cong A C A' C'*
by (*smt assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *cong-right-commutativity cong-symmetry cong-trivial-identity five-segment*)

lemma *bet-cong3*:
assumes *Bet A B C* **and**
Cong A B A' B'
shows ∃ *C'*. *A B C Cong3 A' B' C'*
by (*meson assms*(*1*) *assms*(*2*) *Cong3-def l2-11 not-cong-3412 segment-construction*)

lemma *construction-uniqueness*:
assumes *Q* ≠ *A* **and**
Bet Q A X **and**
Cong A X B C **and**
Bet Q A Y **and**
Cong A Y B C
shows *X* = *Y*
by (*meson assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *cong-identity cong-inner-transitivity cong-reflexivity five-segment*)

lemma *Cong-cases*:
assumes *Cong A B C D* ∨ *Cong A B D C* ∨ *Cong B A C D* ∨ *Cong B A D C* ∨ *Cong C D A B* ∨ *Cong C D B A* ∨ *Cong D C A B* ∨ *Cong D C B A*
shows *Cong A B C D*
using *assms not-cong-3421 not-cong-4321* **by** *blast*

lemma *Cong-perm* :
assumes *Cong A B C D*
shows *Cong A B C D* ∧ *Cong A B D C* ∧ *Cong B A C D* ∧ *Cong B A D C* ∧ *Cong C D A B* ∧ *Cong C D B A* ∧ *Cong D C A B* ∧ *Cong D C B A*

using *assms not-cong-1243 not-cong-3412* **by** *blast*

3.2 Betweeness properties

lemma *bet-col*:
assumes *Bet A B C*
shows *Col A B C*
by (*simp add*: *assms Col-def*)

lemma *between-trivial*:
shows *Bet A B B*
using *cong-identity segment-construction* **by** *blast*

lemma *between-symmetry*:
assumes *Bet A B C*
shows *Bet C B A*
using *assms between-identity between-trivial inner-pasch* **by** *blast*

lemma *Bet-cases*:
assumes *Bet A B C* ∨ *Bet C B A*
shows *Bet A B C*
using *assms between-symmetry* **by** *blast*

lemma *Bet-perm*:
assumes *Bet A B C*
shows *Bet A B C* ∧ *Bet C B A*
using *assms Bet-cases* **by** *blast*

lemma *between-trivial2*:
shows *Bet A A B*
using *Bet-perm between-trivial* **by** *blast*

lemma *between-equality*:
assumes *Bet A B C* **and** *Bet B A C*
shows $A = B$
using *assms*(*1*) *assms*(*2*) *between-identity inner-pasch* **by** *blast*

lemma *between-equality-2*:
assumes *Bet A B C* **and**
Bet A C B
shows $B = C$
using *assms*(*1*) *assms*(*2*) *between-equality between-symmetry* **by** *blast*

lemma *between-exchange3*:
assumes *Bet A B C* **and**
Bet A C D
shows *Bet B C D*
by (*metis Bet-perm assms*(*1*) *assms*(*2*) *between-identity inner-pasch*)

lemma *bet-neq12--neq*:
assumes *Bet A B C* **and**
$A \neq B$
shows $A \neq C$
using *assms*(*1*) *assms*(*2*) *between-identity* **by** *blast*

lemma *bet-neq21--neq*:
assumes *Bet A B C* **and**
$B \neq A$
shows $A \neq C$
using *assms*(*1*) *assms*(*2*) *between-identity* **by** *blast*

lemma *bet-neq23--neq*:
assumes *Bet A B C* **and**
$B \neq C$
shows $A \neq C$
using *assms*(*1*) *assms*(*2*) *between-identity* **by** *blast*

lemma *bet-neq32--neq*:
  assumes *Bet A B C* **and**
    C ≠ *B*
  shows *A* ≠ *C*
  using *assms(1) assms(2) between-identity* **by** *blast*

lemma *not-bet-distincts*:
  assumes ¬ *Bet A B C*
  shows *A* ≠ *B* ∧ *B* ≠ *C*
  using *assms between-trivial between-trivial2* **by** *blast*

lemma *between-inner-transitivity*:
  assumes *Bet A B D* **and**
    Bet B C D
  shows *Bet A B C*
  using *assms(1) assms(2) Bet-perm between-exchange3* **by** *blast*

lemma *outer-transitivity-between2*:
  assumes *Bet A B C* **and**
    Bet B C D **and**
    B ≠ *C*
  shows *Bet A C D*
proof −
  obtain *X* **where** *Bet A C X* ∧ *Cong C X C D*
    using *segment-construction* **by** *blast*
  thus *?thesis*
    using *assms(1) assms(2) assms(3) between-exchange3 cong-inner-transitivity construction-uniqueness* **by** *blast*
qed

lemma *between-exchange2*:
  assumes *Bet A B D* **and**
    Bet B C D
  shows *Bet A C D*
  using *assms(1) assms(2) between-inner-transitivity outer-transitivity-between2* **by** *blast*

lemma *outer-transitivity-between*:
  assumes *Bet A B C* **and**
    Bet B C D **and**
    B ≠ *C*
  shows *Bet A B D*
  using *assms(1) assms(2) assms(3) between-symmetry outer-transitivity-between2* **by** *blast*

lemma *between-exchange4*:
  assumes *Bet A B C* **and**
    Bet A C D
  shows *Bet A B D*
  using *assms(1) assms(2) between-exchange2 between-symmetry* **by** *blast*

lemma *l3-9-4*:
  assumes *Bet4 A1 A2 A3 A4*
  shows *Bet4 A4 A3 A2 A1*
  using *assms Bet4-def Bet-cases* **by** *blast*

lemma *l3-17*:
  assumes *Bet A B C* **and**
    Bet A′ B′ C **and**
    Bet A P A′
  shows (∃ *Q*. *Bet P Q C* ∧ *Bet B Q B′*)
proof −
  obtain *X* **where** *Bet B′ X A* ∧ *Bet P X C*
    using *Bet-perm assms(2) assms(3) inner-pasch* **by** *blast*
  moreover
  then obtain *Y* **where** *Bet X Y C* ∧ *Bet B Y B′*
    using *Bet-perm assms(1) inner-pasch* **by** *blast*
  ultimately show *?thesis*

using *between-exchange2* **by** *blast*
qed

lemma *lower-dim-ex*:
$\exists$ *A B C*. $\neg$ (*Bet A B C* $\vee$ *Bet B C A* $\vee$ *Bet C A B*)
using *lower-dim* **by** *auto*

lemma *two-distinct-points*:
$\exists$ *X*::'*p*. $\exists$ *Y*::'*p*. *X* $\neq$ *Y*
using *lower-dim-ex not-bet-distincts* **by** *blast*

lemma *point-construction-different*:
$\exists$ *C*. *Bet A B C* $\wedge$ *B* $\neq$ *C*
using *Tarski-neutral-dimensionless.two-distinct-points Tarski-neutral-dimensionless-axioms cong-reverse-identity segment-construction* **by** *blast*

lemma *another-point*:
$\exists$ *B*::'*p*. *A* $\neq$ *B*
using *point-construction-different* **by** *blast*

lemma *Cong-stability*:
assumes $\neg$ $\neg$ *Cong A B C D*
shows *Cong A B C D*
using *assms* **by** *simp*

lemma *l2-11-b*:
assumes *Bet A B C* **and**
Bet A' B' C' **and**
Cong A B A' B' **and**
Cong B C B' C'
shows *Cong A C A' C'*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *l2-11* **by** *auto*

lemma *cong-dec-eq-dec-b*:
assumes $\neg$ *A* $\neq$ *B*
shows *A* = *B*
using *assms*(*1*) **by** *simp*

lemma *BetSEq*:
assumes *BetS A B C*
shows *Bet A B C* $\wedge$ *A* $\neq$ *B* $\wedge$ *A* $\neq$ *C* $\wedge$ *B* $\neq$ *C*
using *assms BetS-def between-identity* **by** *auto*

3.3 Collinearity

3.3.1 Collinearity and betweenness

lemma *l4-2*:
assumes *A B C D IFSC A' B' C' D'*
shows *Cong B D B' D'*
proof *cases*
assume *A* = *C*
thus *?thesis*
by (*metis IFSC-def Tarski-neutral-dimensionless.between-identity Tarski-neutral-dimensionless-axioms assms cong-diff-3*)
next
assume *H1*: *A* $\neq$ *C*
have *H2*: *Bet A B C* $\wedge$ *Bet A' B' C'* $\wedge$
Cong A C A' C' $\wedge$ *Cong B C B' C'* $\wedge$
Cong A D A' D' $\wedge$ *Cong C D C' D'*
using *IFSC-def assms* **by** *auto*
obtain *E* **where** *P1*: *Bet A C E* $\wedge$ *Cong C E A C*
using *segment-construction* **by** *blast*
have *P1A*: *Bet A C E*
using *P1* **by** *simp*
have *P1B*: *Cong C E A C*
using *P1* **by** *simp*
obtain *E'* **where** *P2*: *Bet A' C' E'* $\wedge$ *Cong C' E' C E*

```
    using segment-construction by blast
  have P2A: Bet A' C' E'
    using P2 by simp
  have P2B: Cong C' E' C E
    using P2 by simp
  then have Cong C E C' E'
    using not-cong-3412 by blast
  then have Cong E D E' D'
    using H1 H2 P1 P2 five-segment by blast
  thus ?thesis
   by (smt H1 H2 P1A P1B P2A P2B Tarski-neutral-dimensionless.cong-commutativity Tarski-neutral-dimensionless.cong-diff-3
Tarski-neutral-dimensionless.cong-symmetry Tarski-neutral-dimensionless-axioms between-inner-transitivity between-symmetry
five-segment)
qed

lemma l4-3:
  assumes Bet A B C and
    Bet A' B' C' and
    Cong A C A' C'
    and Cong B C B' C'
  shows Cong A B A' B'
proof -
  have A B C A IFSC A' B' C' A'
    using IFSC-def assms(1) assms(2) assms(3) assms(4) cong-trivial-identity not-cong-2143 by blast
  thus ?thesis
    using l4-2 not-cong-2143 by blast
qed


lemma l4-3-1:
  assumes Bet A B C and
    Bet A' B' C' and
    Cong A B A' B' and
    Cong A C A' C'
  shows Cong B C B' C'
  by (meson assms(1) assms(2) assms(3) assms(4) between-symmetry cong-4321 l4-3)

lemma l4-5:
  assumes Bet A B C and
    Cong A C A' C'
  shows ∃ B'. (Bet A' B' C' ∧ A B C Cong3 A' B' C')
proof -
  obtain X' where P1: Bet C' A' X' ∧ A' ≠ X'
    using point-construction-different by auto
  obtain B' where P2: Bet  X' A' B' ∧ Cong A' B' A B
    using segment-construction by blast
  obtain C'' where P3: Bet X' B' C'' ∧ Cong B' C'' B C
    using segment-construction by blast
  then have P4: Bet A' B' C''
    using P2 between-exchange3 by blast
  then have C'' = C'
    by (smt P1 P2 P3 assms(1) assms(2) between-exchange4 between-symmetry cong-symmetry construction-uniqueness
l2-11-b)
  then show ?thesis
   by (smt Cong3-def P1 P2 P3 Tarski-neutral-dimensionless.construction-uniqueness Tarski-neutral-dimensionless-axioms
P4 assms(1) assms(2) between-exchange4 between-symmetry cong-commutativity cong-symmetry cong-trivial-identity
five-segment not-bet-distincts)
qed

lemma l4-6:
  assumes Bet A B C and
    A B C Cong3 A' B' C'
  shows Bet A' B' C'
proof -
  obtain x where P1: Bet A' x C' ∧ A B C Cong3 A' x C'
    using Cong3-def assms(1) assms(2) l4-5 by blast
```

then have *A′ x C′ Cong3 A′ B′ C′*
using *assms*(*2*) *cong3-transitivity cong-3-sym* **by** *blast*
then have *A′ x C′ x IFSC A′ x C′ B′*
by (*meson Cong3-def Cong-perm IFSC-def P1 cong-reflexivity*)
then have *Cong x x x B′*
using *l4-2* **by** *auto*
then show *?thesis*
using *P1 cong-reverse-identity* **by** *blast*
qed

lemma *cong3-bet-eq*:
assumes *Bet A B C* **and**
A B C Cong3 A X C
shows $X = B$
proof −
have *A B C B IFSC A B C X*
by (*meson Cong3-def Cong-perm IFSC-def assms*(*1*) *assms*(*2*) *cong-reflexivity*)
then show *?thesis*
using *cong-reverse-identity l4-2* **by** *blast*
qed

3.3.2 Collinearity

lemma *col-permutation-1*:
assumes *Col A B C*
shows *Col B C A*
using *assms*(*1*) *Col-def* **by** *blast*

lemma *col-permutation-2*:
assumes *Col A B C*
shows *Col C A B*
using *assms*(*1*) *col-permutation-1* **by** *blast*

lemma *col-permutation-3*:
assumes *Col A B C*
shows *Col C B A*
using *assms*(*1*) *Bet-cases Col-def* **by** *auto*

lemma *col-permutation-4*:
assumes *Col A B C*
shows *Col B A C*
using *assms*(*1*) *Bet-perm Col-def* **by** *blast*

lemma *col-permutation-5*:
assumes *Col A B C*
shows *Col A C B*
using *assms*(*1*) *col-permutation-1 col-permutation-3* **by** *blast*

lemma *not-col-permutation-1*:
assumes ¬ *Col A B C*
shows ¬ *Col B C A*
using *assms col-permutation-2* **by** *blast*

lemma *not-col-permutation-2*:
assumes ~ *Col A B C*
shows ~ *Col C A B*
using *assms col-permutation-1* **by** *blast*

lemma *not-col-permutation-3*:
assumes ¬ *Col A B C*
shows ¬ *Col C B A*
using *assms col-permutation-3* **by** *blast*

lemma *not-col-permutation-4*:
assumes ¬ *Col A B C*
shows ¬ *Col B A C*

using *assms col-permutation-4* **by** *blast*

lemma *not-col-permutation-5*:
assumes ¬ *Col A B C*
shows ¬ *Col A C B*
using *assms col-permutation-5* **by** *blast*

lemma *Col-cases*:
assumes *Col A B C* ∨ *Col A C B* ∨ *Col B A C* ∨ *Col B C A* ∨ *Col C A B* ∨ *Col C B A*
shows *Col A B C*
using *assms not-col-permutation-4 not-col-permutation-5* **by** *blast*

lemma *Col-perm*:
assumes *Col A B C*
shows *Col A B C* ∧ *Col A C B* ∧ *Col B A C* ∧ *Col B C A* ∧ *Col C A B* ∧ *Col C B A*
using *Col-cases assms* **by** *blast*

lemma *col-trivial-1*:
Col A A B
using *bet-col not-bet-distincts* **by** *blast*

lemma *col-trivial-2*:
Col A B B
by (*simp add*: *Col-def between-trivial2*)

lemma *col-trivial-3*:
Col A B A
by (*simp add*: *Col-def between-trivial2*)

lemma *l4-13*:
assumes *Col A B C* **and**
A B C Cong3 A′ B′ C′
shows *Col A′ B′ C′*
by (*metis Tarski-neutral-dimensionless.Col-def Tarski-neutral-dimensionless.cong-3-swap Tarski-neutral-dimensionless.cong-3-swap-2 Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*2*) *l4-6*)

lemma *l4-14R1*:
assumes *Bet A B C* **and**
Cong A B A′ B′
shows ∃ *C′*. *A B C Cong3 A′ B′ C′*
by (*simp add*: *assms*(*1*) *assms*(*2*) *bet-cong3*)

lemma *l4-14R2*:
assumes *Bet B C A* **and**
Cong A B A′ B′
shows ∃ *C′*. *A B C Cong3 A′ B′ C′*
by (*meson assms*(*1*) *assms*(*2*) *between-symmetry cong-3-swap-2 l4-5*)

lemma *l4-14R3*:
assumes *Bet C A B* **and**
Cong A B A′ B′
shows ∃ *C′*. *A B C Cong3 A′ B′ C′*
by (*meson assms*(*1*) *assms*(*2*) *between-symmetry cong-3-swap l4-14R1 not-cong-2143*)

lemma *l4-14*:
assumes *Col A B C* **and**
Cong A B A′ B′
shows ∃ *C′*. *A B C Cong3 A′ B′ C′*
using *Col-def assms*(*1*) *assms*(*2*) *l4-14R1 l4-14R2 l4-14R3* **by** *blast*

lemma *l4-16R1*:
assumes *A B C D FSC A′ B′ C′ D′* **and**
A ≠ *B* **and**
Bet A B C
shows *Cong C D C′ D′*
proof −

```
  have A B C Cong3 A' B' C'
    using FSC-def assms(1) by blast
  then have Bet A' B' C'
    using assms(3) l4-6 by blast
  then have A B C D OFSC A' B' C' D'
    by (meson Cong3-def FSC-def OFSC-def assms(1) cong-3-sym l4-6)
  thus ?thesis
    using assms(2) five-segment-with-def by blast
qed

lemma l4-16R2:
  assumes A B C D FSC A' B' C' D'
    and Bet B C A
  shows Cong C D C' D'
proof -
  have A B C Cong3 A' B' C'
    using FSC-def assms(1) by blast
  then have Bet B' C' A'
    using Bet-perm assms(2) cong-3-swap-2 l4-6 by blast
  then have B C A D IFSC B' C' A' D'
    by (meson Cong3-def FSC-def IFSC-def assms(1) assms(2) not-cong-2143)
  then show ?thesis
    using l4-2 by auto
qed

lemma l4-16R3:
  assumes A B C D FSC A' B' C' D' and A ≠ B
    and Bet C A B
  shows Cong C D C' D'
proof -
  have A B C Cong3 A' B' C'
    using FSC-def assms(1) by blast
  then have Bet C' A' B'
    using assms(3) between-symmetry cong-3-swap l4-6 by blast
  thus ?thesis
    by (smt Cong3-def FSC-def assms(1) assms(2) assms(3) between-symmetry cong-commutativity five-segment)
qed

lemma l4-16:
  assumes A B C D FSC A' B' C' D' and
    A ≠ B
  shows Cong C D C' D'
  by (meson Col-def FSC-def assms(1) assms(2) l4-16R1 l4-16R2 l4-16R3)

lemma l4-17:
  assumes A ≠ B and
    Col A B C and
    Cong A P A Q and
    Cong B P B Q
  shows Cong C P C Q
proof -
  {
    assume ¬ Bet B C A
    then have ∃ p pa. Bet p pa C ∧ Cong pa P pa Q ∧ Cong p P p Q ∧ p ≠ pa
      using Col-def assms(1) assms(2) assms(3) assms(4) between-symmetry by blast
    then have ?thesis
      using cong-reflexivity five-segment by blast
  }
  then show ?thesis
    by (meson IFSC-def assms(3) assms(4) cong-reflexivity l4-2)
qed


lemma l4-18:
  assumes A ≠ B and
    Col A B C and
```

Cong A C A C′ **and**
Cong B C B C′
shows $C = C'$
using *assms(1) assms(2) assms(3) assms(4) cong-diff-3 l4-17* **by** *blast*

lemma *l4-19*:
assumes *Bet A C B* **and**
Cong A C A C′ **and**
Cong B C B C′
shows $C = C'$
by (*metis Col-def assms(1) assms(2) assms(3) between-equality between-trivial cong-identity l4-18 not-cong-3421*)

lemma *not-col-distincts*:
assumes ¬ *Col A B C*
shows ¬ *Col A B C* ∧ $A \neq B$ ∧ $B \neq C$ ∧ $A \neq C$
using *Col-def assms between-trivial* **by** *blast*

lemma *NCol-cases*:
assumes ¬ *Col A B C* ∨ ¬ *Col A C B* ∨ ¬ *Col B A C* ∨ ¬ *Col B C A* ∨ ¬ *Col C A B* ∨ ¬ *Col C B A*
shows ¬ *Col A B C*
using *assms not-col-permutation-2 not-col-permutation-3* **by** *blast*

lemma *NCol-perm*:
assumes ¬ *Col A B C*
shows ¬ *Col A B C* ∧ ~ *Col A C B* ∧ ~ *Col B A C* ∧ ~ *Col B C A* ∧ ~ *Col C A B* ∧ ~ *Col C B A*
using *NCol-cases assms* **by** *blast*

lemma *col-cong-3-cong-3-eq*:
assumes $A \neq B$
and *Col A B C*
and *A B C Cong3 A′ B′ C1*
and *A B C Cong3 A′ B′ C2*
shows $C1 = C2$
by (*metis Tarski-neutral-dimensionless.Cong3-def Tarski-neutral-dimensionless.cong-diff Tarski-neutral-dimensionless.l4-18 Tarski-neutral-dimensionless-axioms assms(1) assms(2) assms(3) assms(4) cong-inner-transitivity l4-13*)

3.4 Between transitivity le

lemma *l5-1*:
assumes $A \neq B$ **and**
Bet A B C **and**
Bet A B D
shows *Bet A C D* ∨ *Bet A D C*
proof −
obtain *C′* **where** *P1*: *Bet A D C′* ∧ *Cong D C′ C D*
using *segment-construction* **by** *blast*
obtain *D′* **where** *P2*: *Bet A C D′* ∧ *Cong C D′ C D*
using *segment-construction* **by** *blast*
obtain *B′* **where** *P3*: *Bet A C′ B′* ∧ *Cong C′ B′ C B*
using *segment-construction* **by** *blast*
obtain *B′′* **where** *P4*: *Bet A D′ B′′* ∧ *Cong D′ B′′ D B*
using *segment-construction* **by** *blast*
then have *P5*: *Cong B C′ B′′ C*
by (*smt P1 P2 assms(3) between-exchange3 between-symmetry cong-4312 cong-inner-transitivity l2-11-b*)
then have *Cong B B′ B′′ B*
by (*meson Bet-cases P1 P2 P3 P4 assms(2) assms(3) between-exchange4 between-inner-transitivity l2-11-b*)
then have *P6*: $B'' = B'$
by (*meson P1 P2 P3 P4 assms(1) assms(2) assms(3) between-exchange4 cong-inner-transitivity construction-uniqueness not-cong-2134*)
have *Bet B C D′*
using *P2 assms(2) between-exchange3* **by** *blast*
then have *B C D′ C′ FSC B′ C′ D C*
by (*smt Cong3-def FSC-def P1 P2 P3 P5 P6 bet-col between-exchange3 between-symmetry cong-3421 cong-pseudo-reflexivity cong-transitivity l2-11-b*)
then have *P8*: *Cong D′ C′ D C*
using *P3 P4 P6 cong-identity l4-16* **by** *blast*

obtain *E* **where** *P9*: *Bet C E C′* ∧ *Bet D E D′*
using *P1 P2 between-trivial2 l3-17* **by** *blast*
then have *P10*: *D E D′ C IFSC D E D′ C′*
by (*smt IFSC-def P1 P2 P8 Tarski-neutral-dimensionless.cong-reflexivity Tarski-neutral-dimensionless-axioms cong-3421 cong-inner-transitivity*)
then have *Cong E C E C′*
using *l4-2* **by** *auto*
have *P11*: *C E C′ D IFSC C E C′ D′*
by (*smt IFSC-def P1 P2 Tarski-neutral-dimensionless.cong-reflexivity Tarski-neutral-dimensionless-axioms P8 P9 cong-3421 cong-inner-transitivity*)
then have *Cong E D E D′*
using *l4-2* **by** *auto*
obtain *P* **where** *Bet C′ C P* ∧ *Cong C P C D′*
using *segment-construction* **by** *blast*
obtain *R* **where** *Bet D′ C R* ∧ *Cong C R C E*
using *segment-construction* **by** *blast*
obtain *Q* **where** *Bet P R Q* ∧ *Cong R Q R P*
using *segment-construction* **by** *blast*
have *D′ C R P FSC P C E D′*
by (*meson Bet-perm Cong3-def FSC-def* ‹*Bet C E C′* ∧ *Bet D E D′*› ‹*Bet C′ C P* ∧ *Cong C P C D′*› ‹*Bet D′ C R* ∧ *Cong C R C E*› *bet-col between-exchange3 cong-pseudo-reflexivity l2-11-b not-cong-4321*)
have *Cong R P E D′*
by (*metis Cong-cases* ‹*D′ C R P FSC P C E D′*› ‹*Bet C′ C P* ∧ *Cong C P C D′*› ‹*Bet D′ C R* ∧ *Cong C R C E*› *cong-diff-2 l4-16*)
have *Cong R Q E D*
by (*metis Cong-cases* ‹*Cong E D E D′*› ‹*Cong R P E D′*› ‹*Bet P R Q* ∧ *Cong R Q R P*› *cong-transitivity*)
have *D′ E D C FSC P R Q C*
by (*meson Bet-perm Cong3-def FSC-def* ‹*Cong R P E D′*› ‹*Cong R Q E D*› ‹*Bet C E C′* ∧ *Bet D E D′*› ‹*Bet C′ C P* ∧ *Cong C P C D′*› ‹*Bet D′ C R* ∧ *Cong C R C E*› ‹*Bet P R Q* ∧ *Cong R Q R P*› *bet-col l2-11-b not-cong-2143 not-cong-4321*)
have *Cong D C Q C*
using ‹*D′ E D C FSC P R Q C*› ‹*Cong E D E D′*› ‹*Bet C E C′* ∧ *Bet D E D′*› *cong-identity l4-16 l4-16R2* **by** *blast*
have *Cong C P C Q*
using *P2* ‹*Cong D C Q C*› ‹*Bet C′ C P* ∧ *Cong C P C D′*› *cong-right-commutativity cong-transitivity* **by** *blast*
have *Bet A C D* ∨ *Bet A D C*
proof *cases*
assume *R* = *C*
then show *?thesis*
by (*metis P1* ‹*Cong E C E C′*› ‹*Bet D′ C R* ∧ *Cong C R C E*› *cong-diff-4*)
next
assume *R* ≠ *C*
{
have *Cong D′ P D′ Q*
proof −

have *Col R C D′*
by (*simp add*: ‹*Bet D′ C R* ∧ *Cong C R C E*› *bet-col between-symmetry*)
have *Cong R P R Q*
by (*metis Tarski-neutral-dimensionless.Cong-cases Tarski-neutral-dimensionless-axioms* ‹*Bet P R Q* ∧ *Cong R Q R P*›)
have *Cong C P C Q*
by (*simp add*: ‹*Cong C P C Q*›)
then show *?thesis*
using ‹*Col R C D′*› ‹*Cong R P R Q*› ‹*R* ≠ *C*› *l4-17* **by** *blast*
qed
then have *Cong B P B Q* **using** ‹*Cong C P C Q*› ‹*Bet B C D′*› *cong-diff-4*
by (*metis Col-def* ‹*Bet C′ C P* ∧ *Cong C P C D′*› *cong-reflexivity l4-17 not-cong-3412*)
have *Cong B′ P B′ Q*
by (*metis P2 P4* ‹*B″* = *B′*› ‹*Cong C P C Q*› ‹*Cong D′ P D′ Q*› ‹*Bet C′ C P* ∧ *Cong C P C D′*› *between-exchange3 cong-diff-4 cong-identity cong-reflexivity five-segment*)
have *Cong C′ P C′ Q*
proof −
have *Bet B C′ B′*
using *P1 P3 assms*(*3*) *between-exchange3 between-exchange4* **by** *blast*
then show *?thesis*
by (*metis Col-def* ‹*Cong B P B Q*› ‹*Cong B′ P B′ Q*› *between-equality l4-17 not-bet-distincts*)

qed
have *Cong P P P Q*
by (*metis Tarski-neutral-dimensionless.cong-diff-2 Tarski-neutral-dimensionless-axioms* ‹*Cong C P C Q*› ‹*Cong C′ P C′ Q*› ‹*R* ≠ *C*› ‹*Bet C E C′* ∧ *Bet D E D′*› ‹*Bet C′ C P* ∧ *Cong C P C D′*› ‹*Bet D′ C R* ∧ *Cong C R C E*› *bet-col bet-neq12--neq l4-17*)
thus *?thesis*
by (*metis P2* ‹*Cong R P E D′*› ‹*Cong R Q E D*› ‹*Bet P R Q* ∧ *Cong R Q R P*› *bet-neq12--neq cong-diff-4*)
}
then have *R* ≠ *C* ⟶ *Bet A C D* ∨ *Bet A D C* **by** *blast*
qed
thus *?thesis*
by *simp*
qed

lemma *l5-2*:
assumes *A* ≠ *B* **and**
Bet A B C **and**
Bet A B D
shows *Bet B C D* ∨ *Bet B D C*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *between-exchange3 l5-1* **by** *blast*

lemma *segment-construction-2*:
assumes *A* ≠ *Q*
shows ∃ *X*. ((*Bet Q A X* ∨ *Bet Q X A*) ∧ *Cong Q X B C*)
proof −
obtain *A′* **where** *P1*: *Bet A Q A′* ∧ *Cong Q A′ A Q*
using *segment-construction* **by** *blast*
obtain *X* **where** *P2*: *Bet A′ Q X* ∧ *Cong Q X B C*
using *segment-construction* **by** *blast*
then show *?thesis*
by (*metis P1 Tarski-neutral-dimensionless.cong-diff-4 Tarski-neutral-dimensionless-axioms between-symmetry l5-2*)
qed

lemma *l5-3*:
assumes *Bet A B D* **and**
Bet A C D
shows *Bet A B C* ∨ *Bet A C B*
by (*metis Bet-perm assms*(*1*) *assms*(*2*) *between-inner-transitivity l5-2 point-construction-different*)

lemma *bet3--bet*:
assumes *Bet A B E* **and**
Bet A D E **and**
Bet B C D
shows *Bet A C E*
by (*meson assms*(*1*) *assms*(*2*) *assms*(*3*) *between-exchange2 between-symmetry l5-3*)

lemma *le-bet*:
assumes *C D Le A B*
shows ∃ *X*. (*Bet A X B* ∧ *Cong A X C D*)
by (*meson Le-def assms cong-symmetry*)

lemma *l5-5-1*:
assumes *A B Le C D*
shows ∃ *X*. (*Bet A B X* ∧ *Cong A X C D*)
proof −
obtain *P* **where** *P1*: *Bet C P D* ∧ *Cong A B C P*
using *Le-def assms* **by** *blast*
obtain *X* **where** *P2*: *Bet A B X* ∧ *Cong B X P D*
using *segment-construction* **by** *blast*
then have *Cong A X C D*
using *P1 l2-11-b* **by** *blast*
then show *?thesis*
using *P2* **by** *blast*
qed

lemma *l5-5-2*:

assumes ∃ *X*. (*Bet A B X* ∧ *Cong A X C D*)
shows *A B Le C D*
proof −
obtain *P* **where** *P1*: *Bet A B P* ∧ *Cong A P C D*
using *assms* **by** *blast*
obtain *B′* **where** *P2*: *Bet C B′ D* ∧ *A B P Cong3 C B′ D*
using *P1 l4-5* **by** *blast*
then show *?thesis*
using *Cong3-def Le-def* **by** *blast*
qed

lemma *l5-6*:
assumes *A B Le C D* **and**
Cong A B A′ B′ **and**
Cong C D C′ D′
shows *A′ B′ Le C′ D′*
by (*meson Cong3-def Le-def assms*(*1*) *assms*(*2*) *assms*(*3*) *cong-inner-transitivity l4-5*)

lemma *le-reflexivity*:
shows *A B Le A B*
using *between-trivial cong-reflexivity l5-5-2* **by** *blast*

lemma *le-transitivity*:
assumes *A B Le C D* **and**
C D Le E F
shows *A B Le E F*
by (*meson assms*(*1*) *assms*(*2*) *between-exchange4 cong-reflexivity l5-5-1 l5-5-2 l5-6 le-bet*)

lemma *between-cong*:
assumes *Bet A C B* **and**
Cong A C A B
shows *C* = *B*
by (*smt assms*(*1*) *assms*(*2*) *between-trivial cong-inner-transitivity five-segment l4-19 l4-3-1*)

lemma *cong3-symmetry*:
assumes *A B C Cong3 A′ B′ C′*
shows *A′ B′ C′ Cong3 A B C*
by (*simp add*: *assms cong-3-sym*)

lemma *between-cong-2*:
assumes *Bet A D B* **and**
Bet A E B
and *Cong A D A E*
shows *D* = *E* **using** *l5-3*
by (*smt Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*2*) *assms*(*3*) *cong-diff cong-inner-transitivity l4-3-1*)

lemma *between-cong-3*:
assumes *A* ≠ *B*
and *Bet A B D*
and *Bet A B E*
and *Cong B D B E*
shows *D* = *E*
by (*meson assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *cong-reflexivity construction-uniqueness*)

lemma *le-anti-symmetry*:
assumes *A B Le C D* **and**
C D Le A B
shows *Cong A B C D*
by (*smt Le-def Tarski-neutral-dimensionless.between-exchange4 Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*2*) *bet-neq21--neq between-cong between-exchange3 cong-transitivity l5-5-1 not-cong-3421*)

lemma *cong-dec*:
shows *Cong A B C D* ∨ ¬ *Cong A B C D*
by *simp*

lemma *bet-dec*:

shows *Bet A B C* ∨ ¬ *Bet A B C*
by *simp*

lemma *col-dec*:
shows *Col A B C* ∨ ¬ *Col A B C*
by *simp*

lemma *le-trivial*:
shows *A A Le C D*
using *Le-def between-trivial2 cong-trivial-identity* **by** *blast*

lemma *le-cases*:
shows *A B Le C D* ∨ *C D Le A B*
by (*metis* (*full-types*) *cong-reflexivity l5-5-2 l5-6 not-bet-distincts segment-construction-2*)

lemma *le-zero*:
assumes *A B Le C C*
shows *A* = *B*
by (*metis assms cong-diff-4 le-anti-symmetry le-trivial*)

lemma *le-diff*:
assumes *A* ≠ *B* **and** *A B Le C D*
shows *C* ≠ *D*
using *assms*(*1*) *assms*(*2*) *le-zero* **by** *blast*

lemma *lt-diff*:
assumes *A B Lt C D*
shows *C* ≠ *D*
using *Lt-def assms cong-trivial-identity le-zero* **by** *blast*

lemma *bet-cong-eq*:
assumes *Bet A B C* **and**
Bet A C D **and**
Cong B C A D
shows *C* = *D* ∧ *A* = *B*
proof −
have *Bet C B A*
using *Bet-perm assms*(*1*) **by** *blast*
then show *?thesis*
by (*metis* (*no-types*) *Cong-perm Le-def assms*(*2*) *assms*(*3*) *between-cong cong-pseudo-reflexivity le-anti-symmetry*)
qed

lemma *cong--le*:
assumes *Cong A B C D*
shows *A B Le C D*
using *Le-def assms between-trivial* **by** *blast*

lemma *cong--le3412*:
assumes *Cong A B C D*
shows *C D Le A B*
using *assms cong--le cong-symmetry* **by** *blast*

lemma *le1221*:
shows *A B Le B A*
by (*simp add*: *cong--le cong-pseudo-reflexivity*)

lemma *le-left-comm*:
assumes *A B Le C D*
shows *B A Le C D*
using *assms le1221 le-transitivity* **by** *blast*

lemma *le-right-comm*:
assumes *A B Le C D*
shows *A B Le D C*
by (*meson assms cong-right-commutativity l5-5-1 l5-5-2*)

lemma *le-comm*:
assumes *A B Le C D*
shows *B A Le D C*
using *assms le-left-comm le-right-comm* **by** *blast*

lemma *ge-left-comm*:
assumes *A B Ge C D*
shows *B A Ge C D*
by (*meson Ge-def assms le-right-comm*)

lemma *ge-right-comm*:
assumes *A B Ge C D*
shows *A B Ge D C*
using *Ge-def assms le-left-comm* **by** *presburger*

lemma *ge-comm0*:
assumes *A B Ge C D*
shows *B A Ge D C*
by (*meson assms ge-left-comm ge-right-comm*)

lemma *lt-right-comm*:
assumes *A B Lt C D*
shows *A B Lt D C*
using *Lt-def assms le-right-comm not-cong-1243* **by** *blast*

lemma *lt-left-comm*:
assumes *A B Lt C D*
shows *B A Lt C D*
using *Lt-def assms le-comm lt-right-comm not-cong-2143* **by** *blast*

lemma *lt-comm*:
assumes *A B Lt C D*
shows *B A Lt D C*
using *assms lt-left-comm lt-right-comm* **by** *blast*

lemma *gt-left-comm0*:
assumes *A B Gt C D*
shows *B A Gt C D*
by (*meson Gt-def assms lt-right-comm*)

lemma *gt-right-comm*:
assumes *A B Gt C D*
shows *A B Gt D C*
using *Gt-def assms lt-left-comm* **by** *presburger*

lemma *gt-comm*:
assumes *A B Gt C D*
shows *B A Gt D C*
by (*meson assms gt-left-comm0 gt-right-comm*)

lemma *cong2-lt--lt*:
assumes *A B Lt C D* **and**
Cong A B A′ B′ **and**
Cong C D C′ D′
shows *A′ B′ Lt C′ D′*
by (*meson Lt-def assms*(*1*) *assms*(*2*) *assms*(*3*) *l5-6 le-anti-symmetry not-cong-3412*)

lemma *fourth-point*:
assumes $A \neq B$ **and**
$B \neq C$ **and**
Col A B P **and**
Bet A B C
shows *Bet P A B* ∨ *Bet A P B* ∨ *Bet B P C* ∨ *Bet B C P*
by (*metis Col-def Tarski-neutral-dimensionless.l5-2 Tarski-neutral-dimensionless-axioms assms*(*3*) *assms*(*4*) *between-symmetry*)

lemma *third-point*:

assumes *Col A B P*
shows *Bet P A B* ∨ *Bet A P B* ∨ *Bet A B P*
using *Col-def assms between-symmetry* **by** *blast*

lemma *l5-12-a*:
assumes *Bet A B C*
shows *A B Le A C* ∧ *B C Le A C*
using *assms between-symmetry cong-left-commutativity cong-reflexivity l5-5-2 le-left-comm* **by** *blast*

lemma *bet--le1213*:
assumes *Bet A B C*
shows *A B Le A C*
using *assms l5-12-a* **by** *blast*

lemma *bet--le2313*:
assumes *Bet A B C*
shows *B C Le A C*
by (*simp add*: *assms l5-12-a*)

lemma *bet--lt1213*:
assumes $B \neq C$ **and**
Bet A B C
shows *A B Lt A C*
using *Lt-def assms*(*1*) *assms*(*2*) *bet--le1213 between-cong* **by** *blast*

lemma *bet--lt2313*:
assumes $A \neq B$ **and**
Bet A B C
shows *B C Lt A C*
using *Lt-def assms*(*1*) *assms*(*2*) *bet--le2313 bet-cong-eq l5-1* **by** *blast*

lemma *l5-12-b*:
assumes *Col A B C* **and**
A B Le A C **and**
B C Le A C
shows *Bet A B C*
by (*metis assms*(*1*) *assms*(*2*) *assms*(*3*) *between-cong col-permutation-5 l5-12-a le-anti-symmetry not-cong-2143 third-point*)

lemma *bet-le-eq*:
assumes *Bet A B C*
and *A C Le B C*
shows $A = B$
by (*meson assms*(*1*) *assms*(*2*) *bet--le2313 bet-cong-eq l5-1 le-anti-symmetry*)

lemma *or-lt-cong-gt*:
A B Lt C D ∨ *A B Gt C D* ∨ *Cong A B C D*
by (*meson Gt-def Lt-def cong-symmetry local.le-cases*)

lemma *lt--le*:
assumes *A B Lt C D*
shows *A B Le C D*
using *Lt-def assms* **by** *blast*

lemma *le1234-lt--lt*:
assumes *A B Le C D* **and**
C D Lt E F
shows *A B Lt E F*
by (*meson Lt-def assms*(*1*) *assms*(*2*) *cong--le3412 le-anti-symmetry le-transitivity*)

lemma *le3456-lt--lt*:
assumes *A B Lt C D* **and**
C D Le E F
shows *A B Lt E F*
by (*meson Lt-def assms*(*1*) *assms*(*2*) *cong2-lt--lt cong-reflexivity le1234-lt--lt*)

lemma *lt-transitivity*:

assumes *A B Lt C D* **and**
C D Lt E F
shows *A B Lt E F*
using *Lt-def assms(1) assms(2) le1234-lt--lt* **by** *blast*

lemma *not-and-lt*:
¬ (*A B Lt C D* ∧ *C D Lt A B*)
by (*simp add*: *Lt-def le-anti-symmetry*)

lemma *nlt*:
¬ *A B Lt A B*
using *not-and-lt* **by** *blast*

lemma *le--nlt*:
assumes *A B Le C D*
shows ¬ *C D Lt A B*
using *assms le3456-lt--lt nlt* **by** *blast*

lemma *cong--nlt*:
assumes *Cong A B C D*
shows ¬ *A B Lt C D*
by (*simp add*: *Lt-def assms*)

lemma *nlt--le*:
assumes ¬ *A B Lt C D*
shows *C D Le A B*
using *Lt-def assms cong--le3412 local.le-cases* **by** *blast*

lemma *lt--nle*:
assumes *A B Lt C D*
shows ¬ *C D Le A B*
using *assms le--nlt* **by** *blast*

lemma *nle--lt*:
assumes ¬ *A B Le C D*
shows *C D Lt A B*
using *assms nlt--le* **by** *blast*

lemma *lt1123*:
assumes $B \neq C$
shows *A A Lt B C*
using *assms le-diff nle--lt* **by** *blast*

lemma *bet2-le2--le-R1*:
assumes *Bet a P b* **and**
Bet A Q B **and**
P a Le Q A **and**
P b Le Q B **and**
$B = Q$
shows *a b Le A B*
by (*metis assms(3) assms(4) assms(5) le-comm le-diff*)

lemma *bet2-le2--le-R2*:
assumes *Bet a Po b* **and**
Bet A PO B **and**
Po a Le PO A **and**
Po b Le PO B **and**
$A \neq PO$ **and**
$B \neq PO$
shows *a b Le A B*
proof −
obtain b' **where** *P1*: *Bet A PO* b' ∧ *Cong PO* b' *b Po*
using *segment-construction* **by** *blast*
obtain a' **where** *P2*: *Bet B PO* a' ∧ *Cong PO* a' *a Po*
using *segment-construction* **by** *blast*
obtain a'' **where** *P3*: *Bet PO* a'' *A* ∧ *Cong Po a PO* a''

using *Le-def assms(3)* **by** *blast*
have *P4*: $a' = a''$
by (*meson Bet-cases P2 P3 assms(2) assms(6) between-inner-transitivity cong-right-commutativity construction-uniqueness not-cong-3412*)
have *P5*: *B* a' *Le B A*
using *Bet-cases P3 P4 assms(2) bet--le1213 between-exchange2* **by** *blast*
obtain b'' **where** *P6*: *Bet PO* b'' *B* ∧ *Cong Po b PO* b''
using *Le-def assms(4)* **by** *blast*
then have $b' = b''$
using *P1 assms(2) assms(5) between-inner-transitivity cong-right-commutativity construction-uniqueness not-cong-3412* **by** *blast*
then have a' b' *Le* a' *B*
using *Bet-cases P2 P6 bet--le1213 between-exchange2* **by** *blast*
then have a' b' *Le A B*
using *P5 le-comm le-transitivity* **by** *blast*
thus *?thesis*
by (*smt Cong-cases P1 P3 P4 Tarski-neutral-dimensionless.l5-6 Tarski-neutral-dimensionless-axioms assms(1) between-exchange3 between-symmetry cong-reflexivity l2-11-b*)
qed

lemma *bet2-le2--le*:
assumes *Bet a P b* **and**
Bet A Q B **and**
P a Le Q A **and**
P b Le Q B
shows *a b Le A B*
proof *cases*
assume *A* = *Q*
thus *?thesis*
using *assms(3) assms(4) le-diff* **by** *force*
next
assume ¬ *A* = *Q*
thus *?thesis*
using *assms(1) assms(2) assms(3) assms(4) bet2-le2--le-R1 bet2-le2--le-R2* **by** *blast*
qed

lemma *Le-cases*:
assumes *A B Le C D* ∨ *B A Le C D* ∨ *A B Le D C* ∨ *B A Le D C*
shows *A B Le C D*
using *assms le-left-comm le-right-comm* **by** *blast*

lemma *Lt-cases*:
assumes *A B Lt C D* ∨ *B A Lt C D* ∨ *A B Lt D C* ∨ *B A Lt D C*
shows *A B Lt C D*
using *assms lt-comm lt-left-comm* **by** *blast*

3.5 Out lines

lemma *bet-out*:
assumes *B* ≠ *A* **and**
Bet A B C
shows *A Out B C*
using *Out-def assms(1) assms(2) bet-neq12--neq* **by** *fastforce*

lemma *bet-out-1*:
assumes *B* ≠ *A* **and**
Bet C B A
shows *A Out B C*
by (*simp add*: *assms(1) assms(2) bet-out between-symmetry*)

lemma *out-dec*:
shows *P Out A B* ∨ ¬ *P Out A B*
by *simp*

lemma *out-diff1*:
assumes *A Out B C*

shows $B \neq A$
using *Out-def assms* **by** *auto*

lemma *out-diff2*:
assumes A *Out* B C
shows $C \neq A$
using *Out-def assms* **by** *auto*

lemma *out-distinct*:
assumes A *Out* B C
shows $B \neq A \wedge C \neq A$
using *assms out-diff1 out-diff2* **by** *auto*

lemma *out-col*:
assumes A *Out* B C
shows *Col* A B C
using *Col-def Out-def assms between-symmetry* **by** *auto*

lemma *l6-2*:
assumes $A \neq P$ **and**
$B \neq P$ **and**
$C \neq P$ **and**
Bet A P C
shows *Bet* B P $C \longleftrightarrow P$ *Out* A B
by (*smt Bet-cases Out-def assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *between-inner-transitivity l5-2 outer-transitivity-between*)

lemma *bet-out--bet*:
assumes *Bet* A P C **and**
P *Out* A B
shows *Bet* B P C
by (*metis Tarski-neutral-dimensionless.l6-2 Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*2*) *not-bet-distincts out-diff1*)

lemma *l6-3-1*:
assumes P *Out* A B
shows $A \neq P \wedge B \neq P \wedge (\exists\ C.\ (C \neq P \wedge Bet\ A\ P\ C \wedge Bet\ B\ P\ C))$
using *assms bet-out--bet out-diff1 out-diff2 point-construction-different* **by** *fastforce*

lemma *l6-3-2*:
assumes $A \neq P$ **and**
$B \neq P$ **and**
$\exists\ C.\ (C \neq P \wedge Bet\ A\ P\ C \wedge Bet\ B\ P\ C)$
shows P *Out* A B
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *l6-2* **by** *blast*

lemma *l6-4-1*:
assumes P *Out* A B **and**
Col A P B
shows $\neg$ *Bet* A P B
using *Out-def assms*(*1*) *between-equality between-symmetry* **by** *fastforce*

lemma *l6-4-2*:
assumes *Col* A P B
and $\neg$ *Bet* A P B
shows P *Out* A B
by (*metis Out-def assms*(*1*) *assms*(*2*) *bet-out col-permutation-1 third-point*)

lemma *out-trivial*:
assumes $A \neq P$
shows P *Out* A A
by (*simp add*: *assms bet-out-1 between-trivial2*)

lemma *l6-6*:
assumes P *Out* A B
shows P *Out* B A
using *Out-def assms* **by** *auto*

lemma *l6-7*:
  assumes *P Out A B* **and**
    P Out B C
  shows *P Out A C*
  by (*smt Out-def assms*(*1*) *assms*(*2*) *between-exchange4 l5-1 l5-3*)

lemma *bet-out-out-bet*:
  assumes *Bet A B C* **and**
    B Out A A' **and**
    B Out C C'
  shows *Bet A' B C'*
  by (*metis Out-def assms*(*1*) *assms*(*2*) *assms*(*3*) *bet-out--bet between-inner-transitivity outer-transitivity-between*)

lemma *out2-bet-out*:
  assumes *B Out A C* **and**
    B Out X P **and**
    Bet A X C
  shows *B Out A P* ∧ *B Out C P*
  by (*smt Out-def Tarski-neutral-dimensionless.l6-7 Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*2*) *assms*(*3*) *between-exchange2 between-symmetry*)

lemma *l6-11-uniqueness*:
  assumes *A Out X R* **and**
    Cong A X B C **and**
    A Out Y R **and**
    Cong A Y B C
  shows *X* = *Y*
  by (*metis Out-def assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *between-cong cong-symmetry cong-transitivity l6-6 l6-7*)

lemma *l6-11-existence*:
  assumes *R* ≠ *A* **and**
    B ≠ *C*
  shows ∃ *X*. (*A Out X R* ∧ *Cong A X B C*)
  by (*metis Out-def assms*(*1*) *assms*(*2*) *cong-reverse-identity segment-construction-2*)

lemma *segment-construction-3*:
  assumes *A* ≠ *B* **and**
    X ≠ *Y*
  shows ∃ *C*. (*A Out B C* ∧ *Cong A C X Y*)
  by (*metis assms*(*1*) *assms*(*2*) *l6-11-existence l6-6*)

lemma *l6-13-1*:
  assumes *P Out A B* **and**
    P A Le P B
  shows *Bet P A B*
  by (*metis Out-def assms*(*1*) *assms*(*2*) *bet--lt1213 le--nlt*)

lemma *l6-13-2*:
  assumes *P Out A B* **and**
    Bet P A B
  shows *P A Le P B*
  by (*simp add*: *assms*(*2*) *bet--le1213*)

lemma *l6-16-1*:
  assumes *P* ≠ *Q* **and**
    Col S P Q **and**
    Col X P Q
  shows *Col X P S*
  by (*smt Col-def assms*(*1*) *assms*(*2*) *assms*(*3*) *bet3--bet col-permutation-4 l5-1 l5-3 outer-transitivity-between third-point*)

lemma *col-transitivity-1*:
  assumes *P* ≠ *Q* **and**
    Col P Q A **and**
    Col P Q B

shows *Col P A B*
by (*meson Tarski-neutral-dimensionless.l6-16-1 Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*2*) *assms*(*3*) *not-col-permutation-2*)

lemma *col-transitivity-2*:
assumes $P \neq Q$ **and**
Col P Q A **and**
Col P Q B
shows *Col Q A B*
by (*metis Tarski-neutral-dimensionless.col-transitivity-1 Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*2*) *assms*(*3*) *not-col-permutation-4*)

lemma *l6-21*:
assumes ¬ *Col A B C* **and**
$C \neq D$ **and**
Col A B P **and**
Col A B Q **and**
Col C D P **and**
Col C D Q
shows $P = Q$
by (*metis assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *assms*(*6*) *col-transitivity-1 l6-16-1 not-col-distincts*)

lemma *col2--eq*:
assumes *Col A X Y* **and**
Col B X Y **and**
¬ *Col A X B*
shows $X = Y$
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *l6-16-1* **by** *blast*

lemma *not-col-exists*:
assumes $A \neq B$
shows ∃ *C*. ¬ *Col A B C*
by (*metis Col-def assms col-transitivity-2 lower-dim-ex*)

lemma *col3*:
assumes $X \neq Y$ **and**
Col X Y A **and**
Col X Y B **and**
Col X Y C
shows *Col A B C*
by (*metis assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *col-transitivity-2*)

lemma *colx*:
assumes $A \neq B$ **and**
Col X Y A **and**
Col X Y B **and**
Col A B C
shows *Col X Y C*
by (*metis assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *l6-21 not-col-distincts*)

lemma *out2--bet*:
assumes *A Out B C* **and**
C Out A B
shows *Bet A B C*
by (*metis Out-def assms*(*1*) *assms*(*2*) *between-equality between-symmetry*)

lemma *bet2-le2--le1346*:
assumes *Bet A B C* **and**
Bet A' B' C' **and**
A B Le A' B' **and**
B C Le B' C'
shows *A C Le* A' C'
using *Le-cases assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *bet2-le2--le* **by** *blast*

lemma *bet2-le2--le2356-R1*:
assumes *Bet A A C* **and**

Bet A' B' C' **and**
A A Le A' B' **and**
A' C' Le A C
shows *B' C' Le A C*
using *assms(2) assms(4) bet--le2313 le3456-lt--lt lt--nle nlt--le* **by** *blast*

lemma *bet2-le2--le2356-R2*:
assumes $A \neq B$ **and**
Bet A B C **and**
Bet A' B' C' **and**
A B Le A' B' **and**
A' C' Le A C
shows *B' C' Le B C*
proof −
have $A \neq C$
using *assms(1) assms(2) bet-neq12--neq* **by** *blast*
obtain *B0* **where** *P1*: *Bet A B B0* ∧ *Cong A B0 A' B'*
using *assms(4) l5-5-1* **by** *blast*
then have *P2*: $A \neq B0$
using *assms(1) bet-neq12--neq* **by** *blast*
obtain *C0* **where** *P3*: *Bet A C0 C* ∧ *Cong A' C' A C0*
using *Le-def assms(5)* **by** *blast*
then have $A \neq C0$
using *assms(1) assms(3) assms(4) bet-neq12--neq cong-diff le-diff* **by** *blast*
then have *P4*: *Bet A B0 C0*
by (*smt Out-def P1 P2 P3 assms(1) assms(2) assms(3) bet--le1213 between-exchange2 between-symmetry l5-1 l5-3 l5-6 l6-13-1 not-cong-3412*)
have *K1*: *B0 C0 Le B C0*
using *P1 P4 between-exchange3 l5-12-a* **by** *blast*
have *K2*: *B C0 Le B C*
using *P1 P3 P4 bet--le1213 between-exchange3 between-exchange4* **by** *blast*
then have *Cong B0 C0 B' C'*
using *P1 P3 P4 assms(3) l4-3-1 not-cong-3412* **by** *blast*
then show *?thesis*
by (*meson K1 K2 cong--nlt le-transitivity nlt--le*)
qed

lemma *bet2-le2--le2356*:
assumes *Bet A B C* **and**
Bet A' B' C' **and**
A B Le A' B' **and**
A' C' Le A C
shows *B' C' Le B C*
proof (*cases*)
assume $A = B$
then show *?thesis*
using *assms(1) assms(2) assms(3) assms(4) bet2-le2--le2356-R1* **by** *blast*
next
assume $\neg A = B$
then show *?thesis*
using *assms(1) assms(2) assms(3) assms(4) bet2-le2--le2356-R2* **by** *blast*
qed

lemma *bet2-le2--le1245*:
assumes *Bet A B C* **and**
Bet A' B' C' **and**
B C Le B' C' **and**
A' C' Le A C
shows *A' B' Le A B*
using *assms(1) assms(2) assms(3) assms(4) bet2-le2--le2356 between-symmetry le-comm* **by** *blast*

lemma *cong-preserves-bet*:
assumes *Bet B A' A0* **and**
Cong B A' E D' **and**
Cong B A0 E D0 **and**
E Out D' D0

shows *Bet E D′ D0*
using *Tarski-neutral-dimensionless.l6-13-1 Tarski-neutral-dimensionless-axioms assms(1) assms(2) assms(3) assms(4) bet--le1213 l5-6* **by** *fastforce*

lemma *out-cong-cong*:
assumes *B Out A A0* **and**
E Out D D0 **and**
Cong B A E D **and**
Cong B A0 E D0
shows *Cong A A0 D D0*
by (*meson Out-def assms(1) assms(2) assms(3) assms(4) cong-4321 cong-symmetry l4-3-1 l5-6 l6-13-1 l6-13-2*)

lemma *not-out-bet*:
assumes *Col A B C* **and**
¬ *B Out A C*
shows *Bet A B C*
using *assms(1) assms(2) l6-4-2* **by** *blast*

lemma *or-bet-out*:
shows *Bet A B C* ∨ *B Out A C* ∨ ¬ *Col A B C*
using *not-out-bet* **by** *blast*

lemma *not-bet-out*:
assumes *Col A B C* **and**
¬ *Bet A B C*
shows *B Out A C*
by (*simp add*: *assms(1) assms(2) l6-4-2*)

lemma *not-bet-and-out*:
shows ¬ (*Bet A B C* ∧ *B Out A C*)
using *bet-col l6-4-1* **by** *blast*

lemma *out-to-bet*:
assumes *Col A′ B′ C′* **and**
B Out A C ⟷ *B′ Out A′ C′* **and**
Bet A B C
shows *Bet A′ B′ C′*
using *assms(1) assms(2) assms(3) not-bet-and-out or-bet-out* **by** *blast*

lemma *col-out2-col*:
assumes *Col A B C* **and**
B Out A AA **and**
B Out C CC
shows *Col AA B CC* **using** *l6-21*
by (*smt Tarski-neutral-dimensionless.out-col Tarski-neutral-dimensionless-axioms assms(1) assms(2) assms(3) col-trivial-2 not-col-permutation-1 out-diff1*)

lemma *bet2-out-out*:
assumes *B* ≠ *A* **and**
B′ ≠ *A* **and**
A Out C C′ **and**
Bet A B C **and**
Bet A B′ C′
shows *A Out B B′*
by (*meson assms(1) assms(2) assms(3) assms(4) assms(5) bet-out l6-6 l6-7*)

lemma *bet2--out*:
assumes *A* ≠ *B* **and**
A ≠ *B′* **and**
Bet A B C
and *Bet A B′ C*
shows *A Out B B′*
using *Out-def assms(1) assms(2) assms(3) assms(4) l5-3* **by** *auto*

lemma *out-bet-out-1*:
assumes *P Out A C* **and**

Bet A B C
shows *P Out A B*
by (*metis assms*(*1*) *assms*(*2*) *not-bet-and-out out2-bet-out out-trivial*)

lemma *out-bet-out-2*:
assumes *P Out A C* **and**
Bet A B C
shows *P Out B C*
using *assms*(*1*) *assms*(*2*) *l6-6 l6-7 out-bet-out-1* **by** *blast*

lemma *out-bet--out*:
assumes *Bet P Q A* **and**
Q Out A B
shows *P Out A B*
by (*smt Out-def assms*(*1*) *assms*(*2*) *bet-out-1 bet-out--bet*)

lemma *segment-reverse*:
assumes *Bet A B C*
shows ∃ *B'*. *Bet A B' C* ∧ *Cong C B' A B*
by (*metis Bet-perm Cong-perm assms bet-cong-eq cong-reflexivity segment-construction-2*)

lemma *diff-col-ex*:
shows ∃ *C*. *A* ≠ *C* ∧ *B* ≠ *C* ∧ *Col A B C*
by (*metis bet-col bet-neq12--neq point-construction-different*)

lemma *diff-bet-ex3*:
assumes *Bet A B C*
shows ∃ *D*. *A* ≠ *D* ∧ *B* ≠ *D* ∧ *C* ≠ *D* ∧ *Col A B D*
by (*metis* (*mono-tags*, *opaque-lifting*) *Col-def bet-out-1 between-trivial2 col-transitivity-1 l6-4-1 point-construction-different*)

lemma *diff-col-ex3*:
assumes *Col A B C*
shows ∃ *D*. *A* ≠ *D* ∧ *B* ≠ *D* ∧ *C* ≠ *D* ∧ *Col A B D*
by (*metis Bet-perm Col-def between-equality between-trivial2 point-construction-different*)

lemma *Out-cases*:
assumes *A Out B C* ∨ *A Out C B*
shows *A Out B C*
using *assms l6-6* **by** *blast*

3.6 Midpoint

lemma *midpoint-dec*:
I Midpoint A B ∨ ¬ *I Midpoint A B*
by *simp*

lemma *is-midpoint-id*:
assumes *A Midpoint A B*
shows *A* = *B*
using *Midpoint-def assms between-cong* **by** *blast*

lemma *is-midpoint-id-2*:
assumes *A Midpoint B A*
shows *A* = *B*
using *Midpoint-def assms cong-diff-2* **by** *blast*

lemma *l7-2*:
assumes *M Midpoint A B*
shows *M Midpoint B A*
using *Bet-perm Cong-perm Midpoint-def assms* **by** *blast*

lemma *l7-3*:
assumes *M Midpoint A A*
shows *M* = *A*
using *Midpoint-def assms bet-neq23--neq* **by** *blast*

lemma *l7-3-2*:
  A Midpoint A A
  by (*simp add*: *Midpoint-def between-trivial2 cong-reflexivity*)

lemma *symmetric-point-construction*:
  ∃ *P′*. *A Midpoint P P′*
  by (*meson Midpoint-def cong--le cong--le3412 le-anti-symmetry segment-construction*)

lemma *symmetric-point-uniqueness*:
  assumes *P Midpoint A P1* **and**
    P Midpoint A P2
  shows *P1* = *P2*
  by (*metis Midpoint-def assms*(*1*) *assms*(*2*) *between-cong-3 cong-diff-4 cong-inner-transitivity*)

lemma *l7-9*:
  assumes *A Midpoint P X* **and**
    A Midpoint Q X
  shows *P* = *Q*
  using *assms*(*1*) *assms*(*2*) *l7-2 symmetric-point-uniqueness* **by** *blast*

lemma *l7-9-bis*:
  assumes *A Midpoint P X* **and**
    A Midpoint X Q
  shows *P* = *Q*
  using *assms*(*1*) *assms*(*2*) *l7-2 symmetric-point-uniqueness* **by** *blast*

lemma *l7-13-R1*:
  assumes *A* ≠ *P* **and**
    A Midpoint P′ P **and**
    A Midpoint Q′ Q
  shows *Cong P Q P′ Q′*
proof −
  obtain *X* **where** *P1*: *Bet P′ P X* ∧ *Cong P X Q A*
    using *segment-construction* **by** *blast*
  obtain *X′* **where** *P2*: *Bet X P′ X′* ∧ *Cong P′ X′ Q A*
    using *segment-construction* **by** *blast*
  obtain *Y* **where** *P3*: *Bet Q′ Q Y* ∧ *Cong Q Y P A*
    using *segment-construction* **by** *blast*
  obtain *Y′* **where** *P4*: *Bet Y Q′ Y′* ∧ *Cong Q′ Y′ P A*
    using *segment-construction* **by** *blast*
  have *P5*: *Bet Y A Q′*
    by (*meson Midpoint-def P3 P4 assms*(*3*) *bet3--bet between-symmetry l5-3*)
  have *P6*: *Bet P′ A X*
    using *Midpoint-def P1 assms*(*2*) *between-exchange4* **by** *blast*
  have *P7*: *Bet A P X*
    using *Midpoint-def P1 assms*(*2*) *between-exchange3* **by** *blast*
  have *P8*: *Bet Y Q A*
    using *Midpoint-def P3 assms*(*3*) *between-exchange3 between-symmetry* **by** *blast*
  have *P9*: *Bet A Q′ Y′*
    using *P4 P5 between-exchange3* **by** *blast*
  have *P10*: *Bet X′ P′ A*
    using *P2 P6 between-exchange3 between-symmetry* **by** *blast*
  have *P11*: *Bet X A X′*
    using *P10 P2 P6 between-symmetry outer-transitivity-between2* **by** *blast*
  have *P12*: *Bet Y A Y′*
    using *P4 P5 between-exchange4* **by** *blast*
  have *P13*: *Cong A X Y A*
    using *P1 P3 P7 P8 l2-11-b not-cong-4321* **by** *blast*
  have *P14*: *Cong A Y′ X′ A*
  proof −
    have *Q1*: *Cong Q′ Y′ P′ A*
      using *Midpoint-def P4 assms*(*2*) *cong-transitivity not-cong-3421* **by** *blast*
    have *Cong A Q′ X′ P′*
      by (*meson Midpoint-def P2 assms*(*3*) *cong-transitivity not-cong-3421*)
    then show *?thesis*
      using *P10 P9 Q1 l2-11-b* **by** *blast*

```
  qed
  have P15: Cong A Y A Y'
  proof -
    have Cong Q Y Q' Y'
      using P3 P4 cong-transitivity not-cong-3412 by blast
    then show ?thesis
      using Bet-perm Cong-perm Midpoint-def P8 P9 assms(3) l2-11-b by blast
  qed
  have P16: Cong X A Y' A
    using Cong-cases P13 P15 cong-transitivity by blast
  have P17: Cong A X' A Y
    using P14 P15 cong-transitivity not-cong-3421 by blast
  have P18: X A X' Y' FSC Y' A Y X
  proof -
    have Q3: Col X A X'
      by (simp add: Col-def P11)
    have Cong X X' Y' Y
      using Bet-cases P11 P12 P16 P17 l2-11-b by blast
    then show ?thesis
      by (simp add: Cong3-def FSC-def P16 P17 Q3 cong-4321 cong-pseudo-reflexivity)
  qed
  then have Y Q A X IFSC Y' Q' A X'
   by (smt IFSC-def Midpoint-def P14 P15 P16 P7 P8 P9 assms(1) assms(3) bet-neq12--neq between-symmetry cong-4321
cong-inner-transitivity cong-right-commutativity l4-16)
  then have X P A Q IFSC X' P' A Q'
    by (meson IFSC-def Midpoint-def P10 P7 assms(2) between-symmetry cong-4312 l4-2)
  then show ?thesis
    using l4-2 by auto
qed

lemma l7-13:
  assumes A Midpoint P' P and
    A Midpoint Q' Q
  shows Cong P Q P' Q'
proof (cases)
  assume A = P
  then show ?thesis
    using Midpoint-def assms(1) assms(2) cong-3421 is-midpoint-id-2 by blast
next
  show ?thesis
   by (metis Tarski-neutral-dimensionless.l7-13-R1 Tarski-neutral-dimensionless-axioms assms(1) assms(2) cong-trivial-identity
is-midpoint-id-2 not-cong-2143)
qed

lemma l7-15:
  assumes A Midpoint P P' and
    A Midpoint Q Q' and
    A Midpoint R R' and
    Bet P Q R
  shows Bet P' Q' R'
proof -
  have P Q R Cong3 P' Q' R'
    using Cong3-def assms(1) assms(2) assms(3) l7-13 l7-2 by blast
  then show ?thesis
    using assms(4) l4-6 by blast
qed

lemma l7-16:
  assumes A Midpoint P P' and
    A Midpoint Q Q' and
    A Midpoint R R' and
    A Midpoint S S' and
    Cong P Q R S
  shows Cong P' Q' R' S'
  by (meson assms(1) assms(2) assms(3) assms(4) assms(5) cong-transitivity l7-13 not-cong-3412)
```

lemma *symmetry-preserves-midpoint*:
assumes *Z Midpoint A D* **and**
Z Midpoint B E **and**
Z Midpoint C F **and**
B Midpoint A C
shows *E Midpoint D F*
by (*meson Midpoint-def assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *l7-15 l7-16*)

lemma *Mid-cases*:
assumes *A Midpoint B C* ∨ *A Midpoint C B*
shows *A Midpoint B C*
using *assms l7-2* **by** *blast*

lemma *Mid-perm*:
assumes *A Midpoint B C*
shows *A Midpoint B C* ∧ *A Midpoint C B*
by (*simp add*: *assms l7-2*)

lemma *l7-17*:
assumes *A Midpoint P P′* **and**
B Midpoint P P′
shows *A* = *B*
proof −
obtain *pp* :: *′p* ⇒ *′p* ⇒ *′p* **where**
f1: ∀ *p pa. p Midpoint pa* (*pp p pa*)
by (*meson symmetric-point-construction*)
then have ∀ *p pa. Bet pa p* (*pp p pa*)
by (*meson Midpoint-def*)
then have *f2*: ∀ *p. Bet p p p*
by (*meson between-inner-transitivity*)
have *f3*: ∀ *p pa. Bet* (*pp pa p*) *pa p*
using *f1 Mid-perm Midpoint-def* **by** *blast*
have *f4*: ∀ *p. pp p p* = *p*
using *f2 f1* **by** (*metis Midpoint-def bet-cong-eq*)
have *f5*: *Bet* (*pp P P′*) *P B*
using *f3* **by** (*meson Midpoint-def assms*(*2*) *between-inner-transitivity*)
have *f6*: *A Midpoint P′ P*
using *Mid-perm assms*(*1*) **by** *blast*
have *f7*: *Bet* (*pp P P′*) *P A*
using *f3 Midpoint-def assms*(*1*) *between-inner-transitivity* **by** *blast*
have *f8*: *Bet P′ A P*
using *f6* **by** (*simp add*: *Midpoint-def*)
have *Cong P′ A A P*
using *f6 Midpoint-def* **by** *blast*
then have *P′* = *P* ⟶ *A* = *B*
using *f8* **by** (*metis* (*no-types*) *Midpoint-def assms*(*2*) *bet-cong-eq between-inner-transitivity l5-2*)
then show *?thesis*
using *f7 f6 f5 f4 f1* **by** (*metis* (*no-types*) *Col-perm Mid-perm assms*(*2*) *bet-col l4-18 l5-2 l7-13*)
qed

lemma *l7-17-bis*:
assumes *A Midpoint P P′* **and**
B Midpoint P′ P
shows *A* = *B*
by (*meson Tarski-neutral-dimensionless.l7-17 Tarski-neutral-dimensionless.l7-2 Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*2*))

lemma *l7-20*:
assumes *Col A M B* **and**
Cong M A M B
shows *A* = *B* ∨ *M Midpoint A B*
by (*metis Bet-cases Col-def Midpoint-def assms*(*1*) *assms*(*2*) *between-cong cong-left-commutativity not-cong-3412*)

lemma *l7-20-bis*:
assumes *A* ≠ *B* **and**
Col A M B **and**

Cong M A M B
shows *M Midpoint A B*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *l7-20* **by** *blast*

lemma *cong-col-mid*:
assumes *A* ≠ *C* **and**
Col A B C **and**
Cong A B B C
shows *B Midpoint A C*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *cong-left-commutativity l7-20* **by** *blast*

lemma *l7-21-R1*:
assumes ¬ *Col A B C* **and**
B ≠ *D* **and**
Cong A B C D **and**
Cong B C D A **and**
Col A P C **and**
Col B P D
shows *P Midpoint A C*
proof −
obtain *X* **where** *P1*: *B D P Cong3 D B X*
using *Col-perm assms*(*6*) *cong-pseudo-reflexivity l4-14* **by** *blast*
have *P2*: *Col D B X*
using *P1 assms*(*6*) *l4-13 not-col-permutation-5* **by** *blast*
have *P3*: *B D P A FSC D B X C*
using *FSC-def P1 assms*(*3*) *assms*(*4*) *assms*(*6*) *not-col-permutation-5 not-cong-2143 not-cong-3412* **by** *blast*
have *P4*: *B D P C FSC D B X A*
by (*simp add*: *FSC-def P1 assms*(*3*) *assms*(*4*) *assms*(*6*) *col-permutation-5 cong-4321*)
have *A P C Cong3 C X A*
using *Cong3-def Cong-perm P3 P4 assms*(*2*) *cong-pseudo-reflexivity l4-16* **by** *blast*
then show *?thesis*
by (*smt Cong3-def NCol-cases P2 assms*(*1*) *assms*(*2*) *assms*(*5*) *assms*(*6*) *colx cong-col-mid l4-13 not-col-distincts not-col-permutation-1 not-cong-1243*)
qed

lemma *l7-21*:
assumes ¬ *Col A B C* **and**
B ≠ *D* **and**
Cong A B C D **and**
Cong B C D A **and**
Col A P C **and**
Col B P D
shows *P Midpoint A C* ∧ *P Midpoint B D*
by (*smt assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *assms*(*6*) *col-transitivity-2 is-midpoint-id-2 l7-21-R1 not-col-distincts not-cong-3412*)

lemma *l7-22-aux-R1*:
assumes *Bet A1 C C* **and**
Bet B1 C B2 **and**
Cong C A1 C B1 **and**
Cong C C C B2 **and**
M1 Midpoint A1 B1 **and**
*M2 Midpoint A2 B2***and**
C A1 Le C C
shows *Bet M1 C M2*
by (*metis assms*(*3*) *assms*(*5*) *assms*(*7*) *cong-diff-3 l7-3 le-diff not-bet-distincts*)

lemma *l7-22-aux-R2*:
assumes *A2* ≠ *C* **and**
Bet A1 C A2 **and**
Bet B1 C B2 **and**
Cong C A1 C B1 **and**
Cong C A2 C B2 **and**
M1 Midpoint A1 B1 **and**
M2 Midpoint A2 B2 **and**
C A1 Le C A2

```
  shows Bet M1 C M2
proof -
  obtain X where P1: C Midpoint A2 X
    using symmetric-point-construction by blast
  obtain X0 where P2: C Midpoint B2 X0
    using symmetric-point-construction by blast
  obtain X1 where P3: C Midpoint M2 X1
    using symmetric-point-construction by blast
  have P4: X1 Midpoint X X0
    using P1 P2 P3 assms(7) symmetry-preserves-midpoint by blast
  have P5: C A1 Le C X
    using Cong-perm Midpoint-def P1 assms(8) cong-reflexivity l5-6 by blast
  have P6: Bet C A1 X
    by (smt Midpoint-def P1 P5 assms(1) assms(2) bet2--out between-symmetry is-midpoint-id-2 l5-2 l6-13-1)
  have P7: C B1 Le C X0
  proof -
    have Q1: Cong C A1 C B1
      by (simp add: assms(4))
    have Cong C X C X0
      using P1 P2 assms(5) l7-16 l7-3-2 by blast
    then show ?thesis
      using P5 Q1 l5-6 by blast
  qed
  have P8: Bet C B1 X0
    by (smt Midpoint-def P2 P7 assms(1) assms(3) assms(5) bet2--out between-symmetry cong-identity l5-2 l6-13-1)
  obtain Q where P9: Bet X1 Q C ∧ Bet A1 Q B1
    by (meson Bet-perm Midpoint-def P4 P6 P8 l3-17)
  have P10: X A1 C X1 IFSC X0 B1 C X1
    by (smt Cong-perm IFSC-def Midpoint-def P1 P2 P4 P6 P8 assms(4) assms(5) between-symmetry cong-reflexivity
l7-16 l7-3-2)
  have P11: Cong A1 X1 B1 X1
    using P10 l4-2 by blast
  have P12: Cong Q A1 Q B1
  proof (cases)
    assume C = X1
    then show ?thesis
      using P9 assms(4) bet-neq12--neq by blast
  next
    assume Q1: ¬ C = X1
    have Q2: Col C X1 Q
      using Col-def P9 by blast
    have Q3: Cong C A1 C B1
      by (simp add: assms(4))
    have Cong X1 A1 X1 B1
      using P11 not-cong-2143 by blast
    then show ?thesis
      using Q1 Q2 Q3 l4-17 by blast
  qed
  have P13: Q Midpoint A1 B1
    by (simp add: Midpoint-def P12 P9 cong-left-commutativity)
  then show ?thesis
    using Midpoint-def P3 P9 assms(6) between-inner-transitivity between-symmetry l7-17 by blast
qed

lemma l7-22-aux:
  assumes Bet A1 C A2 and
    Bet B1 C B2 and
    Cong C A1 C B1 and
    Cong C A2 C B2 and
    M1 Midpoint A1 B1 and
    M2 Midpoint A2 B2 and
    C A1 Le C A2
  shows Bet M1 C M2
  by (smt assms(1) assms(2) assms(3) assms(4) assms(5) assms(6) assms(7) l7-22-aux-R1 l7-22-aux-R2)

lemma l7-22:
```

assumes *Bet A1 C A2* **and**
Bet B1 C B2 **and**
Cong C A1 C B1 **and**
Cong C A2 C B2 **and**
M1 Midpoint A1 B1 **and**
M2 Midpoint A2 B2
shows *Bet M1 C M2*
by (*meson assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *assms*(*6*) *between-symmetry l7-22-aux local.le-cases*)

lemma *bet-col1*:
assumes *Bet A B D* **and**
Bet A C D
shows *Col A B C*
using *Bet-perm Col-def assms*(*1*) *assms*(*2*) *l5-3* **by** *blast*

lemma *l7-25-R1*:
assumes *Cong C A C B* **and**
Col A B C
shows ∃ *X*. *X Midpoint A B*
using *assms*(*1*) *assms*(*2*) *l7-20 l7-3-2 not-col-permutation-5* **by** *blast*

lemma *l7-25-R2*:
assumes *Cong C A C B* **and**
¬ *Col A B C*
shows ∃ *X*. *X Midpoint A B*
proof −
obtain *P* **where** *P1*: *Bet C A P* ∧ *A* ≠ *P*
using *point-construction-different* **by** *auto*
obtain *Q* **where** *P2*: *Bet C B Q* ∧ *Cong B Q A P*
using *segment-construction* **by** *blast*
obtain *R* **where** *P3*: *Bet A R Q* ∧ *Bet B R P*
using *P1 P2 between-symmetry inner-pasch* **by** *blast*
obtain *X* **where** *P4*: *Bet A X B* ∧ *Bet R X C*
using *P1 P3 inner-pasch* **by** *blast*
have *Cong X A X B*
proof −
have *Q1*: *Cong R A R B* ⟶ *Cong X A X B*
proof (*cases*)
assume *R* = *C*
then show *?thesis*
using *P4 bet-neq12--neq* **by** *blast*
next
assume *Q2*: ¬ *R* = *C*
have *Col R C X*
using *Col-perm P4 bet-col* **by** *blast*
then show *?thesis*
using *Q2 assms*(*1*) *l4-17* **by** *blast*
qed
have *Cong R A R B*
proof −
have *Q3*: *C A P B OFSC C B Q A*
by (*simp add*: *OFSC-def P1 P2 assms*(*1*) *cong-pseudo-reflexivity cong-symmetry*)
have *Q4*: *Cong P B Q A*
using *Q3 assms*(*2*) *five-segment-with-def not-col-distincts* **by** *blast*
obtain *R′* **where** *Q5*: *Bet A R′ Q* ∧ *B R P Cong3 A R′ Q*
using *Cong-perm P3 Q4 l4-5* **by** *blast*
have *Q6*: *B R P A IFSC A R′ Q B*
by (*meson Cong3-def IFSC-def OFSC-def P3 Q3 Q5 not-cong-2143*)
have *Q7*: *B R P Q IFSC A R′ Q P*
using *IFSC-def P2 Q6 cong-pseudo-reflexivity* **by** *auto*
have *Q8*: *Cong R A R′ B*
using *Q6 l4-2* **by** *auto*
have *Q9*: *Cong R Q R′ P*
using *Q7 l4-2* **by** *auto*
have *Q10*: *A R Q Cong3 B R′ P*
using *Cong3-def Q4 Q8 Q9 cong-commutativity not-cong-4321* **by** *blast*

```
      have Q11: Col B R′ P
        using P3 Q10 bet-col l4-13 by blast
      have R = R′
      proof −
        have R1: B ≠ P
          using P1 assms(1) between-cong by blast
        then have R2: A ≠ Q
          using Q4 cong-diff-2 by blast
        have R3: B ≠ Q
          using P1 P2 cong-diff-3 by blast
        then have R4: B ≠ R
          by (metis Cong3-def P1 Q11 Q5 assms(2) bet-col cong-diff-3 l6-21 not-col-distincts)
        have R5: ¬ Col A Q B
          by (metis P2 R3 assms(2) bet-col col-permutation-3 col-trivial-2 l6-21)
        have R6: B ≠ P
          by (simp add: R1)
        have R7: Col A Q R
          using NCol-cases P3 bet-col by blast
        have R8: Col A Q R′
          using Q5 bet-col col-permutation-5 by blast
        have R9: Col B P R
          using NCol-cases P3 bet-col by blast
        have  Col B P R′
          using Col-perm Q11 by blast
        then show ?thesis
          using R5 R6 R7 R8 R9 l6-21 by blast
      qed
      then show ?thesis
        using Q8 by blast
    qed
    then show ?thesis
      using Q1 by blast
  qed
  then show ?thesis
    using P4 assms(2) bet-col l7-20-bis not-col-distincts by blast
qed

lemma l7-25:
  assumes Cong C A C B
  shows ∃ X. X Midpoint A B
  using assms l7-25-R1 l7-25-R2 by blast

lemma midpoint-distinct-1:
  assumes A ≠ B and
    I Midpoint A B
  shows I ≠ A ∧ I ≠ B
  using assms(1) assms(2) is-midpoint-id is-midpoint-id-2 by blast

lemma midpoint-distinct-2:
  assumes I ≠ A and
    I Midpoint A B
  shows A ≠ B ∧ I ≠ B
  using assms(1) assms(2) is-midpoint-id-2 l7-3 by blast

lemma midpoint-distinct-3:
  assumes I ≠ B and
    I Midpoint A B
  shows A ≠ B ∧ I ≠ A
  using assms(1) assms(2) is-midpoint-id l7-3 by blast

lemma midpoint-def:
  assumes Bet A B C and
    Cong A B B C
  shows B Midpoint A C
  using Midpoint-def assms(1) assms(2) by blast
```

lemma *midpoint-bet*:
  assumes *B Midpoint A C*
  shows *Bet A B C*
  using *Midpoint-def assms* **by** *blast*

lemma *midpoint-col*:
  assumes *M Midpoint A B*
  shows *Col M A B*
  using *assms bet-col col-permutation-4 midpoint-bet* **by** *blast*

lemma *midpoint-cong*:
  assumes *B Midpoint A C*
  shows *Cong A B B C*
  using *Midpoint-def assms* **by** *blast*

lemma *midpoint-out*:
  assumes $A \neq C$ **and**
    B Midpoint A C
  shows *A Out B C*
  using *assms(1) assms(2) bet-out midpoint-bet midpoint-distinct-1* **by** *blast*

lemma *midpoint-out-1*:
  assumes $A \neq C$ **and**
    B Midpoint A C
  shows *C Out A B*
  by (*metis Tarski-neutral-dimensionless.midpoint-bet Tarski-neutral-dimensionless.midpoint-distinct-1 Tarski-neutral-dimensionless-ax*
assms(1) assms(2) bet-out-1 l6-6)

lemma *midpoint-not-midpoint*:
  assumes $A \neq B$ **and**
    I Midpoint A B
  shows ¬ *B Midpoint A I*
  using *assms(1) assms(2) between-equality-2 midpoint-bet midpoint-distinct-1* **by** *blast*

lemma *swap-diff*:
  assumes $A \neq B$
  shows $B \neq A$
  using *assms* **by** *auto*

lemma *cong-cong-half-1*:
  assumes *M Midpoint A B* **and**
    M′ Midpoint A′ B′ **and**
    Cong A B A′ B′
  shows *Cong A M A′ M′*
proof −
  obtain *M′′* **where** *P1*: *Bet A′ M′′ B′* ∧ *A M B Cong3 A′ M′′ B′*
    using *assms(1) assms(3) l4-5 midpoint-bet* **by** *blast*
  have *P2*: *M′′ Midpoint A′ B′*
    by (*meson Cong3-def P1 assms(1) cong-inner-transitivity midpoint-cong midpoint-def*)
  have *P3*: *M′ = M′′*
    using *P2 assms(2) l7-17* **by** *auto*
  then show *?thesis*
    using *Cong3-def P1* **by** *blast*
qed

lemma *cong-cong-half-2*:
  assumes *M Midpoint A B* **and**
    M′ Midpoint A′ B′ **and**
    Cong A B A′ B′
  shows *Cong B M B′ M′*
  using *assms(1) assms(2) assms(3) cong-cong-half-1 l7-2 not-cong-2143* **by** *blast*

lemma *cong-mid2--cong*:
  assumes *M Midpoint A B* **and**
    M′ Midpoint A′ B′ **and**
    Cong A M A′ M′

shows *Cong A B A' B'*
by (*meson assms*(*1*) *assms*(*2*) *assms*(*3*) *cong-inner-transitivity l2-11-b midpoint-bet midpoint-cong*)

lemma *mid--lt*:
assumes $A \neq B$ **and**
M Midpoint A B
shows *A M Lt A B*
using *assms*(*1*) *assms*(*2*) *bet--lt1213 midpoint-bet midpoint-distinct-1* **by** *blast*

lemma *le-mid2--le13*:
assumes *M Midpoint A B* **and**
M' Midpoint A' B' **and**
A M Le A' M'
shows *A B Le A' B'*
by (*smt Tarski-neutral-dimensionless.cong-mid2--cong Tarski-neutral-dimensionless.l7-13 Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*2*) *assms*(*3*) *bet2-le2--le2356 l5-6 l7-3-2 le-anti-symmetry le-comm local.le-cases midpoint-bet*)

lemma *le-mid2--le12*:
assumes *M Midpoint A B* **and**
M' Midpoint A' B'
and *A B Le A' B'*
shows *A M Le A' M'*
by (*meson assms*(*1*) *assms*(*2*) *assms*(*3*) *cong--le3412 cong-cong-half-1 le-anti-symmetry le-mid2--le13 local.le-cases*)

lemma *lt-mid2--lt13*:
assumes *M Midpoint A B* **and**
M' Midpoint A' B' **and**
A M Lt A' M'
shows *A B Lt A' B'*
by (*meson Tarski-neutral-dimensionless.le-mid2--le12 Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*2*) *assms*(*3*) *lt--nle nlt--le*)

lemma *lt-mid2--lt12*:
assumes *M Midpoint A B* **and**
M' Midpoint A' B' **and**
A B Lt A' B'
shows *A M Lt A' M'*
by (*meson Tarski-neutral-dimensionless.le-mid2--le13 Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*2*) *assms*(*3*) *le--nlt nle--lt*)

lemma *midpoint-preserves-out*:
assumes *A Out B C* **and**
M Midpoint A A' **and**
M Midpoint B B' **and**
M Midpoint C C'
shows *A' Out B' C'*
by (*smt Out-def assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *l6-4-2 l7-15 l7-2 not-bet-and-out not-col-distincts*)

lemma *col-cong-bet*:
assumes *Col A B D* **and**
Cong A B C D **and**
Bet A C B
shows *Bet C A D* $\vee$ *Bet C B D*
by (*smt Col-def assms*(*1*) *assms*(*2*) *assms*(*3*) *bet-cong-eq between-inner-transitivity col-transitivity-2 cong-4321 l6-2 not-bet-and-out not-cong-4312 third-point*)

lemma *col-cong2-bet1*:
assumes *Col A B D* **and**
Bet A C B **and**
Cong A B C D **and**
Cong A C B D
shows *Bet C B D*
by (*metis assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *bet--le1213 bet-cong-eq between-symmetry col-cong-bet cong--le cong-left-commutativi l5-12-b l5-6 outer-transitivity-between2*)

lemma *col-cong2-bet2*:

assumes *Col A B D* **and**
Bet A C B **and**
Cong A B C D **and**
Cong A D B C
shows *Bet C A D*
by (*metis assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *bet-cong-eq col-cong-bet cong-identity not-bet-distincts not-cong-3421 outer-transitivity-between2*)

lemma *col-cong2-bet3*:
assumes *Col A B D* **and**
Bet A B C **and**
Cong A B C D **and**
Cong A C B D
shows *Bet B C D*
by (*metis assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *bet--le1213 bet--le2313 bet-col col-transitivity-2 cong-diff-3 cong-reflexivity l5-12-b l5-6 not-bet-distincts*)

lemma *col-cong2-bet4*:
assumes *Col A B C* **and**
Bet A B D **and**
Cong A B C D **and**
Cong A D B C
shows *Bet B D C*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *col-cong2-bet3 cong-right-commutativity* **by** *blast*

lemma *col-bet2-cong1*:
assumes *Col A B D* **and**
Bet A C B **and**
Cong A B C D **and**
Bet C B D
shows *Cong A C D B*
by (*meson assms*(*2*) *assms*(*3*) *assms*(*4*) *between-symmetry cong-pseudo-reflexivity cong-right-commutativity l4-3*)

lemma *col-bet2-cong2*:
assumes *Col A B D* **and**
Bet A C B **and**
Cong A B C D **and**
Bet C A D
shows *Cong D A B C*
by (*meson assms*(*2*) *assms*(*3*) *assms*(*4*) *between-symmetry cong-commutativity cong-pseudo-reflexivity cong-symmetry l4-3*)

lemma *bet2-lt2--lt*:
assumes *Bet a Po b* **and**
Bet A PO B **and**
Po a Lt PO A **and**
Po b Lt PO B
shows *a b Lt A B*
by (*metis Lt-cases Tarski-neutral-dimensionless.nle--lt Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *bet2-le2--le1245 le--nlt lt--le*)

lemma *bet2-lt-le--lt*:
assumes *Bet a Po b* **and**
Bet A PO B **and**
Cong Po a PO A **and**
Po b Lt PO B
shows *a b Lt A B*
by (*smt Lt-def Tarski-neutral-dimensionless.l4-3-1 Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *bet2-le2--le cong--le not-cong-2143*)

3.7 Orthogonality

lemma *per-dec*:
Per A B C ∨ ¬ *Per A B C*
by *simp*

lemma *l8-2*:
  assumes *Per A B C*
  shows *Per C B A*
proof −
  obtain C' **where** *P1*: *B Midpoint C* C' ∧ *Cong A C A* C'
    using *Per-def assms* **by** *blast*
  obtain A' **where** *P2*: *B Midpoint A* A'
    using *symmetric-point-construction* **by** *blast*
  have *Cong* C' *A C* A'
    using *Mid-perm P1 P2 l7-13* **by** *blast*
  thus *?thesis*
    using *P1 P2 Per-def cong-4321 cong-inner-transitivity* **by** *blast*
qed

lemma *Per-cases*:
  assumes *Per A B C* ∨ *Per C B A*
  shows *Per A B C*
  using *assms l8-2* **by** *blast*

lemma *Per-perm* :
  assumes *Per A B C*
  shows *Per A B C* ∧ *Per C B A*
  by (*simp add*: *assms l8-2*)

lemma *l8-3* :
  assumes *Per A B C* **and**
    $A \neq B$ **and**
    Col B A A'
  shows *Per* A' *B C*
  by (*smt Per-def assms*(*1*) *assms*(*2*) *assms*(*3*) *l4-17 l7-13 l7-2 l7-3-2*)

lemma *l8-4*:
  assumes *Per A B C* **and**
    B Midpoint C C'
  shows *Per A B* C'
  by (*metis Tarski-neutral-dimensionless.l8-2 Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*2*) *l8-3 midpoint-col midpoint-distinct-1*)

lemma *l8-5*:
  Per A B B
  using *Per-def cong-reflexivity l7-3-2* **by** *blast*

lemma *l8-6*:
  assumes *Per A B C* **and**
    Per A' *B C* **and**
    Bet A C A'
  shows $B = C$
  by (*metis Per-def assms*(*1*) *assms*(*2*) *assms*(*3*) *l4-19 midpoint-distinct-3 symmetric-point-uniqueness*)

lemma *l8-7*:
  assumes *Per A B C* **and**
    Per A C B
  shows $B = C$
proof −
  obtain C' **where** *P1*: *B Midpoint C* C' ∧ *Cong A C A* C'
    using *Per-def assms*(*1*) **by** *blast*
  obtain A' **where** *P2*: *C Midpoint A* A'
    using *Per-def assms*(*2*) *l8-2* **by** *blast*
  have *Per* C' *C A*
   by (*metis P1 Tarski-neutral-dimensionless.l8-3 Tarski-neutral-dimensionless-axioms assms*(*2*) *bet-col l8-2 midpoint-bet midpoint-distinct-3*)
  then have *Cong A* C' A' C'
    using *Cong-perm P2 Per-def symmetric-point-uniqueness* **by** *blast*
  then have *Cong* A' *C* A' C'
    using *P1 P2 cong-inner-transitivity midpoint-cong not-cong-2134* **by** *blast*
  then have *Q4*: *Per* A' *B C*

using *P1 Per-def* **by** *blast*
have *Bet A' C A*
using *Mid-perm P2 midpoint-bet* **by** *blast*
thus *?thesis*
using *Q4 assms(1) l8-6* **by** *blast*
qed

lemma *l8-8*:
assumes *Per A B A*
shows $A = B$
using *Tarski-neutral-dimensionless.l8-6 Tarski-neutral-dimensionless-axioms assms between-trivial2* **by** *fastforce*

lemma *per-distinct*:
assumes *Per A B C* **and**
$A \neq B$
shows $A \neq C$
using *assms(1) assms(2) l8-8* **by** *blast*

lemma *per-distinct-1*:
assumes *Per A B C* **and**
$B \neq C$
shows $A \neq C$
using *assms(1) assms(2) l8-8* **by** *blast*

lemma *l8-9*:
assumes *Per A B C* **and**
Col A B C
shows $A = B \lor C = B$
using *Col-cases assms(1) assms(2) l8-3 l8-8* **by** *blast*

lemma *l8-10*:
assumes *Per A B C* **and**
A B C Cong3 A' B' C'
shows *Per A' B' C'*
proof −
obtain *D* **where** *P1*: *B Midpoint C D* $\land$ *Cong A C A D*
using *Per-def assms(1)* **by** *blast*
obtain *D'* **where** *P2*: *Bet C' B' D'* $\land$ *Cong B' D' B' C'*
using *segment-construction* **by** *blast*
have *P3*: *B' Midpoint C' D'*
by (*simp add*: *Midpoint-def P2 cong-4312*)
have *Cong A' C' A' D'*
proof (*cases*)
assume $C = B$
thus *?thesis*
by (*metis Cong3-def P3 assms(2) cong-diff-4 cong-reflexivity is-midpoint-id*)
next
assume *Q1*: $\neg$ $C = B$
have *C B D A OFSC C' B' D' A'*
by (*metis Cong3-def OFSC-def P1 P3 Tarski-neutral-dimensionless.cong-mid2--cong Tarski-neutral-dimensionless-axioms assms(2) cong-commutativity l4-3-1 midpoint-bet*)
thus *?thesis*
by (*meson OFSC-def P1 Q1 cong-4321 cong-inner-transitivity five-segment-with-def*)
qed
thus *?thesis*
using *Per-def P3* **by** *blast*
qed

lemma *col-col-per-per*:
assumes $A \neq X$ **and**
$C \neq X$ **and**
Col U A X **and**
Col V C X **and**
Per A X C
shows *Per U X V*
by (*meson Tarski-neutral-dimensionless.l8-2 Tarski-neutral-dimensionless.l8-3 Tarski-neutral-dimensionless-axioms*

assms(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *not-col-permutation-3*)

lemma *perp-in-dec*:
X PerpAt A B C D ∨ ¬ *X PerpAt A B C D*
by *simp*

lemma *perp-distinct*:
assumes *A B Perp C D*
shows *A* ≠ *B* ∧ *C* ≠ *D*
using *PerpAt-def Perp-def assms* **by** *auto*

lemma *l8-12*:
assumes *X PerpAt A B C D*
shows *X PerpAt C D A B*
using *Per-perm PerpAt-def assms* **by** *auto*

lemma *per-col*:
assumes *B* ≠ *C* **and**
Per A B C **and**
Col B C D
shows *Per A B D*
by (*metis Tarski-neutral-dimensionless.l8-3 Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*2*) *assms*(*3*) *l8-2*)

lemma *l8-13-2*:
assumes *A* ≠ *B* **and**
C ≠ *D* **and**
Col X A B **and**
Col X C D **and**
∃ *U*. ∃ *V*. *Col U A B* ∧ *Col V C D* ∧ *U* ≠ *X* ∧ *V* ≠ *X* ∧ *Per U X V*
shows *X PerpAt A B C D*
proof −
obtain *pp* :: *'p* **and** *ppa* :: *'p* **where**
f1: *Col pp A B* ∧ *Col ppa C D* ∧ *pp* ≠ *X* ∧ *ppa* ≠ *X* ∧ *Per pp X ppa*
using *assms*(*5*) **by** *blast*
obtain *ppb* :: *'p* ⇒ *'p* ⇒ *'p* ⇒ *'p* ⇒ *'p* ⇒ *'p* **and** *ppc* :: *'p* ⇒ *'p* ⇒ *'p* ⇒ *'p* ⇒ *'p* ⇒ *'p* **where**
∀ *x0 x1 x2 x3 x4*. (∃ *v5 v6*. (*Col v5 x3 x2* ∧ *Col v6 x1 x0*) ∧ ¬ *Per v5 x4 v6*) = ((*Col* (*ppb x0 x1 x2 x3 x4*) *x3 x2* ∧ *Col* (*ppc x0 x1 x2 x3 x4*) *x1 x0*) ∧ ¬ *Per* (*ppb x0 x1 x2 x3 x4*) *x4* (*ppc x0 x1 x2 x3 x4*))
by *moura*
then have *f2*: ∀ *p pa pb pc pd*. (¬ *p PerpAt pa pb pc pd* ∨ *pa* ≠ *pb* ∧ *pc* ≠ *pd* ∧ *Col p pa pb* ∧ *Col p pc pd* ∧ (∀ *pe pf*. (¬ *Col pe pa pb* ∨ ¬ *Col pf pc pd*) ∨ *Per pe p pf*)) ∧ (*p PerpAt pa pb pc pd* ∨ *pa* = *pb* ∨ *pc* = *pd* ∨ ¬ *Col p pa pb* ∨ ¬ *Col p pc pd* ∨ (*Col* (*ppb pd pc pb pa p*) *pa pb* ∧ *Col* (*ppc pd pc pb pa p*) *pc pd*) ∧ ¬ *Per* (*ppb pd pc pb pa p*) *p* (*ppc pd pc pb pa p*))
using *PerpAt-def* **by** *fastforce*
{ assume ¬ *Col* (*ppb D C B A X*) *pp X*
then have ¬ *Col* (*ppb D C B A X*) *A B* ∨ ¬ *Col* (*ppc D C B A X*) *C D* ∨ *Per* (*ppb D C B A X*) *X* (*ppc D C B A X*)
using *f1* **by** (*meson assms*(*1*) *assms*(*3*) *col3 not-col-permutation-2*) **}**
moreover
{ assume ¬ *Col* (*ppc D C B A X*) *ppa X*
then have ¬ *Col* (*ppb D C B A X*) *A B* ∨ ¬ *Col* (*ppc D C B A X*) *C D* ∨ *Per* (*ppb D C B A X*) *X* (*ppc D C B A X*)
using *f1* **by** (*meson assms*(*2*) *assms*(*4*) *col3 not-col-permutation-2*) **}**
ultimately have ¬ *Col* (*ppb D C B A X*) *A B* ∨ ¬ *Col* (*ppc D C B A X*) *C D* ∨ *Per* (*ppb D C B A X*) *X* (*ppc D C B A X*)
using *f1* **by** (*meson Tarski-neutral-dimensionless.col-col-per-per Tarski-neutral-dimensionless-axioms*)
then have (*X PerpAt A B C D* ∨ *A* = *B* ∨ *C* = *D* ∨ ¬ *Col X A B* ∨ ¬ *Col X C D* ∨ *Col* (*ppb D C B A X*) *A B* ∧ *Col* (*ppc D C B A X*) *C D* ∧ ¬ *Per* (*ppb D C B A X*) *X* (*ppc D C B A X*)) ∧ (¬ *Col* (*ppb D C B A X*) *A B* ∨ ¬ *Col* (*ppc D C B A X*) *C D* ∨ *Per* (*ppb D C B A X*) *X* (*ppc D C B A X*))
using *f2* **by** *presburger*
thus *?thesis*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) **by** *blast*
qed

lemma *l8-14-1*:
¬ *A B Perp A B*
by (*metis PerpAt-def Perp-def Tarski-neutral-dimensionless.col-trivial-1 Tarski-neutral-dimensionless.col-trivial-3 Tarski-neutral-dime

l8-8)

lemma *l8-14-2-1a*:
  assumes *X PerpAt A B C D*
  shows *A B Perp C D*
  using *Perp-def assms* **by** *blast*

lemma *perp-in-distinct*:
  assumes *X PerpAt A B C D*
  shows $A \neq B \wedge C \neq D$
  using *PerpAt-def assms* **by** *blast*

lemma *l8-14-2-1b*:
  assumes *X PerpAt A B C D* **and**
    Col Y A B **and**
    Col Y C D
  shows $X = Y$
  by (*metis PerpAt-def assms*(*1*) *assms*(*2*) *assms*(*3*) *l8-13-2 l8-14-1 l8-14-2-1a*)

lemma *l8-14-2-1b-bis*:
  assumes *A B Perp C D* **and**
    Col X A B **and**
    Col X C D
  shows *X PerpAt A B C D*
  using *Perp-def assms*(*1*) *assms*(*2*) *assms*(*3*) *l8-14-2-1b* **by** *blast*

lemma *l8-14-2-2*:
  assumes *A B Perp C D* **and**
    $\forall\ Y.\ (Col\ Y\ A\ B \wedge Col\ Y\ C\ D) \longrightarrow X = Y$
  shows *X PerpAt A B C D*
 by (*metis Tarski-neutral-dimensionless.PerpAt-def Tarski-neutral-dimensionless.Perp-def Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*2*))

lemma *l8-14-3*:
  assumes *X PerpAt A B C D* **and**
    Y PerpAt A B C D
  shows $X = Y$
  by (*meson PerpAt-def assms*(*1*) *assms*(*2*) *l8-14-2-1b*)

lemma *l8-15-1*:
  assumes *Col A B X* **and**
    A B Perp C X
  shows *X PerpAt A B C X*
  using *NCol-perm assms*(*1*) *assms*(*2*) *col-trivial-3 l8-14-2-1b-bis* **by** *blast*

lemma *l8-15-2*:
  assumes *Col A B X* **and**
    X PerpAt A B C X
  shows *A B Perp C X*
  using *assms*(*2*) *l8-14-2-1a* **by** *blast*

lemma *perp-in-per*:
  assumes *B PerpAt A B B C*
  shows *Per A B C*
  by (*meson NCol-cases PerpAt-def assms col-trivial-3*)

lemma *perp-sym*:
  assumes *A B Perp A B*
  shows *C D Perp C D*
  using *assms l8-14-1* **by** *auto*

lemma *perp-col0*:
  assumes *A B Perp C D* **and**
    $X \neq Y$ **and**
    Col A B X **and**
    Col A B Y

shows *C D Perp X Y*
proof −
obtain *X0* **where** *P1*: *X0 PerpAt A B C D*
using *Perp-def assms*(*1*) **by** *blast*
then have *P2*: $A \neq B \wedge C \neq D \wedge$ *Col X0 A B* $\wedge$ *Col X0 C D* $\wedge$
((*Col U A B* $\wedge$ *Col V C D*) $\longrightarrow$ *Per U X0 V*) **using** *PerpAt-def* **by** *blast*
have *Q1*: $C \neq D$ **using** *P2* **by** *blast*
have *Q2*: $X \neq Y$ **using** *assms*(*2*) **by** *blast*
have *Q3*: *Col X0 C D* **using** *P2* **by** *blast*
have *Q4*: *Col X0 X Y*
proof −
have $\exists$ *p pa*. *Col p pa Y* $\wedge$ *Col p pa X* $\wedge$ *Col p pa X0* $\wedge$ $p \neq pa$
by (*metis* (*no-types*) *Col-cases P2 assms*(*3*) *assms*(*4*))
thus *?thesis*
using *col3* **by** *blast*
qed
have *X0 PerpAt C D X Y*
proof −
have $\forall$ *U V*. (*Col U C D* $\wedge$ *Col V X Y*) $\longrightarrow$ *Per U X0 V*
by (*metis Col-perm P1 Per-perm Q2 Tarski-neutral-dimensionless.PerpAt-def Tarski-neutral-dimensionless-axioms assms*(*3*) *assms*(*4*) *colx*)
thus *?thesis* **using** *Q1 Q2 Q3 Q4 PerpAt-def* **by** *blast*
qed
thus *?thesis*
using *Perp-def* **by** *auto*
qed

lemma *per-perp-in*:
assumes $A \neq B$ **and**
$B \neq C$ **and**
Per A B C
shows *B PerpAt A B B C*
by (*metis Col-def assms*(*1*) *assms*(*2*) *assms*(*3*) *between-trivial2 l8-13-2*)

lemma *per-perp*:
assumes $A \neq B$ **and**
$B \neq C$ **and**
Per A B C
shows *A B Perp B C*
using *Perp-def assms*(*1*) *assms*(*2*) *assms*(*3*) *per-perp-in* **by** *blast*

lemma *perp-left-comm*:
assumes *A B Perp C D*
shows *B A Perp C D*
proof −
obtain *X* **where** *X PerpAt A B C D*
using *Perp-def assms* **by** *blast*
then have *X PerpAt B A C D*
using *PerpAt-def col-permutation-5* **by** *auto*
thus *?thesis*
using *Perp-def* **by** *blast*
qed

lemma *perp-right-comm*:
assumes *A B Perp C D*
shows *A B Perp D C*
by (*meson Perp-def assms l8-12 perp-left-comm*)

lemma *perp-comm*:
assumes *A B Perp C D*
shows *B A Perp D C*
by (*simp add*: *assms perp-left-comm perp-right-comm*)

lemma *perp-in-sym*:
assumes *X PerpAt A B C D*
shows *X PerpAt C D A B*

by (*simp add*: *assms l8-12*)

lemma *perp-in-left-comm*:
assumes *X PerpAt A B C D*
shows *X PerpAt B A C D*
by (*metis Col-cases PerpAt-def assms*)

lemma *perp-in-right-comm*:
assumes *X PerpAt A B C D*
shows *X PerpAt A B D C*
using *assms perp-in-left-comm perp-in-sym* **by** *blast*

lemma *perp-in-comm*:
assumes *X PerpAt A B C D*
shows *X PerpAt B A D C*
by (*simp add*: *assms perp-in-left-comm perp-in-right-comm*)

lemma *Perp-cases*:
assumes *A B Perp C D* ∨ *B A Perp C D* ∨ *A B Perp D C* ∨ *B A Perp D C* ∨ *C D Perp A B* ∨ *C D Perp B A* ∨ *D C Perp A B* ∨ *D C Perp B A*
shows *A B Perp C D*
by (*meson Perp-def assms perp-in-sym perp-left-comm*)

lemma *Perp-perm* :
assumes *A B Perp C D*
shows *A B Perp C D* ∧ *B A Perp C D* ∧ *A B Perp D C* ∧ *B A Perp D C* ∧ *C D Perp A B* ∧ *C D Perp B A* ∧ *D C Perp A B* ∧ *D C Perp B A*
by (*meson Perp-def assms perp-in-sym perp-left-comm*)

lemma *Perp-in-cases*:
assumes *X PerpAt A B C D* ∨ *X PerpAt B A C D* ∨ *X PerpAt A B D C* ∨ *X PerpAt B A D C* ∨ *X PerpAt C D A B* ∨ *X PerpAt C D B A* ∨ *X PerpAt D C A B* ∨ *X PerpAt D C B A*
shows *X PerpAt A B C D*
using *assms perp-in-left-comm perp-in-sym* **by** *blast*

lemma *Perp-in-perm*:
assumes *X PerpAt A B C D*
shows *X PerpAt A B C D* ∧ *X PerpAt B A C D* ∧ *X PerpAt A B D C* ∧ *X PerpAt B A D C* ∧ *X PerpAt C D A B* ∧ *X PerpAt C D B A* ∧ *X PerpAt D C A B* ∧ *X PerpAt D C B A*
using *Perp-in-cases assms* **by** *blast*

lemma *perp-in-col*:
assumes *X PerpAt A B C D*
shows *Col A B X* ∧ *Col C D X*
using *PerpAt-def assms col-permutation-2* **by** *presburger*

lemma *perp-perp-in*:
assumes *A B Perp C A*
shows *A PerpAt A B C A*
using *assms l8-15-1 not-col-distincts* **by** *blast*

lemma *perp-per-1*:
assumes *A B Perp C A*
shows *Per B A C*
using *Perp-in-cases assms perp-in-per perp-perp-in* **by** *blast*

lemma *perp-per-2*:
assumes *A B Perp A C*
shows *Per B A C*
by (*simp add*: *Perp-perm assms perp-per-1*)

lemma *perp-col*:
assumes *A* ≠ *E* **and**
A B Perp C D **and**
Col A B E
shows *A E Perp C D*

using *Perp-perm assms*(*1*) *assms*(*2*) *assms*(*3*) *col-trivial-3 perp-col0* **by** *blast*

lemma *perp-col2*:
assumes *A B Perp X Y* **and**
C ≠ *D* **and**
Col A B C **and**
Col A B D
shows *C D Perp X Y*
using *Perp-perm assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *perp-col0* **by** *blast*

lemma *perp-col4*:
assumes *P* ≠ *Q* **and**
R ≠ *S* **and**
Col A B P **and**
Col A B Q **and**
Col C D R **and**
Col C D S **and**
A B Perp C D
shows *P Q Perp R S*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *assms*(*6*) *assms*(*7*) *perp-col0* **by** *blast*

lemma *perp-not-eq-1*:
assumes *A B Perp C D*
shows *A* ≠ *B*
using *assms perp-distinct* **by** *auto*

lemma *perp-not-eq-2*:
assumes *A B Perp C D*
shows *C* ≠ *D*
using *assms perp-distinct* **by** *auto*

lemma *diff-per-diff*:
assumes *A* ≠ *B* **and**
Cong A P B R **and**
Per B A P
and *Per A B R*
shows *P* ≠ *R*
using *assms*(*1*) *assms*(*3*) *assms*(*4*) *l8-2 l8-7* **by** *blast*

lemma *per-not-colp*:
assumes *A* ≠ *B* **and**
A ≠ *P* **and**
B ≠ *R* **and**
Per B A P
and *Per A B R*
shows ¬ *Col P A R*
by (*metis Per-cases Tarski-neutral-dimensionless.col-permutation-4 Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*2*) *assms*(*4*) *assms*(*5*) *l8-3 l8-7*)

lemma *per-not-col*:
assumes *A* ≠ *B* **and**
B ≠ *C* **and**
Per A B C
shows ¬ *Col A B C*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *l8-9* **by** *auto*

lemma *perp-not-col2*:
assumes *A B Perp C D*
shows ¬ *Col A B C* ∨ ¬ *Col A B D*
using *assms l8-14-1 perp-col2 perp-distinct* **by** *blast*

lemma *perp-not-col*:
assumes *A B Perp P A*
shows ¬ *Col A B P*
proof −
have *A PerpAt A B P A*

using *assms perp-perp-in* **by** *auto*
then have *Per P A B*
by (*simp add*: *perp-in-per perp-in-sym*)
then have ¬ *Col B A P*
by (*metis NCol-perm Tarski-neutral-dimensionless.perp-not-eq-1 Tarski-neutral-dimensionless.perp-not-eq-2 Tarski-neutral-dimensio*
assms per-not-col)
thus *?thesis*
using *Col-perm* **by** *blast*
qed

lemma *perp-in-col-perp-in*:
assumes $C \neq E$ **and**
Col C D E **and**
P PerpAt A B C D
shows *P PerpAt A B C E*
proof −
have *P2*: $C \neq D$
using *assms*(*3*) *perp-in-distinct* **by** *blast*
have *P3*: *Col P A B*
using *PerpAt-def assms*(*3*) **by** *auto*
have *Col P C D*
using *PerpAt-def assms*(*3*) **by** *blast*
then have *Col P C E*
using *P2 assms*(*2*) *col-trivial-2 colx* **by** *blast*
thus *?thesis*
by (*smt P3 Perp-perm Tarski-neutral-dimensionless.l8-14-2-1b-bis Tarski-neutral-dimensionless.perp-col Tarski-neutral-dimensionles*
assms(*1*) *assms*(*2*) *assms*(*3*) *l8-14-2-1a*)
qed

lemma *perp-col2-bis*:
assumes *A B Perp C D* **and**
Col C D P **and**
Col C D Q **and**
$P \neq Q$
shows *A B Perp P Q*
using *Perp-cases assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *perp-col0* **by** *blast*

lemma *perp-in-perp-bis-R1*:
assumes $X \neq A$ **and**
X PerpAt A B C D
shows *X B Perp C D* ∨ *A X Perp C D*
by (*metis assms*(*2*) *l8-14-2-1a perp-col perp-in-col*)

lemma *perp-in-perp-bis*:
assumes *X PerpAt A B C D*
shows *X B Perp C D* ∨ *A X Perp C D*
by (*metis assms l8-14-2-1a perp-in-perp-bis-R1*)

lemma *col-per-perp*:
assumes $A \neq B$ **and**
$B \neq C$ **and**

$D \neq C$ **and**
Col B C D **and**
Per A B C
shows *C D Perp A B*
by (*metis Perp-cases assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *col-trivial-2 per-perp perp-col2-bis*)

lemma *per-cong-mid-R1*:
assumes $B = H$ **and**

Bet A B C **and**
Cong A H C H **and**
Per H B C
shows *B Midpoint A C*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *midpoint-def not-cong-1243* **by** *blast*

lemma *per-cong-mid-R2*:
assumes
B ≠ C **and**
Bet A B C **and**
Cong A H C H **and**
Per H B C
shows *B Midpoint A C*
proof −
have *P1*: *Per C B H*
using *Per-cases assms*(*4*) **by** *blast*
have *P2*: *Per H B A*
using *assms*(*1*) *assms*(*2*) *assms*(*4*) *bet-col col-permutation-1 per-col* **by** *blast*
then have *P3*: *Per A B H*
using *Per-cases* **by** *blast*
obtain *C′* **where** *P4*: *B Midpoint C C′ ∧ Cong H C H C′*
using *Per-def assms*(*4*) **by** *blast*
obtain *H′* **where** *P5*: *B Midpoint H H′ ∧ Cong C H C H′*
using *P1 Per-def* **by** *blast*
obtain *A′* **where** *P6*: *B Midpoint A A′ ∧ Cong H A H A′*
using *P2 Per-def* **by** *blast*
obtain *H′′* **where** *P7*: *B Midpoint H H′′ ∧ Cong A H A H′*
using *P3 P5 Tarski-neutral-dimensionless.Per-def Tarski-neutral-dimensionless-axioms symmetric-point-uniqueness*
by *fastforce*
then have *P8*: *H′ = H′′*
using *P5 symmetric-point-uniqueness* **by** *blast*
have *H B H′ A IFSC H B H′ C*
proof −
have *Q1*: *Bet H B H′*
by (*simp add*: *P7 P8 midpoint-bet*)
have *Q2*: *Cong H H′ H H′*
by (*simp add*: *cong-reflexivity*)
have *Q3*: *Cong B H′ B H′*
by (*simp add*: *cong-reflexivity*)
have *Q4*: *Cong H A H C*
using *assms*(*3*) *not-cong-2143* **by** *blast*
have *Cong H′ A H′ C*
using *P5 P7 assms*(*3*) *cong-commutativity cong-inner-transitivity* **by** *blast*
thus *?thesis*
by (*simp add*: *IFSC-def Q1 Q2 Q3 Q4*)
qed
thus *?thesis*
using *assms*(*1*) *assms*(*2*) *bet-col bet-neq23--neq l4-2 l7-20-bis* **by** *auto*
qed

lemma *per-cong-mid*:
assumes *B ≠ C* **and**
Bet A B C **and**
Cong A H C H **and**
Per H B C
shows *B Midpoint A C*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *per-cong-mid-R1 per-cong-mid-R2* **by** *blast*

lemma *per-double-cong*:
assumes *Per A B C* **and**
B Midpoint C C′
shows *Cong A C A C′*
using *Mid-cases Per-def assms*(*1*) *assms*(*2*) *l7-9-bis* **by** *blast*

lemma *cong-perp-or-mid-R1*:
assumes *Col A B X* **and**
A ≠ B **and**
M Midpoint A B **and**
Cong A X B X
shows *X = M ∨ ¬ Col A B X ∧ M PerpAt X M A B*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *col-permutation-5 cong-commutativity l7-17-bis l7-2 l7-20* **by** *blast*

lemma *cong-perp-or-mid-R2*:
assumes ¬ *Col A B X* **and**
A ≠ *B* **and**
M Midpoint A B **and**
Cong A X B X
shows *X* = *M* ∨ ¬ *Col A B X* ∧ *M PerpAt X M A B*
proof −
have *P1*: *Col M A B*
by (*simp add*: *assms*(*3*) *midpoint-col*)
have *Per X M A*
using *Per-def assms*(*3*) *assms*(*4*) *cong-commutativity* **by** *blast*
thus *?thesis*
by (*metis P1 assms*(*1*) *assms*(*2*) *assms*(*3*) *midpoint-distinct-1 not-col-permutation-4 per-perp-in perp-in-col-perp-in perp-in-right-comm*)
qed

lemma *cong-perp-or-mid*:
assumes *A* ≠ *B* **and**
M Midpoint A B **and**
Cong A X B X
shows *X* = *M* ∨ ¬ *Col A B X* ∧ *M PerpAt X M A B*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *cong-perp-or-mid-R1 cong-perp-or-mid-R2* **by** *blast*

lemma *col-per2-cases*:
assumes *B* ≠ *C* **and**
B' ≠ *C* **and**
C ≠ *D* **and**
Col B C D **and**
Per A B C **and**
Per A B' C
shows *B* = *B'* ∨ ¬ *Col B' C D*
by (*meson Tarski-neutral-dimensionless.l8-7 Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *assms*(*6*) *l6-16-1 per-col*)

lemma *l8-16-1*:
assumes *Col A B X* **and**
Col A B U **and**
A B Perp C X
shows ¬ *Col A B C* ∧ *Per C X U*
by (*metis assms*(*1*) *assms*(*2*) *assms*(*3*) *l8-5 perp-col0 perp-left-comm perp-not-col2 perp-per-2*)

lemma *l8-16-2*:
assumes *Col A B X* **and**
Col A B U
and *U* ≠ *X* **and**
¬ *Col A B C* **and**
Per C X U
shows *A B Perp C X*
proof −
obtain *X* **where** *X PerpAt A B C X*
by (*metis* (*no-types*) *NCol-perm assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *l8-13-2 l8-2 not-col-distincts*)
thus *?thesis*
by (*smt Perp-perm assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *col3 col-per-perp not-col-distincts per-col per-perp*)
qed

lemma *l8-18-uniqueness*:
assumes
Col A B X **and**
A B Perp C X **and**
Col A B Y **and**
A B Perp C Y
shows *X* = *Y*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *l8-16-1 l8-7* **by** *blast*

lemma *midpoint-distinct*:
  assumes $\neg$ *Col A B C* **and**
    Col A B X **and**
    X Midpoint C C'
  shows $C \neq C'$
  using *assms(1) assms(2) assms(3) l7-3* **by** *auto*

lemma *l8-20-1-R1*:
  assumes $A = B$
  shows *Per B A P*
  by (*simp add*: *assms l8-2 l8-5*)

lemma *l8-20-1-R2*:
  assumes $A \neq B$ **and**
    Per A B C **and**
    P Midpoint C' D **and**
    A Midpoint C' C **and**
    B Midpoint D C
  shows *Per B A P*
proof −
  obtain *B'* **where** *P1*: *A Midpoint B B'*
    using *symmetric-point-construction* **by** *blast*
  obtain *D'* **where** *P2*: *A Midpoint D D'*
    using *symmetric-point-construction* **by** *blast*
  obtain *P'* **where** *P3*: *A Midpoint P P'*
    using *symmetric-point-construction* **by** *blast*
  have *P4*: *Per B' B C*
   by (*metis P1 Tarski-neutral-dimensionless.Per-cases Tarski-neutral-dimensionless.per-col Tarski-neutral-dimensionless-axioms assms(1) assms(2) midpoint-col not-col-permutation-4*)
  have *P5*: *Per B B' C'*
  proof −
    have *Per B' B C*
      by (*simp add*: *P4*)
    have *B' B C Cong3 B B' C'*
      by (*meson Cong3-def P1 assms(4) l7-13 l7-2*)
    thus *?thesis*
      using *P4 l8-10* **by** *blast*
  qed
  have *P6*: *B' Midpoint D' C'*
    by (*meson P1 P2 assms(4) assms(5) l7-15 l7-16 l7-2 midpoint-bet midpoint-cong midpoint-def*)
  have *P7*: *P' Midpoint C D'*
    using *P2 P3 assms(3) assms(4) symmetry-preserves-midpoint* **by** *blast*
  have *P8*: *A Midpoint P P'*
    by (*simp add*: *P3*)
  obtain *D''* **where** *P9*: *B Midpoint C D''* $\wedge$ *Cong B' C B' D*
    using *P4 assms(5) l7-2 per-double-cong* **by** *blast*
  have *P10*: $D'' = D$
    using *P9 assms(5) l7-9-bis* **by** *blast*
  obtain *D''* **where** *P11*: *B' Midpoint C' D''* $\wedge$ *Cong B C' B D''*
    using *P5 Per-def* **by** *blast*
  have *P12*: $D' = D''$
    by (*meson P11 P6 Tarski-neutral-dimensionless.l7-9-bis Tarski-neutral-dimensionless-axioms*)
  have *P13*: *P Midpoint C' D*
    using *assms(3)* **by** *blast*
  have *P14*: *Cong C D C' D'*
    using *P2 assms(4) l7-13 l7-2* **by** *blast*
  have *P15*: *Cong C' D C D'*
    using *P2 assms(4) cong-4321 l7-13* **by** *blast*
  have *P16*: *Cong P D P' D'*
    using *P2 P8 cong-symmetry l7-13* **by** *blast*
  have *P17*: *Cong P D P' C*
    using *P16 P7 cong-3421 cong-transitivity midpoint-cong* **by** *blast*
  have *P18*: *C' P D B IFSC D' P' C B*
    by (*metis Bet-cases IFSC-def P10 P11 P12 P13 P15 P17 P7 P9 cong-commutativity cong-right-commutativity l7-13 l7-3-2 midpoint-bet*)
  then have *Cong B P B P'*

using *Tarski-neutral-dimensionless.l4-2 Tarski-neutral-dimensionless-axioms not-cong-2143* **by** *fastforce*
thus *?thesis*
using *P8 Per-def* **by** *blast*
qed

lemma *l8-20-1*:
assumes *Per A B C* **and**
P Midpoint C′ D **and**
A Midpoint C′ C **and**
B Midpoint D C
shows *Per B A P*
using *assms(1) assms(2) assms(3) assms(4) l8-20-1-R1 l8-20-1-R2* **by** *fastforce*

lemma *l8-20-2*:
assumes *P Midpoint C′ D* **and**
A Midpoint C′ C **and**
B Midpoint D C **and**
$B \neq C$
shows $A \neq P$
using *assms(1) assms(2) assms(3) assms(4) l7-3 symmetric-point-uniqueness* **by** *blast*

lemma *perp-col1*:
assumes $C \neq X$ **and**
A B Perp C D **and**
Col C D X
shows *A B Perp C X*
using *assms(1) assms(2) assms(3) col-trivial-3 perp-col2-bis* **by** *blast*

lemma *l8-18-existence*:
assumes ¬ *Col A B C*
shows ∃ *X. Col A B X* ∧ *A B Perp C X*
proof −
obtain *Y* **where** *P1*: *Bet B A Y* ∧ *Cong A Y A C*
using *segment-construction* **by** *blast*
then obtain *P* **where** *P2*: *P Midpoint C Y*
using *Mid-cases l7-25* **by** *blast*
then have *P3*: *Per A P Y*
using *P1 Per-def l7-2* **by** *blast*
obtain *Z* **where** *P3*: *Bet A Y Z* ∧ *Cong Y Z Y P*
using *segment-construction* **by** *blast*
obtain *Q* **where** *P4*: *Bet P Y Q* ∧ *Cong Y Q Y A*
using *segment-construction* **by** *blast*
obtain *Q′* **where** *P5*: *Bet Q Z Q′* ∧ *Cong Z Q′ Q Z*
using *segment-construction* **by** *blast*
then have *P6*: *Z Midpoint Q Q′*
using *midpoint-def not-cong-3412* **by** *blast*
obtain *C′* **where** *P7*: *Bet Q′ Y C′* ∧ *Cong Y C′ Y C*
using *segment-construction* **by** *blast*
obtain *X* **where** *P8*: *X Midpoint C C′*
using *Mid-cases P7 l7-25* **by** *blast*
have *P9*: *A Y Z Q OFSC Q Y P A*
by (*simp add*: *OFSC-def P3 P4 between-symmetry cong-4321 cong-pseudo-reflexivity*)
have *P10*: $A \neq Y$
using *P1 assms cong-reverse-identity not-col-distincts* **by** *blast*
then have *P11*: *Cong Z Q P A*
using *P9 five-segment-with-def* **by** *blast*
then have *P12*: *A P Y Cong3 Q Z Y*
using *Cong3-def P3 P4 not-cong-4321* **by** *blast*
have *P13*: *Per Q Z Y*
using *Cong-perm P1 P12 P2 Per-def l8-10 l8-4* **by** *blast*
then have *P14*: *Per Y Z Q*
by (*simp add*: *l8-2*)
have *P15*: $P \neq Y$
using *NCol-cases P1 P2 assms bet-col l7-3-2 l7-9-bis* **by** *blast*
obtain *Q″* **where** *P16*:*Z Midpoint Q Q″* ∧ *Cong Y Q Y Q′*
using *P14 P6 per-double-cong* **by** *blast*

then have *P17*: $Q' = Q''$
  using *P6 symmetric-point-uniqueness* **by** *blast*
 have *P18*: *Bet Z Y X*
 proof −
  have *Bet Q Y C*
   using *P15 P2 P4 between-symmetry midpoint-bet outer-transitivity-between2* **by** *blast*
  thus *?thesis*
   using *P16 P6 P7 P8 l7-22 not-cong-3412* **by** *blast*
 qed
 have *P19*: $Q \neq Y$
  using *P10 P4 cong-reverse-identity* **by** *blast*
 have *P20*: *Per Y X C*
 proof −
  have *Bet C P Y*
   by (*simp add*: *P2 midpoint-bet*)
  thus *?thesis*
   using *P7 P8 Per-def not-cong-3412* **by** *blast*
 qed
 have *P21*: *Col P Y Q*
  by (*simp add*: *Col-def P4*)
 have *P22*: *Col P Y C*
  using *P2 midpoint-col not-col-permutation-5* **by** *blast*
 have *P23*: *Col P Q C*
  using *P15 P21 P22 col-transitivity-1* **by** *blast*
 have *P24*: *Col Y Q C*
  using *P15 P21 P22 col-transitivity-2* **by** *auto*
 have *P25*: *Col A Y B*
  by (*simp add*: *Col-def P1*)
 have *P26*: *Col A Y Z*
  using *P3 bet-col* **by** *blast*
 have *P27*: *Col A B Z*
  using *P10 P25 P26 col-transitivity-1* **by** *blast*
 have *P28*: *Col Y B Z*
  using *P10 P25 P26 col-transitivity-2* **by** *blast*
 have *P29*: *Col Q Y P*
  using *P21 not-col-permutation-3* **by** *blast*
 have *P30*: $Q \neq C$
  using *P15 P2 P4 between-equality-2 between-symmetry midpoint-bet* **by** *blast*
 have *P31*: *Col Y B Z*
  using *P28* **by** *auto*
 have *P32*: *Col Y* Q' C'
  by (*simp add*: *P7 bet-col col-permutation-4*)
 have *P33*: $Q \neq Q'$
  using *P11 P15 P22 P25 P5 assms bet-neq12--neq col-transitivity-1 cong-reverse-identity* **by** *blast*
 have *P34*: $C \neq C'$
  by (*smt P15 P18 P3 P31 P8 assms bet-col col3 col-permutation-2 col-permutation-3 cong-3421 cong-diff midpoint-distinct-3*)
 have *P35*: *Q Y C Z OFSC* Q' *Y* C' *Z*
  by (*meson OFSC-def P15 P16 P2 P4 P5 P7 between-symmetry cong-3421 cong-reflexivity midpoint-bet not-cong-3412 outer-transitivity-between2*)
 then have *P36*: *Cong C Z* C' *Z*
  using *P19 five-segment-with-def* **by** *blast*
 have *P37*: *Col Z Y X*
  by (*simp add*: *P18 bet-col*)
 have *P38*: $Y \neq Z$
  using *P15 P3 cong-reverse-identity* **by** *blast*
 then have *P40*: $X \neq Y$
  by (*metis* (*mono-tags*, *opaque-lifting*) *Col-perm Cong-perm P14 P24 P25 P27 P36 P8 Per-def assms colx per-not-colp*)
 have *Col A B X*
  using *Col-perm P26 P31 P37 P38 col3* **by** *blast*
 thus *?thesis*
  by (*metis P18 P20 P27 P37 P40 Tarski-neutral-dimensionless.per-col Tarski-neutral-dimensionless-axioms assms between-equality col-permutation-3 l5-2 l8-16-2 l8-2*)
qed

lemma *l8-21-aux*:

assumes ¬ *Col A B C*
shows ∃ *P*. ∃ *T*. (*A B Perp P A* ∧ *Col A B T* ∧ *Bet C T P*)
proof −
obtain *X* **where** *P1*: *Col A B X* ∧ *A B Perp C X*
using *assms l8-18-existence* **by** *blast*
have *P2*: *X PerpAt A B C X*
by (*simp add*: *P1 l8-15-1*)
have *P3*: *Per A X C*
by (*meson P1 Per-perm Tarski-neutral-dimensionless.l8-16-1 Tarski-neutral-dimensionless-axioms col-trivial-3*)
obtain *C′* **where** *P4*: *X Midpoint C C′* ∧ *Cong A C A C′*
using *P3 Per-def* **by** *blast*
obtain *C′′* **where** *P5*: *A Midpoint C C′′*
using *symmetric-point-construction* **by** *blast*
obtain *P* **where** *P6*: *P Midpoint C′ C′′*
by (*metis Cong-perm P4 P5 Tarski-neutral-dimensionless.Midpoint-def Tarski-neutral-dimensionless-axioms cong-inner-transitivity l7-25*)
have *P7*: *Per X A P*
by (*smt P3 P4 P5 P6 l7-2 l8-20-1-R2 l8-4 midpoint-distinct-3 symmetric-point-uniqueness*)
have *P8*: *X* ≠ *C*
using *P1 assms* **by** *auto*
have *P9*: *A* ≠ *P*
using *P4 P5 P6 P8 l7-9 midpoint-distinct-2* **by** *blast*
obtain *T* **where** *P10*: *Bet P T C* ∧ *Bet A T X*
by (*meson Mid-perm Midpoint-def P4 P5 P6 l3-17*)
have *A B Perp P A* ∧ *Col A B T* ∧ *Bet C T P*
proof *cases*
assume *A* = *X*
thus *?thesis*
by (*metis Bet-perm Col-def P1 P10 P9 between-identity col-trivial-3 perp-col2-bis*)
next
assume *A* ≠ *X*
thus *?thesis*
by (*metis Bet-perm Col-def P1 P10 P7 P9 Perp-perm col-transitivity-2 col-trivial-1 l8-3 per-perp perp-not-col2*)
qed
thus *?thesis*
by *blast*
qed

lemma *l8-21*:
assumes *A* ≠ *B*
shows ∃ *P T*. *A B Perp P A* ∧ *Col A B T* ∧ *Bet C T P*
by (*meson assms between-trivial2 l8-21-aux not-col-exists*)

lemma *per-cong*:
assumes *A* ≠ *B* **and**
A ≠ *P* **and**
Per B A P **and**
Per A B R **and**
Cong A P B R **and**
Col A B X **and**
Bet P X R
shows *Cong A R P B*
proof −
have *P1*: *Per P A B*
using *Per-cases assms*(*3*) **by** *blast*
obtain *Q* **where** *P2*: *R Midpoint B Q*
using *symmetric-point-construction* **by** *auto*
have *P3*: *B* ≠ *R*
using *assms*(*2*) *assms*(*5*) *cong-identity* **by** *blast*
have *P4*: *Per A B Q*
by (*metis P2 P3 assms*(*1*) *assms*(*4*) *bet-neq23--neq col-permutation-4 midpoint-bet midpoint-col per-perp-in perp-in-col-perp-in perp-in-per*)
have *P5*: *Per P A X*
using *P1 assms*(*1*) *assms*(*6*) *per-col* **by** *blast*
have *P6*: *B* ≠ *Q*
using *P2 P3 l7-3* **by** *blast*

```
  have P7: Per R B X
    by (metis assms(1) assms(4) assms(6) l8-2 not-col-permutation-4 per-col)
  have P8: X ≠ A
    using P3 assms(1) assms(2) assms(3) assms(4) assms(7) bet-col per-not-colp by blast
  obtain P′ where P9: A Midpoint P P′
    using Per-def assms(3) by blast
  obtain R′ where P10: Bet P′ X R′ ∧ Cong X R′ X R
    using segment-construction by blast
  obtain M where P11: M Midpoint R R′
    by (meson P10 Tarski-neutral-dimensionless.l7-2 Tarski-neutral-dimensionless-axioms l7-25)
  have P12: Per X M R
    using P10 P11 Per-def cong-symmetry by blast
  have P13: Cong X P X P′
    using P9 assms(1) assms(3) assms(6) cong-left-commutativity l4-17 midpoint-cong per-double-cong by blast
  have P14: X ≠ P′
    using P13 P8 P9 cong-identity l7-3 by blast
  have P15: P ≠ P′
    using P9 assms(2) midpoint-distinct-2 by blast
  have P16: ¬ Col X P P′
    using P13 P15 P8 P9 l7-17 l7-20 not-col-permutation-4 by blast
  have P17: Bet A X M
    using P10 P11 P13 P9 assms(7) cong-symmetry l7-22 by blast
  have P18: X ≠ R
    using P3 P7 per-distinct-1 by blast
  have P19: X ≠ R′
    using P10 P18 cong-diff-3 by blast
  have P20: X ≠ M
    by (metis Col-def P10 P11 P16 P18 P19 assms(7) col-transitivity-1 midpoint-col)
  have P21: M = B
    by (smt Col-def P12 P17 P20 P8 Per-perm assms(1) assms(4) assms(6) col-transitivity-2 l8-3 l8-7)
  have P X R P′ OFSC P′ X R′ P
    by (simp add: OFSC-def P10 P13 assms(7) cong-commutativity cong-pseudo-reflexivity cong-symmetry)
  then have Cong R P′ R′ P
    using P13 P14 cong-diff-3 five-segment-with-def by blast
  then have P′ A P R IFSC R′ B R P
   by (metis Bet-perm Cong-perm Midpoint-def P11 P21 P9 Tarski-neutral-dimensionless.IFSC-def Tarski-neutral-dimensionless-axioms
assms(5) cong-mid2--cong cong-pseudo-reflexivity)
  thus ?thesis
    using l4-2 not-cong-1243 by blast
qed

lemma perp-cong:
  assumes A ≠ B and
    A ≠ P and
    A B Perp P A and
    A B Perp R B and
    Cong A P B R and
    Col A B X and
    Bet P X R
  shows Cong A R P B
  using Perp-cases assms(1) assms(2) assms(3) assms(4) assms(5) assms(6) assms(7) per-cong perp-per-1 by blast

lemma perp-exists:
  assumes A ≠ B
  shows ∃ X. PO X Perp A B
proof cases
  assume Col A B PO
  then obtain C where P1: A ≠ C ∧ B ≠ C ∧ PO ≠ C ∧ Col A B C
    using diff-col-ex3 by blast
  then obtain P T where P2: PO C Perp P PO ∧ Col PO C T ∧ Bet PO T P using l8-21
    by blast
  then have PO P Perp A B
    by (metis P1 Perp-perm ‹Col A B PO› assms col3 col-trivial-2 col-trivial-3 perp-col2)
  thus ?thesis
    by blast
next
```

assume ¬ *Col A B PO*
thus *?thesis* **using** *l8-18-existence*
using *assms col-trivial-2 col-trivial-3 l8-18-existence perp-col0* **by** *blast*
qed

lemma *perp-vector*:
assumes $A \neq B$
shows ∃ *X Y*. *A B Perp X Y*
using *assms l8-21* **by** *blast*

lemma *midpoint-existence-aux*:
assumes $A \neq B$ **and**
A B Perp Q B **and**
A B Perp P A **and**
Col A B T **and**
Bet Q T P **and**
A P Le B Q
shows ∃ *X*. *X Midpoint A B*
proof −
obtain *R* **where** *P1*: *Bet B R Q* ∧ *Cong A P B R*
using *Le-def assms*(*6*) **by** *blast*
obtain *X* **where** *P2*: *Bet T X B* ∧ *Bet R X P*
using *P1 assms*(*5*) *between-symmetry inner-pasch* **by** *blast*
have *P3*: *Col A B X*
by (*metis Col-def Out-cases P2 assms*(*4*) *between-equality l6-16-1 not-out-bet out-diff1*)
have *P4*: $B \neq R$
using *P1 assms*(*3*) *cong-identity perp-not-eq-2* **by** *blast*
have *P5*: ¬ *Col A B Q*
using *assms*(*2*) *col-trivial-2 l8-16-1* **by** *blast*
have *P6*: ¬ *Col A B R*
using *Col-def P1 P4 P5 l6-16-1* **by** *blast*
have *P7*: $P \neq R$
using *P2 P3 P6 between-identity* **by** *blast*
have ∃ *X*. *X Midpoint A B*
proof *cases*
assume $A = P$
thus *?thesis*
using *assms*(*3*) *col-trivial-3 perp-not-col2* **by** *blast*
next
assume *Q1*: ¬ $A = P$
have *Q2*: *A B Perp R B*
by (*metis P1 P4 Perp-perm Tarski-neutral-dimensionless.bet-col1 Tarski-neutral-dimensionless-axioms assms*(*2*) *l5-1 perp-col1*)
then have *Q3*: *Cong A R P B*
using *P1 P2 P3 Q1 assms*(*1*) *assms*(*3*) *between-symmetry perp-cong* **by** *blast*
then have *X Midpoint A B* ∧ *X Midpoint P R*
by (*smt P1 P2 P3 P6 P7 bet-col cong-left-commutativity cong-symmetry l7-2 l7-21 not-col-permutation-1*)
thus *?thesis*
by *blast*
qed
thus *?thesis* **by** *blast*
qed

lemma *midpoint-existence*:
∃ *X*. *X Midpoint A B*
proof *cases*
assume $A = B$
thus *?thesis*
using *l7-3-2* **by** *blast*
next
assume *P1*: ¬ $A = B$
obtain *Q* **where** *P2*: *A B Perp B Q*
by (*metis P1 l8-21 perp-comm*)
obtain *P T* **where** *P3*: *A B Perp P A* ∧ *Col A B T* ∧ *Bet Q T P*
using *P2 l8-21-aux not-col-distincts perp-not-col2* **by** *blast*
have *P4*: *A P Le B Q* ∨ *B Q Le A P*

by (*simp add*: *local.le-cases*)
have *P5*: *A P Le B Q* ⟶ (∃ *X*. *X Midpoint A B*)
by (*meson P1 P2 P3 Tarski-neutral-dimensionless.Perp-cases Tarski-neutral-dimensionless.midpoint-existence-aux Tarski-neutral-dimensionless-axioms*)
have *P6*: *B Q Le A P* ⟶ (∃ *X*. *X Midpoint A B*)
proof −
{
assume *H1*: *B Q Le A P*
have *Q6*: *B* ≠ *A*
using *P1* **by** *auto*
have *Q2*: *B A Perp P A*
by (*simp add*: *P3 perp-left-comm*)
have *Q3*: *B A Perp Q B*
using *P2 Perp-perm* **by** *blast*
have *Q4*: *Col B A T*
using *Col-perm P3* **by** *blast*
have *Q5*: *Bet P T Q*
using *Bet-perm P3* **by** *blast*
obtain *X* **where** *X Midpoint B A*
using *H1 Q2 Q3 Q4 Q5 Q6 midpoint-existence-aux* **by** *blast*
then have ∃ *X*. *X Midpoint A B*
using *l7-2* **by** *blast*
}
thus *?thesis*
by *simp*
qed
thus *?thesis*
using *P4 P5* **by** *blast*
qed

lemma *perp-in-id*:
assumes *X PerpAt A B C A*
shows *X* = *A*
by (*meson Col-cases assms col-trivial-3 l8-14-2-1b*)

lemma *l8-22*:
assumes *A* ≠ *B* **and**
A ≠ *P* **and**
Per B A P **and**
Per A B R **and**
Cong A P B R **and**
Col A B X **and**
Bet P X R **and**
Cong A R P B
shows *X Midpoint A B* ∧ *X Midpoint P R*
by (*metis assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *assms*(*6*) *assms*(*7*) *assms*(*8*) *bet-col cong-commutativity cong-diff cong-right-commutativity l7-21 not-col-permutation-5 per-not-colp*)

lemma *l8-22-bis*:
assumes *A* ≠ *B* **and**
A ≠ *P* **and**
A B Perp P A **and**
A B Perp R B **and**
Cong A P B R **and**
Col A B X **and**
Bet P X R
shows *Cong A R P B* ∧ *X Midpoint A B* ∧ *X Midpoint P R*
by (*metis l8-22 Perp-cases assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *assms*(*6*) *assms*(*7*) *perp-cong perp-per-2*)

lemma *perp-in-perp*:
assumes *X PerpAt A B C D*
shows *A B Perp C D*
using *assms l8-14-2-1a* **by** *auto*

lemma *perp-proj*:
assumes *A B Perp C D* **and**

¬ *Col A C D*
shows ∃ *X*. *Col A B X* ∧ *A X Perp C D*
using *assms*(*1*) *not-col-distincts* **by** *auto*

lemma *l8-24* :
assumes *P A Perp A B* **and**
Q B Perp A B **and**
Col A B T **and**
Bet P T Q **and**
Bet B R Q **and**
Cong A P B R
shows ∃ *X*. *X Midpoint A B* ∧ *X Midpoint P R*
proof −
obtain *X* **where** *P1*: *Bet T X B* ∧ *Bet R X P*
using *assms*(*4*) *assms*(*5*) *inner-pasch* **by** *blast*
have *P2*: *Col A B X*
by (*metis Out-cases P1 assms*(*3*) *bet-out-1 col-out2-col not-col-distincts out-trivial*)
have *P3*: *A* ≠ *B*
using *assms*(*1*) *col-trivial-2 l8-16-1* **by** *blast*
have *P4*: *A* ≠ *P*
using *assms*(*1*) *col-trivial-1 l8-16-1* **by** *blast*
have ∃ *X*. *X Midpoint A B* ∧ *X Midpoint P R*
proof *cases*
assume *Col A B P*
thus *?thesis*
using *Perp-perm assms*(*1*) *perp-not-col* **by** *blast*
next
assume *Q1*: ¬ *Col A B P*
have *Q2*: *B* ≠ *R*
using *P4 assms*(*6*) *cong-diff* **by** *blast*
have *Q3*: *Q* ≠ *B*
using *Q2 assms*(*5*) *between-identity* **by** *blast*
have *Q4*: ¬ *Col A B Q*
by (*metis assms*(*2*) *col-permutation-3 l8-14-1 perp-col1 perp-not-col*)
have *Q5*: ¬ *Col A B R*
by (*meson Q2 Q4 assms*(*5*) *bet-col col-transitivity-1 not-col-permutation-2*)
have *Q6*: *P* ≠ *R*
using *P1 P2 Q5 between-identity* **by** *blast*
have ∃ *X*. *X Midpoint A B* ∧ *X Midpoint P R*
proof *cases*
assume *A* = *P*
thus *?thesis*
using *P4* **by** *blast*
next
assume *R0*: ¬ *A* = *P*
have *R1*: *A B Perp R B*
by (*metis Perp-cases Q2 Tarski-neutral-dimensionless.bet-col1 Tarski-neutral-dimensionless-axioms assms*(*2*) *assms*(*5*) *bet-col col-transitivity-1 perp-col1*)
have *R2*: *Cong A R P B*
using *P1 P2 P3 Perp-perm R0 R1 assms*(*1*) *assms*(*6*) *between-symmetry perp-cong* **by** *blast*
have *R3*: ¬ *Col A P B*
using *Col-perm Q1* **by** *blast*
have *R4*: *P* ≠ *R*
by (*simp add*: *Q6*)
have *R5*: *Cong A P B R*
by (*simp add*: *assms*(*6*))
have *R6*: *Cong P B R A*
using *R2 not-cong-4312* **by** *blast*
have *R7*: *Col A X B*
using *Col-perm P2* **by** *blast*
have *R8*: *Col P X R*
by (*simp add*: *P1 bet-col between-symmetry*)
thus *?thesis* **using** *l7-21*
using *R3 R4 R5 R6 R7* **by** *blast*
qed
thus *?thesis* **by** *simp*

qed
thus *?thesis*
by *simp*
qed

lemma *col-per2--per*:
assumes $A \neq B$ **and**
Col A B C **and**
Per A X P **and**
Per B X P
shows *Per C X P*
by (*meson Per-def assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *l4-17 per-double-cong*)

lemma *perp-in-per-1*:
assumes *X PerpAt A B C D*
shows *Per A X C*
using *PerpAt-def assms col-trivial-1* **by** *auto*

lemma *perp-in-per-2*:
assumes *X PerpAt A B C D*
shows *Per A X D*
using *assms perp-in-per-1 perp-in-right-comm* **by** *blast*

lemma *perp-in-per-3*:
assumes *X PerpAt A B C D*
shows *Per B X C*
using *assms perp-in-comm perp-in-per-2* **by** *blast*

lemma *perp-in-per-4*:
assumes *X PerpAt A B C D*
shows *Per B X D*
using *assms perp-in-per-3 perp-in-right-comm* **by** *blast*

3.8 Planes

3.8.1 Coplanar

lemma *coplanar-perm-1*:
assumes *Coplanar A B C D*
shows *Coplanar A B D C*
proof −
obtain *X* **where** *P1*: $(Col\ A\ B\ X \wedge Col\ C\ D\ X) \vee (Col\ A\ C\ X \wedge Col\ B\ D\ X) \vee (Col\ A\ D\ X \wedge Col\ B\ C\ X)$
using *Coplanar-def assms* **by** *blast*
then show *?thesis*
using *Coplanar-def col-permutation-4* **by** *blast*
qed

lemma *coplanar-perm-2*:
assumes *Coplanar A B C D*
shows *Coplanar A C B D*
proof −
obtain *X* **where** *P1*: $(Col\ A\ B\ X \wedge Col\ C\ D\ X) \vee (Col\ A\ C\ X \wedge Col\ B\ D\ X) \vee (Col\ A\ D\ X \wedge Col\ B\ C\ X)$
using *Coplanar-def assms* **by** *blast*
then show *?thesis*
using *Coplanar-def col-permutation-4* **by** *blast*
qed

lemma *coplanar-perm-3*:
assumes *Coplanar A B C D*
shows *Coplanar A C D B*
proof −
obtain *X* **where** *P1*: $(Col\ A\ B\ X \wedge Col\ C\ D\ X) \vee (Col\ A\ C\ X \wedge Col\ B\ D\ X) \vee (Col\ A\ D\ X \wedge Col\ B\ C\ X)$
using *Coplanar-def assms* **by** *blast*
then show *?thesis*
using *Coplanar-def col-permutation-4* **by** *blast*
qed

lemma *coplanar-perm-4*:
assumes *Coplanar A B C D*
shows *Coplanar A D B C*
proof –
obtain *X* **where** *P1*: (*Col A B X* ∧ *Col C D X*) ∨ (*Col A C X* ∧ *Col B D X*) ∨ (*Col A D X* ∧ *Col B C X*)
using *Coplanar-def assms* **by** *blast*
then show *?thesis*
using *Coplanar-def col-permutation-4* **by** *blast*
qed

lemma *coplanar-perm-5*:
assumes *Coplanar A B C D*
shows *Coplanar A D C B*
proof –
obtain *X* **where** *P1*: (*Col A B X* ∧ *Col C D X*) ∨ (*Col A C X* ∧ *Col B D X*) ∨ (*Col A D X* ∧ *Col B C X*)
using *Coplanar-def assms* **by** *blast*
then show *?thesis*
using *Coplanar-def col-permutation-4* **by** *blast*
qed

lemma *coplanar-perm-6*:
assumes *Coplanar A B C D*
shows *Coplanar B A C D*
proof –
obtain *X* **where** *P1*: (*Col A B X* ∧ *Col C D X*) ∨ (*Col A C X* ∧ *Col B D X*) ∨ (*Col A D X* ∧ *Col B C X*)
using *Coplanar-def assms* **by** *blast*
then show *?thesis*
using *Coplanar-def col-permutation-4* **by** *blast*
qed

lemma *coplanar-perm-7*:
assumes *Coplanar A B C D*
shows *Coplanar B A D C*
proof –
obtain *X* **where** *P1*: (*Col A B X* ∧ *Col C D X*) ∨ (*Col A C X* ∧ *Col B D X*) ∨ (*Col A D X* ∧ *Col B C X*)
using *Coplanar-def assms* **by** *blast*
then show *?thesis*
using *Coplanar-def col-permutation-4* **by** *blast*
qed

lemma *coplanar-perm-8*:
assumes *Coplanar A B C D*
shows *Coplanar B C A D*
proof –
obtain *X* **where** *P1*: (*Col A B X* ∧ *Col C D X*) ∨ (*Col A C X* ∧ *Col B D X*) ∨ (*Col A D X* ∧ *Col B C X*)
using *Coplanar-def assms* **by** *blast*
then show *?thesis*
using *Coplanar-def col-permutation-4* **by** *blast*
qed

lemma *coplanar-perm-9*:
assumes *Coplanar A B C D*
shows *Coplanar B C D A*
proof –
obtain *X* **where** *P1*: (*Col A B X* ∧ *Col C D X*) ∨ (*Col A C X* ∧ *Col B D X*) ∨ (*Col A D X* ∧ *Col B C X*)
using *Coplanar-def assms* **by** *blast*
then show *?thesis*
using *Coplanar-def col-permutation-4* **by** *blast*
qed

lemma *coplanar-perm-10*:
assumes *Coplanar A B C D*
shows *Coplanar B D A C*
proof –
obtain *X* **where** *P1*: (*Col A B X* ∧ *Col C D X*) ∨ (*Col A C X* ∧ *Col B D X*) ∨ (*Col A D X* ∧ *Col B C X*)
using *Coplanar-def assms* **by** *blast*

then show *?thesis*
using *Coplanar-def col-permutation-4* **by** *blast*
qed

lemma *coplanar-perm-11*:
assumes *Coplanar A B C D*
shows *Coplanar B D C A*
proof –
obtain *X* **where** *P1*: (*Col A B X* ∧ *Col C D X*) ∨ (*Col A C X* ∧ *Col B D X*) ∨ (*Col A D X* ∧ *Col B C X*)
using *Coplanar-def assms* **by** *blast*
then show *?thesis*
using *Coplanar-def col-permutation-4* **by** *blast*
qed

lemma *coplanar-perm-12*:
assumes *Coplanar A B C D*
shows *Coplanar C A B D*
proof –
obtain *X* **where** *P1*: (*Col A B X* ∧ *Col C D X*) ∨ (*Col A C X* ∧ *Col B D X*) ∨ (*Col A D X* ∧ *Col B C X*)
using *Coplanar-def assms* **by** *blast*
then show *?thesis*
using *Coplanar-def col-permutation-4* **by** *blast*
qed

lemma *coplanar-perm-13*:
assumes *Coplanar A B C D*
shows *Coplanar C A D B*
proof –
obtain *X* **where** *P1*: (*Col A B X* ∧ *Col C D X*) ∨ (*Col A C X* ∧ *Col B D X*) ∨ (*Col A D X* ∧ *Col B C X*)
using *Coplanar-def assms* **by** *blast*
then show *?thesis*
using *Coplanar-def col-permutation-4* **by** *blast*
qed

lemma *coplanar-perm-14*:
assumes *Coplanar A B C D*
shows *Coplanar C B A D*
proof –
obtain *X* **where** *P1*: (*Col A B X* ∧ *Col C D X*) ∨ (*Col A C X* ∧ *Col B D X*) ∨ (*Col A D X* ∧ *Col B C X*)
using *Coplanar-def assms* **by** *blast*
then show *?thesis*
using *Coplanar-def col-permutation-4* **by** *blast*
qed

lemma *coplanar-perm-15*:
assumes *Coplanar A B C D*
shows *Coplanar C B D A*
proof –
obtain *X* **where** *P1*: (*Col A B X* ∧ *Col C D X*) ∨ (*Col A C X* ∧ *Col B D X*) ∨ (*Col A D X* ∧ *Col B C X*)
using *Coplanar-def assms* **by** *blast*
then show *?thesis*
using *Coplanar-def col-permutation-4* **by** *blast*
qed

lemma *coplanar-perm-16*:
assumes *Coplanar A B C D*
shows *Coplanar C D A B*
proof –
obtain *X* **where** *P1*: (*Col A B X* ∧ *Col C D X*) ∨ (*Col A C X* ∧ *Col B D X*) ∨ (*Col A D X* ∧ *Col B C X*)
using *Coplanar-def assms* **by** *blast*
then show *?thesis*
using *Coplanar-def col-permutation-4* **by** *blast*
qed

lemma *coplanar-perm-17*:
assumes *Coplanar A B C D*

shows *Coplanar C D B A*
proof –
obtain *X* **where** *P1*: (*Col A B X* ∧ *Col C D X*) ∨ (*Col A C X* ∧ *Col B D X*) ∨ (*Col A D X* ∧ *Col B C X*)
using *Coplanar-def assms* **by** *blast*
then show *?thesis*
using *Coplanar-def col-permutation-4* **by** *blast*
qed

lemma *coplanar-perm-18*:
assumes *Coplanar A B C D*
shows *Coplanar D A B C*
proof –
obtain *X* **where** *P1*: (*Col A B X* ∧ *Col C D X*) ∨ (*Col A C X* ∧ *Col B D X*) ∨ (*Col A D X* ∧ *Col B C X*)
using *Coplanar-def assms* **by** *blast*
then show *?thesis*
using *Coplanar-def col-permutation-4* **by** *blast*
qed

lemma *coplanar-perm-19*:
assumes *Coplanar A B C D*
shows *Coplanar D A C B*
proof –
obtain *X* **where** *P1*: (*Col A B X* ∧ *Col C D X*) ∨ (*Col A C X* ∧ *Col B D X*) ∨ (*Col A D X* ∧ *Col B C X*)
using *Coplanar-def assms* **by** *blast*
then show *?thesis*
using *Coplanar-def col-permutation-4* **by** *blast*
qed

lemma *coplanar-perm-20*:
assumes *Coplanar A B C D*
shows *Coplanar D B A C*
proof –
obtain *X* **where** *P1*: (*Col A B X* ∧ *Col C D X*) ∨ (*Col A C X* ∧ *Col B D X*) ∨ (*Col A D X* ∧ *Col B C X*)
using *Coplanar-def assms* **by** *blast*
then show *?thesis*
using *Coplanar-def col-permutation-4* **by** *blast*
qed

lemma *coplanar-perm-21*:
assumes *Coplanar A B C D*
shows *Coplanar D B C A*
proof –
obtain *X* **where** *P1*: (*Col A B X* ∧ *Col C D X*) ∨ (*Col A C X* ∧ *Col B D X*) ∨ (*Col A D X* ∧ *Col B C X*)
using *Coplanar-def assms* **by** *blast*
then show *?thesis*
using *Coplanar-def col-permutation-4* **by** *blast*
qed

lemma *coplanar-perm-22*:
assumes *Coplanar A B C D*
shows *Coplanar D C A B*
proof –
obtain *X* **where** *P1*: (*Col A B X* ∧ *Col C D X*) ∨ (*Col A C X* ∧ *Col B D X*) ∨ (*Col A D X* ∧ *Col B C X*)
using *Coplanar-def assms* **by** *blast*
then show *?thesis*
using *Coplanar-def col-permutation-4* **by** *blast*
qed

lemma *coplanar-perm-23*:
assumes *Coplanar A B C D*
shows *Coplanar D C B A*
proof –
obtain *X* **where** *P1*: (*Col A B X* ∧ *Col C D X*) ∨ (*Col A C X* ∧ *Col B D X*) ∨ (*Col A D X* ∧ *Col B C X*)
using *Coplanar-def assms* **by** *blast*
then show *?thesis*
using *Coplanar-def col-permutation-4* **by** *blast*

qed

lemma *ncoplanar-perm-1*:
assumes ¬ *Coplanar A B C D*
shows ¬ *Coplanar A B D C*
using *assms coplanar-perm-1* **by** *blast*

lemma *ncoplanar-perm-2*:
assumes ¬ *Coplanar A B C D*
shows ¬ *Coplanar A C B D*
using *assms coplanar-perm-2* **by** *blast*

lemma *ncoplanar-perm-3*:
assumes ¬ *Coplanar A B C D*
shows ¬ *Coplanar A C D B*
using *assms coplanar-perm-4* **by** *blast*

lemma *ncoplanar-perm-4*:
assumes ¬ *Coplanar A B C D*
shows ¬ *Coplanar A D B C*
using *assms coplanar-perm-3* **by** *blast*

lemma *ncoplanar-perm-5*:
assumes ¬ *Coplanar A B C D*
shows ¬ *Coplanar A D C B*
using *assms coplanar-perm-5* **by** *blast*

lemma *ncoplanar-perm-6*:
assumes ¬ *Coplanar A B C D*
shows ¬ *Coplanar B A C D*
using *assms coplanar-perm-6* **by** *blast*

lemma *ncoplanar-perm-7*:
assumes ¬ *Coplanar A B C D*
shows ¬ *Coplanar B A D C*
using *assms coplanar-perm-7* **by** *blast*

lemma *ncoplanar-perm-8*:
assumes ¬ *Coplanar A B C D*
shows ¬ *Coplanar B C A D*
using *assms coplanar-perm-12* **by** *blast*

lemma *ncoplanar-perm-9*:
assumes ¬ *Coplanar A B C D*
shows ¬ *Coplanar B C D A*
using *assms coplanar-perm-18* **by** *blast*

lemma *ncoplanar-perm-10*:
assumes ¬ *Coplanar A B C D*
shows ¬ *Coplanar B D A C*
using *assms coplanar-perm-13* **by** *blast*

lemma *ncoplanar-perm-11*:
assumes ¬ *Coplanar A B C D*
shows ¬ *Coplanar B D C A*
using *assms coplanar-perm-19* **by** *blast*

lemma *ncoplanar-perm-12*:
assumes ¬ *Coplanar A B C D*
shows ¬ *Coplanar C A B D*
using *assms coplanar-perm-8* **by** *blast*

lemma *ncoplanar-perm-13*:
assumes ¬ *Coplanar A B C D*
shows ¬ *Coplanar C A D B*
using *assms coplanar-perm-10* **by** *blast*

lemma *ncoplanar-perm-14*:
assumes ¬ *Coplanar A B C D*
shows ¬ *Coplanar C B A D*
using *assms coplanar-perm-14* **by** *blast*

lemma *ncoplanar-perm-15*:
assumes ¬ *Coplanar A B C D*
shows ¬ *Coplanar C B D A*
using *assms coplanar-perm-20* **by** *blast*

lemma *ncoplanar-perm-16*:
assumes ¬ *Coplanar A B C D*
shows ¬ *Coplanar C D A B*
using *assms coplanar-perm-16* **by** *blast*

lemma *ncoplanar-perm-17*:
assumes ¬ *Coplanar A B C D*
shows ¬ *Coplanar C D B A*
using *assms coplanar-perm-22* **by** *blast*

lemma *ncoplanar-perm-18*:
assumes ¬ *Coplanar A B C D*
shows ¬ *Coplanar D A B C*
using *assms coplanar-perm-9* **by** *blast*

lemma *ncoplanar-perm-19*:
assumes ¬ *Coplanar A B C D*
shows ¬ *Coplanar D A C B*
using *assms coplanar-perm-11* **by** *blast*

lemma *ncoplanar-perm-20*:
assumes ¬ *Coplanar A B C D*
shows ¬ *Coplanar D B A C*
using *assms coplanar-perm-15* **by** *blast*

lemma *ncoplanar-perm-21*:
assumes ¬ *Coplanar A B C D*
shows ¬ *Coplanar D B C A*
using *assms coplanar-perm-21* **by** *blast*

lemma *ncoplanar-perm-22*:
assumes ¬ *Coplanar A B C D*
shows ¬ *Coplanar D C A B*
using *assms coplanar-perm-17* **by** *blast*

lemma *ncoplanar-perm-23*:
assumes ¬ *Coplanar A B C D*
shows ¬ *Coplanar D C B A*
using *assms coplanar-perm-23* **by** *blast*

lemma *coplanar-trivial*:
shows *Coplanar A A B C*
using *Coplanar-def NCol-cases col-trivial-1* **by** *blast*

lemma *col--coplanar*:
assumes *Col A B C*
shows *Coplanar A B C D*
using *Coplanar-def assms not-col-distincts* **by** *blast*

lemma *ncop--ncol*:
assumes ¬ *Coplanar A B C D*
shows ¬ *Col A B C*
using *assms col--coplanar* **by** *blast*

lemma *ncop--ncols*:

assumes ¬ *Coplanar A B C D*
shows ¬ *Col A B C* ∧ ¬ *Col A B D* ∧ ¬ *Col A C D* ∧ ¬ *Col B C D*
by (*meson assms col--coplanar coplanar-perm-4 ncoplanar-perm-9*)

lemma *bet--coplanar*:
assumes *Bet A B C*
shows *Coplanar A B C D*
using *assms bet-col ncop--ncol* **by** *blast*

lemma *out--coplanar*:
assumes *A Out B C*
shows *Coplanar A B C D*
using *assms col--coplanar out-col* **by** *blast*

lemma *midpoint--coplanar*:
assumes *A Midpoint B C*
shows *Coplanar A B C D*
using *assms midpoint-col ncop--ncol* **by** *blast*

lemma *perp--coplanar*:
assumes *A B Perp C D*
shows *Coplanar A B C D*
proof −
obtain *P* **where** *P PerpAt A B C D*
using *Perp-def assms* **by** *blast*
then show *?thesis*
using *Coplanar-def perp-in-col* **by** *blast*
qed

lemma *ts--coplanar*:
assumes *A B TS C D*
shows *Coplanar A B C D*
by (*metis* (*full-types*) *Coplanar-def TS-def assms bet-col col-permutation-2 col-permutation-3*)

lemma *reflectl--coplanar*:
assumes *A B ReflectL C D*
shows *Coplanar A B C D*
by (*metis* (*no-types*) *ReflectL-def Tarski-neutral-dimensionless.perp--coplanar Tarski-neutral-dimensionless-axioms assms col--coplanar col-trivial-1 ncoplanar-perm-17*)

lemma *reflect--coplanar*:
assumes *A B Reflect C D*
shows *Coplanar A B C D*
by (*metis* (*no-types*) *Reflect-def Tarski-neutral-dimensionless.reflectl--coplanar Tarski-neutral-dimensionless-axioms assms col-trivial-2 ncop--ncols*)

lemma *inangle--coplanar*:
assumes *A InAngle B C D*
shows *Coplanar A B C D*
proof −
obtain *X* **where** *P1*: *Bet B X D* ∧ (*X* = *C* ∨ *C Out X A*)
using *InAngle-def assms* **by** *auto*
then show *?thesis*
by (*meson Col-cases Coplanar-def bet-col ncop--ncols out-col*)
qed

lemma *pars--coplanar*:
assumes *A B ParStrict C D*
shows *Coplanar A B C D*
using *ParStrict-def assms* **by** *auto*

lemma *par--coplanar*:
assumes *A B Par C D*
shows *Coplanar A B C D*
using *Par-def assms ncop--ncols pars--coplanar* **by** *blast*

lemma *plg--coplanar*:
assumes *Plg A B C D*
shows *Coplanar A B C D*
proof −
obtain *M* **where** *Bet A M C* ∧ *Bet B M D*
by (*meson Plg-def assms midpoint-bet*)
then show *?thesis*
by (*metis InAngle-def bet-out-1 inangle--coplanar ncop--ncols not-col-distincts*)
qed

lemma *plgs--coplanar*:
assumes *ParallelogramStrict A B C D*
shows *Coplanar A B C D*
using *ParallelogramStrict-def assms par--coplanar* **by** *blast*

lemma *plgf--coplanar*:
assumes *ParallelogramFlat A B C D*
shows *Coplanar A B C D*
using *ParallelogramFlat-def assms col--coplanar* **by** *auto*

lemma *parallelogram--coplanar*:
assumes *Parallelogram A B C D*
shows *Coplanar A B C D*
using *Parallelogram-def assms plgf--coplanar plgs--coplanar* **by** *auto*

lemma *rhombus--coplanar*:
assumes *Rhombus A B C D*
shows *Coplanar A B C D*
using *Rhombus-def assms plg--coplanar* **by** *blast*

lemma *rectangle--coplanar*:
assumes *Rectangle A B C D*
shows *Coplanar A B C D*
using *Rectangle-def assms plg--coplanar* **by** *blast*

lemma *square--coplanar*:
assumes *Square A B C D*
shows *Coplanar A B C D*
using *Square-def assms rectangle--coplanar* **by** *blast*

lemma *lambert--coplanar*:
assumes *Lambert A B C D*
shows *Coplanar A B C D*
using *Lambert-def assms* **by** *presburger*

3.8.2 Planes

lemma *ts-distincts*:
assumes *A B TS P Q*
shows $A \neq B \wedge A \neq P \wedge A \neq Q \wedge B \neq P \wedge B \neq Q \wedge P \neq Q$
using *TS-def assms bet-neq12--neq not-col-distincts* **by** *blast*

lemma *l9-2*:
assumes *A B TS P Q*
shows *A B TS Q P*
using *TS-def assms between-symmetry* **by** *blast*

lemma *invert-two-sides*:
assumes *A B TS P Q*
shows *B A TS P Q*
using *TS-def assms not-col-permutation-5* **by** *blast*

lemma *l9-3*:
assumes *P Q TS A C* **and**
Col M P Q **and**
M Midpoint A C **and**

```
    Col R P Q and
    R Out A B
  shows P Q TS B C
proof –
  have P1: ¬ Col A P Q
    using TS-def assms(1) by blast
  have P2: P ≠ Q
    using P1 not-col-distincts by auto
  obtain T where P3: Col T P Q ∧ Bet A T C
    using assms(2) assms(3) midpoint-bet by blast
  have P4: A ≠ C
    using assms(1) ts-distincts by blast
  have P5: T = M
   by (smt P1 P3 Tarski-neutral-dimensionless.bet-col1 Tarski-neutral-dimensionless-axioms assms(2) assms(3) col-permutation-2
l6-21 midpoint-bet)
  have P Q TS B C
  proof cases
    assume C = M
    then show ?thesis
      using P4 assms(3) midpoint-distinct-1 by blast
  next
    assume P6: ¬ C = M
    have P7: ¬ Col B P Q
      by (metis P1 assms(4) assms(5) col-permutation-1 colx l6-3-1 out-col)
    have P97: Bet R A B ∨ Bet R B A
      using Out-def assms(5) by auto
    {
      assume Q1: Bet R A B
      obtain B′ where Q2: M Midpoint B B′
        using symmetric-point-construction by blast
      obtain R′ where Q3: M Midpoint R R′
        using symmetric-point-construction by blast
      have Q4: Bet B′ C R′
        using Q1 Q2 Q3 assms(3) between-symmetry l7-15 by blast
      obtain X where Q5: Bet M X R′ ∧ Bet C X B
        using Bet-perm Midpoint-def Q2 Q4 between-trivial2 l3-17 by blast
      have Q6: Col X P Q
      proof –
        have R1: Col P M R
          using P2 assms(2) assms(4) col-permutation-4 l6-16-1 by blast
        have R2: Col Q M R
          by (metis R1 assms(2) assms(4) col-permutation-5 l6-16-1 not-col-permutation-3)
        {
          assume M = X
          then have Col X P Q
            using assms(2) by blast
        }
        then have R3: M = X ⟶ Col X P Q by simp
        {
          assume M ≠ X
          then have S1: M ≠ R′
            using Q5 bet-neq12--neq by blast
          have M ≠ R
            using Q3 S1 midpoint-distinct-1 by blast
          then have Col X P Q
            by (smt Col-perm Q3 Q5 R1 R2 S1 bet-out col-transitivity-2 midpoint-col out-col)
        }
        then have M ≠ X ⟶ Col X P Q by simp
        then show ?thesis using R3 by blast
      qed
      have Bet B X C
        using Q5 between-symmetry by blast
      then have P Q TS B C using Q6
        using P7 TS-def assms(1) by blast
    }
    then have P98: Bet R A B ⟶ P Q TS B C by simp
```

```
    {
      assume S2: Bet R B A
      have S3: Bet C M A
        using Bet-perm P3 P5 by blast
      then obtain X where Bet B X C ∧ Bet M X R
        using S2 inner-pasch by blast
      then have P Q TS B C
          by (metis Col-def P7 TS-def assms(1) assms(2) assms(4) between-inner-transitivity between-trivial l6-16-1
not-col-permutation-5)
    }
    then have Bet R B A ⟶ P Q TS B C by simp
    then show ?thesis using P97 P98
      by blast
  qed
  then show ?thesis by blast
qed

lemma mid-preserves-col:
  assumes Col A B C and
    M Midpoint A A' and
    M Midpoint B B' and
    M Midpoint C C'
  shows Col A' B' C'
  using Col-def assms(1) assms(2) assms(3) assms(4) l7-15 by auto

lemma per-mid-per:
  assumes
    Per X A B and
    M Midpoint A B and
    M Midpoint X Y
  shows Cong A X B Y ∧ Per Y B A
  by (meson Cong3-def Mid-perm assms(1) assms(2) assms(3) l7-13 l8-10)

lemma sym-preserve-diff:
  assumes A ≠ B and
    M Midpoint A A' and
    M Midpoint B B'
  shows A'≠ B'
  using assms(1) assms(2) assms(3) l7-9 by blast

lemma l9-4-1-aux-R1:
  assumes R = S and
    S C Le R A and
    P Q TS A C and
    Col R P Q and
    P Q Perp A R and
    Col S P Q and
    P Q Perp C S and
    M Midpoint R S
  shows ∀ U C'. M Midpoint U C' ⟶ (R Out U A ⟷ S Out C C')
proof -
  have P1: M = R
    using assms(1) assms(8) l7-3 by blast
  have P2: ¬ Col A P Q
    using TS-def assms(3) by auto
  then have P3: P ≠ Q
    using not-col-distincts by blast
  obtain T where P4: Col T P Q ∧ Bet A T C
    using TS-def assms(3) by blast
  {
    assume ¬ M = T
    then have M PerpAt M T A M using perp-col2
      by (metis P1 P4 assms(4) assms(5) not-col-permutation-3 perp-left-comm perp-perp-in)
    then have M T Perp C M
      using P1 P4 ‹M ≠ T› assms(1) assms(4) assms(7) col-permutation-1 perp-col2 by blast
    then have Per T M A
```

```
      using ‹M PerpAt M T A M› perp-in-per-3 by blast
    have Per T M C
      by (simp add: ‹M T Perp C M› perp-per-1)
    have M = T
    proof –
      have Per C M T
        by (simp add: ‹Per T M C› l8-2)
      then show ?thesis using l8-6 l8-2
        using P4 ‹Per T M A› by blast
    qed
    then have False
      using ‹M ≠ T› by blast
  }
  then have Q0: M = T by blast
  have R1: ∀ U C'. ((M Midpoint U C' ∧ M Out U A) ⟶ M Out C C')
  proof –
    {
      fix U C'
      assume Q1: M Midpoint U C' ∧ M Out U A
      have Q2: C ≠ M
        using P1 assms(1) assms(7) perp-not-eq-2 by blast
      have Q3: C' ≠ M
        using Q1 midpoint-not-midpoint out-diff1 by blast
      have Q4: Bet U M C
        using P4 Q0 Q1 bet-out--bet l6-6 by blast
      then have M Out C C'
        by (metis (full-types) Out-def Q1 Q2 Q3 l5-2 midpoint-bet)
    }
    then show ?thesis by blast
  qed
  have R2: ∀ U C'. ((M Midpoint U C' ∧ M Out C C') ⟶ M Out U A)
  proof –
    {
      fix U C'
      assume Q1: M Midpoint U C' ∧ M Out C C'
      have Q2: C ≠ M
        using P1 assms(1) assms(7) perp-not-eq-2 by blast
      have Q3: C' ≠ M
        using Q1 l6-3-1 by blast
      have Q4: Bet U M C
        by (metis Out-def Q1 between-inner-transitivity midpoint-bet outer-transitivity-between)
      then have M Out U A
        by (metis P2 P4 Q0 Q1 Q2 Q3 l6-2 midpoint-distinct-1)
    }
    then show ?thesis by blast
  qed
  then show ?thesis
    using R1 P1 P2 assms by blast
qed

lemma l9-4-1-aux-R21:
  assumes R ≠ S and
    S C Le R A and
    P Q TS A C and
    Col R P Q and
    P Q Perp A R and
    Col S P Q and
    P Q Perp C S and
    M Midpoint R S
  shows ∀ U C'. M Midpoint U C' ⟶ (R Out U A ⟷ S Out C C')
proof –
  obtain D where P1: Bet R D A ∧ Cong S C R D
    using Le-def assms(2) by blast
  have P2: C ≠ S
    using assms(7) perp-not-eq-2 by auto
  have P3: R ≠ D
```

```
  using P1 P2 cong-identity by blast
have P4: R S Perp A R
  using assms(1) assms(4) assms(5) assms(6) not-col-permutation-2 perp-col2 by blast
have ∃ M. (M Midpoint S R ∧ M Midpoint C D)
proof –
  have Q1: ¬ Col A P Q
    using TS-def assms(3) by blast
  have Q2: P ≠ Q
    using Q1 not-col-distincts by blast
  obtain T where Q3: Col T P Q ∧ Bet A T C
    using TS-def assms(3) by blast
  have Q4: C S Perp S R
    by (metis NCol-perm assms(1) assms(4) assms(6) assms(7) perp-col0)
  have Q5: A R Perp S R
    using P4 Perp-perm by blast
  have Q6: Col S R T
    using Col-cases Q2 Q3 assms(4) assms(6) col3 by blast
  have Q7: Bet C T A
    using Bet-perm Q3 by blast
  have Q8: Bet R D A
    by (simp add: P1)
  have Cong S C R D
    by (simp add: P1)
  then show ?thesis using P1 Q4 Q5 Q6 Q7 l8-24 by blast
qed
then obtain M' where P5: M' Midpoint S R ∧ M' Midpoint C D by blast
have P6: M = M'
  by (meson P5 assms(8) l7-17-bis)
have L1: ∀ U C'. (M Midpoint U C' ∧ R Out U A) ⟶ S Out C C'
proof –
  {
    fix U C'
    assume R1: M Midpoint U C' ∧ R Out U A
    have R2: C ≠ S
      using P2 by auto
    have R3: C' ≠ S
      using P5 R1 P6 l7-9-bis out-diff1 by blast
    have R4: Bet S C C' ∨ Bet S C' C
    proof –
      have R5: Bet R U A ∨ Bet R A U
        using Out-def R1 by auto
      {
        assume Bet R U A
        then have Bet R U D ∨ Bet R D U
          using P1 l5-3 by blast
        then have Bet S C C' ∨ Bet S C' C
          using P5 P6 R1 l7-15 l7-2 by blast
      }
      then have R6: Bet R U A ⟶ Bet S C C' ∨ Bet S C' C by simp
      have Bet R A U ⟶ Bet S C C' ∨ Bet S C' C
        using P1 P5 P6 R1 between-exchange4 l7-15 l7-2 by blast
      then show ?thesis using R5 R6 by blast
    qed
    then have S Out C C'
      by (simp add: Out-def R2 R3)
  }
  then show ?thesis by simp
qed
have ∀ U C'. (M Midpoint U C' ∧ S Out C C') ⟶ R Out U A
proof –
  {
    fix U C'
    assume Q1: M Midpoint U C' ∧ S Out C C'
    then have Q2: U ≠ R
      using P5 P6 l7-9-bis out-diff2 by blast
    have Q3: A ≠ R
```

        using *assms*(*5*) *perp-not-eq-2* **by** *auto*
      have *Q4*: *Bet S C C′* ∨ *Bet S C′ C*
        using *Out-def Q1* **by** *auto*
      {
        assume *V0*: *Bet S C C′*
        have *V1*: *R* ≠ *D*
          by (*simp add*: *P3*)
        then have *V2*: *Bet R D U*
        proof −
          have *W1*: *M Midpoint S R*
            using *P5 P6* **by** *blast*
          have *W2*: *M Midpoint C D*
            by (*simp add*: *P5 P6*)
          have *M Midpoint C′ U*
            by (*simp add*: *Q1 l7-2*)
          then show *?thesis*
            using *V0 P5 P6 l7-15* **by** *blast*
        qed
        have *Bet R D A*
          using *P1* **by** *auto*
        then have *Bet R U A* ∨ *Bet R A U*
          using *V1 V2 l5-1* **by** *blast*
      }
      then have *Q5*: *Bet S C C′* ⟶ *Bet R U A* ∨ *Bet R A U* **by** *simp*
      {
        assume *R1*: *Bet S C′ C*
        have *Bet R U A*
          using *P1 P5 P6 Q1 R1 between-exchange4 l7-15 l7-2* **by** *blast*
      }
      then have *Bet S C′ C* ⟶ *Bet R U A* ∨ *Bet R A U* **by** *simp*
      then have *Bet R U A* ∨ *Bet R A U*
        using *Q4 Q5* **by** *blast*
      then have *R Out U A*
        by (*simp add*: *Out-def Q2 Q3*)
    }
    then show *?thesis* **by** *simp*
  qed
  then show *?thesis*
    using *L1* **by** *blast*
qed

lemma *l9-4-1-aux*:
  assumes *S C Le R A* **and**
    P Q TS A C **and**
    Col R P Q **and**
    P Q Perp A R **and**
    Col S P Q **and**
    P Q Perp C S **and**
    M Midpoint R S
  shows ∀ *U C′*. (*M Midpoint U C′* ⟶ (*R Out U A* ⟷ *S Out C C′*))
  using *l9-4-1-aux-R1 l9-4-1-aux-R21 assms* **by** *smt*

lemma *per-col-eq*:
  assumes *Per A B C* **and**
    Col A B C **and**
    B ≠ *C*
  shows *A* = *B*
  using *assms*(*1*) *assms*(*2*) *assms*(*3*) *l8-9* **by** *blast*

lemma *l9-4-1*:
  assumes *P Q TS A C* **and**
    Col R P Q **and**
    P Q Perp A R **and**
    Col S P Q **and**
    P Q Perp C S **and**
    M Midpoint R S

shows ∀ *U C′*. *M Midpoint U C′* ⟶ (*R Out U A* ⟷ *S Out C C′*)
proof −
  have *P1*: *S C Le R A* ∨ *R A Le S C*
    using *local.le-cases* **by** *blast*
  {
    assume *Q1*: *S C Le R A*
    {
      fix *U C′*
      assume *M Midpoint U C′*
      then have (*R Out U A* ⟷ *S Out C C′*)
        using *Q1 assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *assms*(*6*) *l9-4-1-aux* **by** *blast*
    }
    then have ∀ *U C′*. *M Midpoint U C′* ⟶ (*R Out U A* ⟷ *S Out C C′*) **by** *simp*
  }
  then have *P2*: *S C Le R A* ⟶ (∀ *U C′*. *M Midpoint U C′* ⟶ (*R Out U A* ⟷ *S Out C C′*)) **by** *simp*
  {
    assume *Q2*: *R A Le S C*
    {
      fix *U C′*
      assume *M Midpoint U C′*
      then have (*R Out A U* ⟷ *S Out C′ C*)
        using *Q2 assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *assms*(*6*) *l7-2 l9-2 l9-4-1-aux* **by** *blast*
      then have (*R Out U A* ⟷ *S Out C C′*)
        using *l6-6* **by** *blast*
    }
    then have ∀ *U C′*. *M Midpoint U C′* ⟶ (*R Out U A* ⟷ *S Out C C′*) **by** *simp*
  }
  then have *P3*: *R A Le S C* ⟶ (∀ *U C′*. *M Midpoint U C′* ⟶ (*R Out U A* ⟷ *S Out C C′*)) **by** *simp*

  then show *?thesis*
    using *P1 P2* **by** *blast*
qed

lemma *mid-two-sides*:
  assumes *M Midpoint A B* **and**
    ¬ *Col A B X* **and**
    M Midpoint X Y
  shows *A B TS X Y*
proof −
  have *f1*: ¬ *Col Y A B*
    by (*meson Mid-cases Tarski-neutral-dimensionless.mid-preserves-col Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*2*) *assms*(*3*) *col-permutation-3*)
  have *Bet X M Y*
    using *assms*(*3*) *midpoint-bet* **by** *blast*
  then show *?thesis*
    using *f1* **by** (*metis* (*no-types*) *TS-def assms*(*1*) *assms*(*2*) *col-permutation-1 midpoint-col*)
qed

lemma *col-preserves-two-sides*:
  assumes *C* ≠ *D* **and**
    Col A B C **and**
    Col A B D **and**
    A B TS X Y
  shows *C D TS X Y*
proof −
  have *P1*: ¬ *Col X A B*
    using *TS-def assms*(*4*) **by** *blast*
  then have *P2*: *A* ≠ *B*
    using *not-col-distincts* **by** *blast*
  have *P3*: ¬ *Col X C D*
    by (*metis Col-cases P1 Tarski-neutral-dimensionless.colx Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*2*) *assms*(*3*))
  have *P4*: ¬ *Col Y C D*
    by (*metis Col-cases TS-def Tarski-neutral-dimensionless.colx Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*))

then show *?thesis*
proof −
obtain *pp* :: *'p ⇒ 'p ⇒ 'p ⇒ 'p ⇒ 'p* **where**
∀ *x0 x1 x2 x3*. (∃ *v4*. *Col v4 x3 x2* ∧ *Bet x1 v4 x0*) = (*Col* (*pp x0 x1 x2 x3*) *x3 x2* ∧ *Bet x1* (*pp x0 x1 x2 x3*) *x0*)
by *moura*
then have *f1*: ¬ *Col X A B* ∧ ¬ *Col Y A B* ∧ *Col* (*pp Y X B A*) *A B* ∧ *Bet X* (*pp Y X B A*) *Y*
using *TS-def assms*(*4*) **by** *presburger*
then have *Col* (*pp Y X B A*) *C D*
by (*meson P2 assms*(*2*) *assms*(*3*) *col3 not-col-permutation-3 not-col-permutation-4*)
then show *?thesis*
using *f1 TS-def P3 P4* **by** *blast*
qed
qed

lemma *out-out-two-sides*:
assumes *A* ≠ *B* **and**
A B TS X Y **and**
Col I A B **and**
Col I X Y **and**
I Out X U **and**
I Out Y V
shows *A B TS U V*
proof −
have *P0*: ¬ *Col X A B*
using *TS-def assms*(*2*) **by** *blast*
then have *P1*: ¬ *Col V A B*
by (*smt assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*6*) *col-out2-col col-transitivity-1 not-col-permutation-3 not-col-permutation-4 out-diff2 out-trivial ts-distincts*)
have *P2*: ¬ *Col U A B*
by (*metis P0 assms*(*3*) *assms*(*5*) *col-permutation-2 colx out-col out-distinct*)
obtain *T* **where** *P3*: *Col T A B* ∧ *Bet X T Y*
using *TS-def assms*(*2*) **by** *blast*
have *I* = *T*
proof −
have *f1*: ∀ *p pa pb*. ¬ *Col p pa pb* ∧ ¬ *Col p pb pa* ∧ ¬ *Col pa p pb* ∧ ¬ *Col pa pb p* ∧ ¬ *Col pb p pa* ∧ ¬ *Col pb pa p* ∨ *Col p pa pb*
using *Col-cases* **by** *blast*
then have *f2*: *Col X Y I*
using *assms*(*4*) **by** *blast*
have *f3*: *Col B A I*
using *f1 assms*(*3*) **by** *blast*
have *f4*: *Col B A T*
using *f1 P3* **by** *blast*
have *f5*: ¬ *Col X A B* ∧ ¬ *Col X B A* ∧ ¬ *Col A X B* ∧ ¬ *Col A B X* ∧ ¬ *Col B X A* ∧ ¬ *Col B A X*
using *f1* ‹¬ *Col X A B*› **by** *blast*
have *f6*: *A* ≠ *B* ∧ *A* ≠ *X* ∧ *A* ≠ *Y* ∧ *B* ≠ *X* ∧ *B* ≠ *Y* ∧ *X* ≠ *Y*
using *assms*(*2*) *ts-distincts* **by** *presburger*
have *Col X Y T*
using *f1* **by** (*meson P3 bet-col*)
then show *?thesis*
using *f6 f5 f4 f3 f2* **by** (*meson Tarski-neutral-dimensionless.l6-21 Tarski-neutral-dimensionless-axioms*)
qed
then have *Bet U T V*
using *P3 assms*(*5*) *assms*(*6*) *bet-out-out-bet* **by** *blast*
then show *?thesis*
using *P1 P2 P3 TS-def* **by** *blast*
qed

lemma *l9-4-2-aux-R1*:
assumes *R* = *S* **and**
S C Le R A **and**
P Q TS A C **and**
Col R P Q **and**
P Q Perp A R **and**
Col S P Q **and**
P Q Perp C S **and**

R Out U A **and**
S Out V C
shows *P Q TS U V*
proof −
have ¬ *Col A P Q*
using *TS-def assms(3)* **by** *auto*
then have *P2*: $P \neq Q$
using *not-col-distincts* **by** *blast*
obtain *T* **where** *P3*: *Col T P Q* ∧ *Bet A T C*
using *TS-def assms(3)* **by** *blast*
have $R = T$ **using** *assms(1) assms(5) assms(6) assms(7) col-permutation-1 l8-16-1 l8-6*
by (*meson P3*)
then show *?thesis*
by (*smt P2 P3 assms(1) assms(3) assms(8) assms(9) bet-col col-transitivity-2 l6-6 not-col-distincts out-out-two-sides*)
qed

lemma *l9-4-2-aux-R2*:
assumes $R \neq S$ **and**
S C Le R A **and**
P Q TS A C **and**
Col R P Q **and**
P Q Perp A R **and**
Col S P Q **and**
P Q Perp C S **and**
R Out U A **and**
S Out V C
shows *P Q TS U V*
proof −
have *P1*: $P \neq Q$
using *assms(7) perp-distinct* **by** *auto*
have *P2*: *R S TS A C*
using *assms(1) assms(3) assms(4) assms(6) col-permutation-1 col-preserves-two-sides* **by** *blast*
have *P3*: *Col R S P*
using *P1 assms(4) assms(6) col2--eq not-col-permutation-1* **by** *blast*
have *P4*: *Col R S Q*
by (*metis P3 Tarski-neutral-dimensionless.colx Tarski-neutral-dimensionless-axioms assms(4) assms(6) col-trivial-2*)
have *P5*: *R S Perp A R*
using *NCol-perm assms(1) assms(4) assms(5) assms(6) perp-col2* **by** *blast*
have *P6*: *R S Perp C S*
using *assms(1) assms(4) assms(6) assms(7) col-permutation-1 perp-col2* **by** *blast*
have *P7*: ¬ *Col A R S*
using *P2 TS-def* **by** *blast*
obtain *T* **where** *P8*: *Col T R S* ∧ *Bet A T C*
using *P2 TS-def* **by** *blast*
obtain *C′* **where** *P9*: *Bet R C′ A* ∧ *Cong S C R C′*
using *Le-def assms(2)* **by** *blast*
have ∃ *X*. *X Midpoint S R* ∧ *X Midpoint C C′*
proof −
have *Q1*: *C S Perp S R*
using *P6 Perp-perm* **by** *blast*
have *Q2*: *A R Perp S R*
using *P5 Perp-perm* **by** *blast*
have *Q3*: *Col S R T*
using *Col-cases P8* **by** *blast*
have *Q4*: *Bet C T A*
using *Bet-perm P8* **by** *blast*
have *Q5*: *Bet R C′ A*
by (*simp add*: *P9*)
have *Cong S C R C′*
by (*simp add*: *P9*)
then show *?thesis* **using** *Q1 Q2 Q3 Q4 Q5 l8-24*
by *blast*
qed
then obtain *M* **where** *P10*: *M Midpoint S R* ∧ *M Midpoint C C′* **by** *blast*
obtain *U′* **where** *P11*: *M Midpoint U U′*
using *symmetric-point-construction* **by** *blast*

```
  have P12: R ≠ U
    using assms(8) out-diff1 by blast
  have P13: R S TS U U'
   by (smt P10 P11 P12 P7 assms(8) col-transitivity-2 invert-two-sides mid-two-sides not-col-permutation-3 not-col-permutation-4
out-col)
  have P14: R S TS V U
  proof -
    have Q1: Col M R S
      using P10 midpoint-col not-col-permutation-5 by blast
    have Q2: M Midpoint U' U
      by (meson P11 Tarski-neutral-dimensionless.Mid-cases Tarski-neutral-dimensionless-axioms)
    have S Out U' V
       by (meson P10 P11 P2 P5 P6 Tarski-neutral-dimensionless.l7-2 Tarski-neutral-dimensionless-axioms assms(1)
assms(2) assms(8) assms(9) l6-6 l6-7 l9-4-1-aux-R21 not-col-distincts)
    then show ?thesis
      using P13 Q1 Q2 col-trivial-3 l9-2 l9-3 by blast
  qed
  then show ?thesis
    using P1 P3 P4 col-preserves-two-sides l9-2 by blast
qed

lemma l9-4-2-aux:
  assumes S C Le R A and
    P Q TS A C and
    Col R P Q and
    P Q Perp A R and
    Col S P Q and
    P Q Perp C S and
    R Out U A and
    S Out V C
  shows P Q TS U V
  using l9-4-2-aux-R1 l9-4-2-aux-R2
  by (metis assms(1) assms(2) assms(3) assms(4) assms(5) assms(6) assms(7) assms(8))

lemma l9-4-2:
  assumes P Q TS A C and
    Col R P Q and
    P Q Perp A R and
    Col S P Q and
    P Q Perp C S and
    R Out U A and
    S Out V C
  shows P Q TS U V
proof -
  have P1: S C Le R A ∨ R A Le S C
    by (simp add: local.le-cases)
  have P2: S C Le R A ⟶ P Q TS U V
    by (simp add: assms(1) assms(2) assms(3) assms(4) assms(5) assms(6) assms(7) l9-4-2-aux)
  have R A Le S C ⟶ P Q TS U V
    by (simp add: assms(1) assms(2) assms(3) assms(4) assms(5) assms(6) assms(7) l9-2 l9-4-2-aux)
  then show ?thesis
    using P1 P2 by blast
qed

lemma l9-5:
  assumes P Q TS A C and
    Col R P Q and
    R Out A B
  shows P Q TS B C
proof -
  have P1: P ≠ Q
    using assms(1) ts-distincts by blast
  obtain A' where P2: Col P Q A' ∧ P Q Perp A A'
   by (metis NCol-perm Tarski-neutral-dimensionless.TS-def Tarski-neutral-dimensionless-axioms assms(1) l8-18-existence)
  obtain C' where P3: Col P Q C' ∧ P Q Perp C C'
    using Col-perm TS-def assms(1) l8-18-existence by blast
```

```
  obtain M where P5: M Midpoint A' C'
    using midpoint-existence by blast
  obtain D where S2: M Midpoint A D
    using symmetric-point-construction by auto
  have ∃ B0. Col P Q B0 ∧ P Q Perp B B0
  proof -
    have S1: ¬ Col P Q B
     by (metis P2 Tarski-neutral-dimensionless.colx Tarski-neutral-dimensionless.perp-not-col2 Tarski-neutral-dimensionless-axioms
assms(2) assms(3) col-permutation-1 l6-3-1 out-col)
    then show ?thesis
      by (simp add: l8-18-existence)
  qed
  then obtain B' where P99: Col P Q B' ∧ P Q Perp B B' by blast
  have P Q TS B C
  proof -
    have S3: C' Out D C ⟷ A' Out A A
      using Out-cases P2 P3 P5 S2 assms(1) l9-4-1 not-col-permutation-1 by blast
    then have S4: C' Out D C
      using P2 Tarski-neutral-dimensionless.perp-not-eq-2 Tarski-neutral-dimensionless-axioms out-trivial by fastforce
    have S5: P Q TS A D
      using P2 P3 S3 S4 assms(1) col-permutation-2 l9-4-2 by blast
    {
      assume A' ≠ C'
      then have Col M P Q
        by (smt P2 P3 P5 col-trivial-2 l6-21 midpoint-col not-col-permutation-1)
      then have P Q TS B D
        using S2 S5 assms(2) assms(3) l9-3 by blast
    }
    then have A' ≠ C' ⟶ P Q TS B D by simp
    then have S6: P Q TS B D
      by (metis P3 P5 S2 S5 assms(2) assms(3) l9-3 midpoint-distinct-2 not-col-permutation-1)
    have S7: Col B' P Q
      using Col-perm P99 by blast
    have S8: P Q Perp B B'
      using P99 by blast
    have S9: Col C' P Q
      using Col-cases P3 by auto
    have S10: P Q Perp D C'
      by (metis Col-perm P3 S4 l6-3-1 out-col perp-col1 perp-right-comm)
    have S11: B' Out B B
      by (metis (no-types) P99 out-trivial perp-not-eq-2)
    have C' Out C D
      by (simp add: S4 l6-6)
    then show ?thesis using S6 S7 S8 S9 S10 S11 l9-4-2 by blast
  qed
  then show ?thesis using l8-18-existence by blast
qed

lemma outer-pasch-R1:
  assumes Col P Q C and
    Bet A C P and
    Bet B Q C
  shows ∃ X. Bet A X B ∧ Bet P Q X
  by (smt Bet-perm Col-def assms(1) assms(2) assms(3) between-exchange3 between-trivial outer-transitivity-between2)

lemma outer-pasch-R2:
  assumes ¬ Col P Q C and
    Bet A C P and
    Bet B Q C
  shows ∃ X. Bet A X B ∧ Bet P Q X
proof cases
  assume B = Q
  then show ?thesis
    using between-trivial by blast
next
  assume P1: B ≠ Q
```

have *P2*: $A \neq P$
using *assms*(*1*) *assms*(*2*) *between-identity col-trivial-3* **by** *blast*
have *P3*: $P \neq Q$
using *assms*(*1*) *col-trivial-1* **by** *blast*
have *P4*: $P \neq B$
using *assms*(*1*) *assms*(*3*) *bet-col* **by** *blast*
have *P5*: *P Q TS C B*
proof −
have *Q1*: ¬ *Col C P Q*
using *Col-cases assms*(*1*) **by** *blast*
have *Q2*: ¬ *Col B P Q*
by (*metis Col-cases P1 Tarski-neutral-dimensionless.colx Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*3*) *bet-col col-trivial-2*)
have ∃ *T*. *Col T P Q* ∧ *Bet C T B*
using *Col-cases assms*(*3*) *between-symmetry col-trivial-2* **by** *blast*
then show *?thesis*
by (*simp add*: *Q1 Q2 TS-def*)
qed
have *P6*: *P Q TS A B*
by (*metis P5 assms*(*1*) *assms*(*2*) *bet-out-1 l9-5 not-col-distincts*)
obtain *X* **where** *P7*: *Col X P Q* ∧ *Bet A X B*
using *P6 TS-def* **by** *blast*
have *Bet P Q X*
proof −
obtain *T* **where** *P8*: *Bet X T P* ∧ *Bet C T B*
using *P7 assms*(*2*) *between-symmetry inner-pasch* **by** *blast*
have *P9*: $B \neq C$
using *P1 assms*(*3*) *bet-neq12--neq* **by** *blast*
have *P10*: $T = Q$
proof −
have *f1*: ∀ *p pa pb*. *Col pb pa p* ∨ ¬ *Bet pb pa p*
by (*meson bet-col1 between-trivial*)
then have *f2*: *Col Q C B*
using *NCol-cases assms*(*3*) **by** *blast*
have *Col T C B*
using *f1 NCol-cases P8* **by** *blast*
then show *?thesis*
using *f2 f1* **by** (*metis* (*no-types*) *NCol-cases P7 P8 assms*(*1*) *between-trivial l6-16-1 l6-2 not-bet-and-out*)
qed
then show *?thesis*
using *P8 between-symmetry* **by** *blast*
qed
then show *?thesis* **using** *P7* **by** *blast*
qed

lemma *outer-pasch*:
assumes *Bet A C P* **and**
Bet B Q C
shows ∃ *X*. *Bet A X B* ∧ *Bet P Q X*
using *assms*(*1*) *assms*(*2*) *outer-pasch-R1 outer-pasch-R2* **by** *blast*

lemma *os-distincts*:
assumes *A B OS X Y*
shows $A \neq B \wedge A \neq X \wedge A \neq Y \wedge B \neq X \wedge B \neq Y$
using *OS-def assms ts-distincts* **by** *blast*

lemma *invert-one-side*:
assumes *A B OS P Q*
shows *B A OS P Q*
proof −
obtain *T* **where** *A B TS P T* ∧ *A B TS Q T*
using *OS-def assms* **by** *blast*
then have *B A TS P T* ∧ *B A TS Q T*
using *invert-two-sides* **by** *blast*
thus *?thesis*
using *OS-def* **by** *blast*

qed

lemma *l9-8-1*:
  assumes *P Q TS A C* **and**
    P Q TS B C
  shows *P Q OS A B*
proof −
  have ∃ *R*::′*p*. (*P Q TS A R* ∧ *P Q TS B R*)
    using *assms*(*1*) *assms*(*2*) **by** *blast*
  then show *?thesis*
    using *OS-def* **by** *blast*
qed

lemma *not-two-sides-id*:
  shows ¬ *P Q TS A A*
  using *ts-distincts* **by** *blast*

lemma *l9-8-2*:
  assumes *P Q TS A C* **and**
    P Q OS A B
  shows *P Q TS B C*
proof −
  obtain *D* **where** *P1*: *P Q TS A D* ∧ *P Q TS B D*
    using *assms*(*2*) *OS-def* **by** *blast*
  then have *P* ≠ *Q*
    using *ts-distincts* **by** *blast*
  obtain *T* **where** *P2*: *Col T P Q* ∧ *Bet A T C*
    using *TS-def assms*(*1*) **by** *blast*
  obtain *X* **where** *P3*: *Col X P Q* ∧ *Bet A X D*
    using *TS-def P1* **by** *blast*
  obtain *Y* **where** *P4*: *Col Y P Q* ∧ *Bet B Y D*
    using *TS-def P1* **by** *blast*
  then obtain *M* **where** *P5*: *Bet Y M A* ∧ *Bet X M B* **using** *P3 inner-pasch* **by** *blast*
  have *P6*: *A* ≠ *D*
    using *P1 ts-distincts* **by** *blast*
  have *P7*: *B* ≠ *D*
    using *P1 not-two-sides-id* **by** *blast*
  {
    assume *Q0*: *Col A B D*
    have *P Q TS B C*
    proof *cases*
      assume *Q1*: *M* = *Y*
      have *X* = *Y*
      proof −
        have *S1*: ¬ *Col P Q A*
          using *TS-def assms*(*1*) *not-col-permutation-1* **by** *blast*
        have *S3*: *Col P Q X*
          using *Col-perm P3* **by** *blast*
        have *S4*: *Col P Q Y*
          using *Col-perm P4* **by** *blast*
        have *S5*: *Col A D X*
          by (*simp add*: *P3 bet-col col-permutation-5*)
        have *Col A D Y*
          by (*metis Col-def P5 Q1 S5 Q0 between-equality between-trivial l6-16-1*)
        then show *?thesis* **using** *S1 S3 S4 S5 P6 l6-21*
          by *blast*
      qed
      then have *X Out A B*
        by (*metis P1 P3 P4 TS-def l6-2*)
      then show *?thesis* **using** *assms*(*1*) *P3 l9-5* **by** *blast*
    next
      assume *Z1*: ¬ *M* = *Y*
      have *X* = *Y*
      proof −
        have *S1*: ¬ *Col P Q A*
          using *TS-def assms*(*1*) *not-col-permutation-1* **by** *blast*

```
      have S3: Col P Q X
        using Col-perm P3 by blast
      have S4: Col P Q Y
        using Col-perm P4 by blast
      have S5: Col A D X
        by (simp add: P3 bet-col col-permutation-5)
      have Col A D Y
        by (metis Col-def P4 Q0 P7 l6-16-1)
      then show ?thesis using S1 S3 S4 S5 P6 l6-21
        by blast
    qed
    then have Z3: M ≠ X using Z1 by blast
    have Z4: P Q TS M C
      by (meson Out-cases P4 P5 Tarski-neutral-dimensionless.l9-5 Tarski-neutral-dimensionless-axioms Z1 assms(1)
bet-out)
    have X Out M B
      using P5 Z3 bet-out by auto
    then show ?thesis using Z4 P3 l9-5 by blast
  qed
 }
 then have Z99: Col A B D ⟶ P Q TS B C by blast
 {
  assume Q0: ¬ Col A B D
  have Q1: P Q TS M C
  proof −
    have S3: Y Out A M
    proof −
      have T1: A ≠ Y
        using Col-def P4 Q0 col-permutation-4 by blast
      have T2: M ≠ Y
      proof −
        {
          assume T3: M = Y
          have Col B D X
          proof −
            have U1: B ≠ M
              using P1 P4 T3 TS-def by blast
            have U2: Col B M D
              by (simp add: P4 T3 bet-col)
            have Col B M X
              by (simp add: P5 bet-col between-symmetry)
            then show ?thesis using U1 U2
              using col-transitivity-1 by blast
          qed
          have False
            by (metis NCol-cases P1 P3 TS-def ‹Col B D X› Q0 bet-col col-trivial-2 l6-21)
        }
        then show ?thesis by blast
      qed
      have Bet Y A M ∨ Bet Y M A using P5 by blast
      then show ?thesis using T1 T2
        by (simp add: Out-def)
    qed
    then have X Out M B
      by (metis P1 P3 P4 P5 TS-def bet-out l9-5)
    then show ?thesis using assms(1) S3 l9-5 P3 P4 by blast
  qed
  have X Out M B
    by (metis P3 P5 Q1 TS-def bet-out)
  then have P Q TS B C using Q1 P3 l9-5 by blast
 }
 then have ¬ Col A B D ⟶ P Q TS B C by blast
 then show ?thesis using Z99 by blast
qed

lemma l9-9:
```

assumes *P Q TS A B*
shows ¬ *P Q OS A B*
using *assms l9-8-2 not-two-sides-id* by *blast*

lemma *l9-9-bis*:
assumes *P Q OS A B*
shows ¬ *P Q TS A B*
using *assms l9-9* by *blast*

lemma *one-side-chara*:
assumes *P Q OS A B*
shows ∀ *X*. *Col X P Q* ⟶ ¬ *Bet A X B*
proof −
have ¬ *Col A P Q* ∧ ¬ *Col B P Q*
using *OS-def TS-def assms* by *auto*
then show *?thesis*
using *l9-9-bis TS-def assms* by *blast*
qed

lemma *l9-10*:
assumes ¬ *Col A P Q*
shows ∃ *C*. *P Q TS A C*
by (*meson Col-perm assms mid-two-sides midpoint-existence symmetric-point-construction*)

lemma *one-side-reflexivity*:
assumes ¬ *Col A P Q*
shows *P Q OS A A*
using *assms l9-10 l9-8-1* by *blast*

lemma *one-side-symmetry*:
assumes *P Q OS A B*
shows *P Q OS B A*
by (*meson Tarski-neutral-dimensionless.OS-def Tarski-neutral-dimensionless-axioms assms invert-two-sides*)

lemma *one-side-transitivity*:
assumes *P Q OS A B* and
P Q OS B C
shows *P Q OS A C*
by (*meson Tarski-neutral-dimensionless.OS-def Tarski-neutral-dimensionless.l9-8-2 Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*2*))

lemma *l9-17*:
assumes *P Q OS A C* and
Bet A B C
shows *P Q OS A B*
proof *cases*
assume *A* = *C*
then show *?thesis*
using *assms*(*1*) *assms*(*2*) *between-identity* by *blast*
next
assume *P1*: ¬ *A* = *C*
obtain *D* where *P2*: *P Q TS A D* ∧ *P Q TS C D*
using *OS-def assms*(*1*) by *blast*
then have *P3*: *P* ≠ *Q*
using *ts-distincts* by *blast*
obtain *X* where *P4*: *Col X P Q* ∧ *Bet A X D*
using *P2 TS-def* by *blast*
obtain *Y* where *P5*: *Col Y P Q* ∧ *Bet C Y D*
using *P2 TS-def* by *blast*
obtain *T* where *P6*: *Bet B T D* ∧ *Bet X T Y*
using *P4 P5 assms*(*2*) *l3-17* by *blast*
have *P7*: *P Q TS A D*
by (*simp add*: *P2*)
have *P Q TS B D*
proof −
have *Q1*: ¬ *Col B P Q*

using *assms*(*1*) *assms*(*2*) *one-side-chara* **by** *blast*
have *Q2*: ¬ *Col D P Q*
using *P2 TS-def* **by** *blast*
obtain *T0* **where** *Col T0 P Q* ∧ *Bet B T0 D*
proof −
assume *a1*: ⋀*T0*. *Col T0 P Q* ∧ *Bet B T0 D* ⟹ *thesis*
obtain *pp* :: ′*p* **where**
f2: *Bet B pp D* ∧ *Bet X pp Y*
using ‹⋀*thesis*. (⋀*T*. *Bet B T D* ∧ *Bet X T Y* ⟹ *thesis*) ⟹ *thesis*› **by** *blast*
have *Col P Q Y*
using *Col-def P5* **by** *blast*
then have *Y* = *X* ∨ *Col P Q pp*
using *f2 Col-def P4 colx* **by** *blast*
then show *?thesis*
using *f2 a1* **by** (*metis BetSEq BetS-def Col-def P4*)
qed
then show *?thesis* **using** *Q1 Q2*
using *TS-def* **by** *blast*
qed
then show *?thesis* **using** *P7*
using *OS-def* **by** *blast*
qed

lemma *l9-18-R1*:
assumes *Col X Y P* **and**
Col A B P
and *X Y TS A B*
shows *Bet A P B* ∧ ¬ *Col X Y A* ∧ ¬ *Col X Y B*
by (*meson TS-def assms*(*1*) *assms*(*2*) *assms*(*3*) *col-permutation-5 l9-5 not-col-permutation-1 not-out-bet not-two-sides-id*)

lemma *l9-18-R2*:
assumes *Col X Y P* **and**
Col A B P **and**
Bet A P B **and**
¬ *Col X Y A* **and**
¬ *Col X Y B*
shows *X Y TS A B*
using *Col-perm TS-def assms*(*1*) *assms*(*3*) *assms*(*4*) *assms*(*5*) **by** *blast*

lemma *l9-18*:
assumes *Col X Y P* **and**
Col A B P
shows *X Y TS A B* ⟷ (*Bet A P B* ∧ ¬ *Col X Y A* ∧ ¬ *Col X Y B*)
using *l9-18-R1 l9-18-R2 assms*(*1*) *assms*(*2*) **by** *blast*

lemma *l9-19-R1*:
assumes *Col X Y P* **and**
Col A B P **and**
X Y OS A B
shows *P Out A B* ∧ ¬ *Col X Y A*
by (*meson OS-def TS-def assms*(*1*) *assms*(*2*) *assms*(*3*) *col-permutation-5 not-col-permutation-1 not-out-bet one-side-chara*)

lemma *l9-19-R2*:
assumes *Col X Y P* **and**

P Out A B **and**
¬ *Col X Y A*
shows *X Y OS A B*
proof −
obtain *D* **where** *X Y TS A D*
using *Col-perm assms*(*3*) *l9-10* **by** *blast*
then show *?thesis*
using *OS-def assms*(*1*) *assms*(*2*) *l9-5 not-col-permutation-1* **by** *blast*
qed

lemma *l9-19*:

assumes *Col X Y P* **and**
Col A B P
shows *X Y OS A B* ⟷ (*P Out A B* ∧ ¬ *Col X Y A*)
using *l9-19-R1 l9-19-R2 assms*(*1*) *assms*(*2*) **by** *blast*

lemma *one-side-not-col123*:
assumes *A B OS X Y*
shows ¬ *Col A B X*
using *assms col-trivial-3 l9-19* **by** *blast*

lemma *one-side-not-col124*:
assumes *A B OS X Y*
shows ¬ *Col A B Y*
using *assms one-side-not-col123 one-side-symmetry* **by** *blast*

lemma *col-two-sides*:
assumes *Col A B C* **and**
A ≠ *C* **and**
A B TS P Q
shows *A C TS P Q*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *col-preserves-two-sides col-trivial-3* **by** *blast*

lemma *col-one-side*:
assumes *Col A B C* **and**
A ≠ *C* **and**
A B OS P Q
shows *A C OS P Q*
proof −
obtain *T* **where** *A B TS P T* ∧ *A B TS Q T* **using** *assms*(*1*) *assms*(*2*) *assms*(*3*) *OS-def* **by** *blast*
then show *?thesis*
using *col-two-sides OS-def assms*(*1*) *assms*(*2*) **by** *blast*
qed

lemma *out-out-one-side*:
assumes *A B OS X Y* **and**
A Out Y Z
shows *A B OS X Z*
by (*meson Col-cases Tarski-neutral-dimensionless.OS-def Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*2*) *col-trivial-3 l9-5*)

lemma *out-one-side*:
assumes ¬ *Col A B X* ∨ ¬ *Col A B Y* **and**
A Out X Y
shows *A B OS X Y*
using *assms*(*1*) *assms*(*2*) *l6-6 not-col-permutation-2 one-side-reflexivity one-side-symmetry out-out-one-side* **by** *blast*

lemma *bet--ts*:
assumes *A* ≠ *Y* **and**
¬ *Col A B X* **and**
Bet X A Y
shows *A B TS X Y*
proof −
have ¬ *Col Y A B*
using *NCol-cases assms*(*1*) *assms*(*2*) *assms*(*3*) *bet-col col2--eq* **by** *blast*
then show *?thesis*
by (*meson TS-def assms*(*2*) *assms*(*3*) *col-permutation-3 col-permutation-5 col-trivial-3*)
qed

lemma *bet-ts--ts*:
assumes *A B TS X Y* **and**
Bet X Y Z
shows *A B TS X Z*
proof −
have ¬ *Col Z A B*
using *assms*(*1*) *assms*(*2*) *bet-col between-equality-2 col-permutation-1 l9-18* **by** *blast*

then show *?thesis*
using *TS-def assms(1) assms(2) between-exchange4* **by** *blast*
qed

lemma *bet-ts--os*:
assumes *A B TS X Y* **and**
Bet X Y Z
shows *A B OS Y Z*
using *OS-def assms(1) assms(2) bet-ts--ts l9-2* **by** *blast*

lemma *l9-31* :
assumes *A X OS Y Z* **and**
A Z OS Y X
shows *A Y TS X Z*
proof −
have *P1*: $A \neq X \wedge A \neq Z \wedge \neg$ *Col Y A X* $\wedge \neg$ *Col Z A X* $\wedge \neg$ *Col Y A Z*
using *assms(1) assms(2) col-permutation-1 one-side-not-col123 one-side-not-col124 os-distincts* **by** *blast*
obtain *Z′* **where** *P2*: *Bet Z A Z′* $\wedge$ *Cong A Z′ Z A*
using *segment-construction* **by** *blast*
have *P3*: $Z' \neq A$
using *P1 P2 cong-diff-4* **by** *blast*
have *P4*: *A X TS Y Z′*
by (*metis* (*no-types*) *P2 P3 assms(1) bet--ts l9-8-2 one-side-not-col124 one-side-symmetry*)
have *P5*: ¬ *Col Y A X*
using *P1* **by** *blast*
obtain *T* **where** *P6*: *Col A T X* $\wedge$ *Bet Y T Z′*
using *P4 TS-def not-col-permutation-4* **by** *blast*
then have *P7*: $T \neq A$
proof −
have ¬ *Col A Z Y*
by (*simp add*: *P1 not-col-permutation-1*)
then have *f1*: ¬ *A Out Z Y*
using *out-col* **by** *blast*
have $A \neq Z'$
using *P1 P2 cong-diff-4* **by** *blast*
then show *?thesis*
using *f1* **by** (*metis* (*no-types*) *P1 P2 P6 l6-2*)
qed
have *P8*: *Y A OS Z′ T*
by (*smt P1 P2 P3 P6 Tarski-neutral-dimensionless.l6-6 Tarski-neutral-dimensionless-axioms bet-col bet-out col-trivial-2 l6-21 not-col-permutation-1 out-one-side*)
have *P9*: *A Y TS Z′ Z*
using *Col-perm P1 P2 P8 bet--ts between-symmetry one-side-not-col123* **by** *blast*
{
assume *Q0*: *Bet T A X*
have *Q1*: *Z′ Z OS Y T*
by (*metis BetSEq BetS-def P1 P2 P4 P6 TS-def Tarski-neutral-dimensionless.l6-6 Tarski-neutral-dimensionless-axioms bet-col bet-out-1 col-trivial-3 colx not-col-permutation-3 not-col-permutation-4 out-one-side*)
then have *Q2*: *Z′ Out T Y*
by (*metis P6 bet-out-1 os-distincts*)
then have *Q3*: *A Z OS Y T*
by (*meson Out-cases P1 P2 P6 bet-col col-permutation-3 invert-one-side l9-19-R2*)
have *A Z TS X T*
proof −
have *R1*: ¬ *Col X A Z*
using *P1 col-permutation-3* **by** *blast*
have *R2*: ¬ *Col T A Z*
using *Q3 between-trivial one-side-chara* **by** *blast*
have ∃ *T0*. *Col T0 A Z* $\wedge$ *Bet X T0 T*
proof −
have *S1*: *Col A A Z*
by (*simp add*: *col-trivial-1*)
have *Bet X A T*
by (*simp add*: *Q0 between-symmetry*)
then show *?thesis* **using** *S1* **by** *blast*
qed

```
      then show ?thesis using R1 R2
        using TS-def by auto
    qed
    have A Y TS X Z
     by (meson Q3 Tarski-neutral-dimensionless.l9-8-2 Tarski-neutral-dimensionless.one-side-symmetry Tarski-neutral-dimensionless-ax
‹A Z TS X T› assms(2) l9-9-bis)
  }
  then have P10: Bet T A X ⟶ A Y TS X Z by blast
  {
    assume R1: Bet A X T
    then have R3: A Y OS Z′ X
     by (meson Bet-cases P1 P6 P8 R1 between-equality invert-one-side not-col-permutation-4 not-out-bet out-out-one-side)
    have A Y TS X Z
      using R3 P9 l9-8-2 by blast
  }
  then have P11: Bet A X T ⟶ A Y TS X Z by blast
  {
    assume R1: Bet X T A
    then have R3: A Y OS T X
      by (simp add: P5 P7 R1 bet-out-1 not-col-permutation-4 out-one-side)
    then have A Y TS X Z
      using P8 P9 invert-two-sides l9-8-2 by blast
  }
  then have Bet X T A ⟶ A Y TS X Z by blast
  then show ?thesis using P10 P11
    using P6 between-symmetry third-point by blast
qed

lemma col123--nos:
  assumes Col P Q A
  shows ¬ P Q OS A B
  using assms one-side-not-col123 by blast

lemma col124--nos:
  assumes Col P Q B
  shows ¬ P Q OS A B
  using assms one-side-not-col124 by blast

lemma col2-os--os:
  assumes C ≠ D and
    Col A B C and
    Col A B D and
    A B OS X Y
  shows C D OS X Y
  by (metis assms(1) assms(2) assms(3) assms(4) col3 col-one-side col-trivial-3 invert-one-side os-distincts)

lemma os-out-os:
  assumes Col A B P and
    A B OS C D and
    P Out C C′
  shows A B OS C′ D
  using OS-def assms(1) assms(2) assms(3) l9-5 not-col-permutation-1 by blast

lemma ts-ts-os:
  assumes A B TS C D and
    C D TS A B
  shows A C OS B D
proof −
  obtain T1 where P1: Col T1 A B ∧ Bet C T1 D
    using TS-def assms(1) by blast
  obtain T where P2: Col T C D ∧ Bet A T B
    using TS-def assms(2) by blast
  have P3: T1 = T
  proof −
    have A ≠ B
      using assms(2) ts-distincts by blast
```

```
    then show ?thesis
    proof -
      have Col T1 D C
        using Col-def P1 by blast
      then have f1: ∀ p. (C = T1 ∨ Col C p T1) ∨ ¬ Col C T1 p
        by (metis assms(1) col-transitivity-1 l6-16-1 ts-distincts)
      have f2: ¬ Col C A B
        using TS-def assms(1) by presburger
      have f3: (Bet B T1 A ∨ Bet T1 A B) ∨ Bet A B T1
        using Col-def P1 by blast
      {
        assume T1 ≠ B
        then have C ≠ T1 ∧ ¬ Col C T1 B ∨ (∃ p. ¬ Col p T1 B ∧ Col p T1 T) ∨ T ≠ A ∧ T ≠ B
          using f3 f2 by (metis (no-types) Col-def col-transitivity-1 l6-16-1)
        then have T ≠ A ∧ T ≠ B ∨ C ≠ T1 ∧ ¬ Col C T1 T ∨ T1 = T
          using f3 by (meson Col-def l6-16-1)
      }
      moreover
      {
        assume T ≠ A ∧ T ≠ B
        then have C ≠ T1 ∧ ¬ Col C T1 T ∨ T1 = T
          using f2 by (metis (no-types) Col-def P1 P2 ‹A ≠ B› col-transitivity-1 l6-16-1)
      }
      ultimately have C ≠ T1 ∧ ¬ Col C T1 T ∨ T1 = T
        using f2 f1 assms(1) ts-distincts by blast
      then show ?thesis
        by (metis (no-types) Col-def P1 P2 assms(1) l6-16-1 ts-distincts)
    qed
  qed
  have P4: A C OS T B
    by (metis Col-cases P2 TS-def assms(1) assms(2) bet-out out-one-side)
  then have C A OS T D
    by (metis Col-cases P1 TS-def P3 assms(2) bet-out os-distincts out-one-side)
  then show ?thesis
   by (meson P4 Tarski-neutral-dimensionless.invert-one-side Tarski-neutral-dimensionless.one-side-symmetry Tarski-neutral-dimensio
one-side-transitivity)
qed

lemma col-one-side-out:
  assumes Col A X Y and
    A B OS X Y
  shows A Out X Y
  by (meson assms(1) assms(2) l6-4-2 not-col-distincts not-col-permutation-4 one-side-chara)

lemma col-two-sides-bet:
  assumes Col A X Y and
    A B TS X Y
  shows Bet X A Y
  using Col-cases assms(1) assms(2) l9-8-1 l9-9 or-bet-out out-out-one-side by blast

lemma os-ts1324--os:
  assumes A X OS Y Z and
    A Y TS X Z
  shows A Z OS X Y
proof -
  obtain P where P1: Col P A Y ∧ Bet X P Z
    using TS-def assms(2) by blast
  have P2: A Z OS X P
    by (metis Col-cases P1 TS-def assms(1) assms(2) bet-col bet-out-1 col124--nos col-trivial-2 l6-6 l9-19)
  have A Z OS P Y
  proof -
    have ¬ Col A Z P ∨ ¬ Col A Z Y
      using P2 col124--nos by blast
    moreover have A Out P Y
    proof -
      have X A OS P Z
```

by (*metis Col-cases P1 P2 assms*(*1*) *bet-out col123--nos out-one-side*)
then have *A X OS P Y*
by (*meson Tarski-neutral-dimensionless.invert-one-side Tarski-neutral-dimensionless.one-side-symmetry Tarski-neutral-dimension*
assms(*1*) *one-side-transitivity*)
then show *?thesis*
using *P1 col-one-side-out not-col-permutation-4* **by** *blast*
qed
ultimately show *?thesis*
by (*simp add*: *out-one-side*)
qed
then show *?thesis*
using *P2 one-side-transitivity* **by** *blast*
qed

lemma *ts2--ex-bet2*:
assumes *A C TS B D* **and**
B D TS A C
shows ∃ *X*. *Bet A X C* ∧ *Bet B X D*
by (*metis TS-def assms*(*1*) *assms*(*2*) *bet-col col-permutation-5 l9-18-R1 not-col-permutation-2*)

lemma *out-one-side-1*:
assumes ¬ *Col A B C* **and**
Col A B X **and**
X Out C D
shows *A B OS C D*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *not-col-permutation-2 one-side-reflexivity one-side-symmetry os-out-os* **by** *blast*

lemma *out-two-sides-two-sides*:
assumes
Col A B PX **and**
PX Out X P **and**
A B TS P Y
shows *A B TS X Y*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *l6-6 l9-5 not-col-permutation-1* **by** *blast*

lemma *l8-21-bis*:
assumes $X \neq Y$ **and**
¬ *Col C A B*
shows ∃ *P*. *Cong A P X Y* ∧ *A B Perp P A* ∧ *A B TS C P*
proof −
have *P1*: $A \neq B$
using *assms*(*2*) *not-col-distincts* **by** *blast*
then have ∃ *P T*. *A B Perp P A* ∧ *Col A B T* ∧ *Bet C T P*
using *l8-21* **by** *auto*
then obtain *P T* **where** *P2*: *A B Perp P A* ∧ *Col A B T* ∧ *Bet C T P* **by** *blast*
have *P3*: *A B TS C P*
proof −
have ¬ *Col P A B*
using *P2 col-permutation-1 perp-not-col* **by** *blast*
then show *?thesis*
using *P2 TS-def assms*(*2*) *not-col-permutation-1* **by** *blast*
qed
have *P4*: $P \neq A$
using *P3 ts-distincts* **by** *blast*
obtain *P′* **where** *P5*: (*Bet A P P′* ∨ *Bet A P′ P*) ∧ *Cong A P′ X Y*
using *segment-construction-2 P4* **by** *blast*
have *P6*: *A B Perp P′ A*
by (*smt P2 P5 Perp-perm assms*(*1*) *bet-col cong-identity cong-symmetry not-bet-distincts not-col-permutation-2*
perp-col2)
have *P7*: ¬ *Col P′ A B*
using *NCol-perm P6 col-trivial-3 l8-16-1* **by** *blast*
then have *P8*: *A B OS P P′*
by (*metis Out-def P4 P5 P6 col-permutation-2 out-one-side perp-not-eq-2*)
then have *P9*: *A B TS C P′*
using *P3 l9-2 l9-8-2* **by** *blast*
then show *?thesis*

```
    using P5 P6 by blast
qed

lemma ts--ncol:
  assumes A B TS X Y
  shows  ¬ Col A X Y ∨ ¬ Col B X Y
  by (metis TS-def assms col-permutation-1 col-transitivity-2 ts-distincts)

lemma one-or-two-sides-aux:
  assumes ¬ Col C A B and
    ¬ Col D A B and
    Col A C X
    and Col B D X
  shows A B TS C D ∨ A B OS C D
proof -
  have P1: A ≠ X
    using assms(2) assms(4) col-permutation-2 by blast
  have P2: B ≠ X
    using assms(1) assms(3) col-permutation-4 by blast
  have P3: ¬ Col X A B
    using P1 assms(1) assms(3) col-permutation-5 col-transitivity-1 not-col-permutation-4 by blast
  {
    assume Q0: Bet A C X ∧ Bet B D X
    then have Q1: A B OS C X
      using assms(1) bet-out not-col-distincts not-col-permutation-1 out-one-side by blast
    then have A B OS X D
     by (metis Q0 assms(2) assms(4) bet-out-1 col-permutation-2 col-permutation-3 invert-one-side l6-4-2 not-bet-and-out
not-col-distincts out-one-side)
    then have A B OS C D
      using Q1 one-side-transitivity by blast
  }
  then have P4: Bet A C X ∧ Bet B D X ⟶ A B OS C D by blast
  {
    assume Bet A C X ∧ Bet D X B
    then have A B OS C D
        by (smt P2 assms(1) assms(4) bet-out between-equality-2 l9-10 l9-5 l9-8-1 not-bet-and-out not-col-distincts
not-col-permutation-4 out-to-bet out-two-sides-two-sides)
  }
  then have P5: Bet A C X ∧ Bet D X B ⟶ A B OS C D  by blast
  {
    assume Q0: Bet A C X ∧ Bet X B D
    have Q1: A B TS X D
      using P3 Q0 TS-def assms(2) col-trivial-3 by blast
    have A B OS X C
      using Q0 assms(1) bet-out not-col-distincts one-side-reflexivity one-side-symmetry out-out-one-side by blast
    then have A B TS C D
      using Q1 l9-8-2 by blast
  }
  then have P6: Bet A C X ∧ Bet X B D ⟶ A B TS C D by blast
  {
    assume Q1: Bet C X A ∧ Bet B D X
    then have Q2: A B OS C X
     using P1 assms(1) assms(3) between-equality-2 l6-4-2 not-col-permutation-1 not-col-permutation-4 out-one-side by
blast
    have A B OS X D
      using Q1 assms(2) bet-out not-col-distincts one-side-reflexivity os-out-os by blast
    then have A B OS C D using Q2
      using one-side-transitivity by blast
  }
  then have P7: Bet C X A ∧ Bet B D X ⟶ A B OS C D by blast
  {
    assume Bet C X A ∧ Bet D X B
    then have A B OS C D
      by (smt ‹Bet A C X ∧ Bet D X B ⟹ A B OS C D› ‹Bet C X A ∧ Bet B D X ⟹ A B OS C D› assms(1)
assms(2) assms(3) assms(4) between-symmetry l6-21 l9-18-R2 not-col-distincts ts-ts-os)
  }
```

then have *P8*: *Bet C X A* ∧ *Bet D X B* ⟶ *A B OS C D* **by** *blast*
{
assume *Q1*: *Bet C X A* ∧ *Bet X B D*
have *Q2*: *A B TS X D*
by (*metis P3 Q1 assms*(*2*) *bet--ts invert-two-sides not-col-distincts not-col-permutation-3*)
have *Q3*: *A B OS X C*
using *P1 Q1 assms*(*1*) *bet-out-1 not-col-permutation-1 out-one-side* **by** *auto*
then have *A B TS C D*
using *Q2 l9-8-2* **by** *blast*
}
then have *P9*: *Bet C X A* ∧ *Bet X B D* ⟶ *A B TS C D* **by** *blast*
{
assume *Q0*: *Bet X A C* ∧ *Bet B D X*
have *Q1*: *A B TS X C*
by (*metis P3 Q0 assms*(*1*) *bet--ts col-permutation-2 not-col-distincts*)
have *A B OS X D*
by (*metis NCol-cases Q0 Tarski-neutral-dimensionless.out-one-side Tarski-neutral-dimensionless-axioms assms*(*2*) *assms*(*4*) *bet-out-1 invert-one-side l6-4-1 not-col-distincts not-out-bet*)
then have *A B TS C D*
using *Q1 l9-2 l9-8-2* **by** *blast*
}
then have *P10*: *Bet X A C* ∧ *Bet B D X* ⟶ *A B TS C D* **by** *blast*
{
assume *Q0*: *Bet X A C* ∧ *Bet D X B*
have *Q1*: *A B TS X C*
by (*metis NCol-cases P3 Q0 assms*(*1*) *bet--ts not-col-distincts*)
have *A B OS X D*
by (*metis P2 P3 Q0 bet-out-1 col-permutation-3 invert-one-side out-one-side*)
then have *A B TS C D*
using *Q1 l9-2 l9-8-2* **by** *blast*
}
then have *P11*: *Bet X A C* ∧ *Bet D X B* ⟶ *A B TS C D*
by *blast*
{
assume *Q0*: *Bet X A C* ∧ *Bet X B D*
then have *Q1*: *A B TS C X*
by (*simp add*: *P1 Q0 assms*(*1*) *bet--ts between-symmetry not-col-permutation-1*)
have *A B TS D X*
by (*simp add*: *P2 Q0 assms*(*2*) *bet--ts between-symmetry invert-two-sides not-col-permutation-3*)
then have *A B OS C D*
using *Q1 l9-8-1* **by** *blast*
}
then have *P12*: *Bet X A C* ∧ *Bet X B D* ⟶ *A B OS C D* **by** *blast*
then show *?thesis* **using** *P4 P5 P6 P7 P8 P9 P10 P11*
using *Col-def assms*(*3*) *assms*(*4*) **by** *auto*
qed

lemma *cop--one-or-two-sides*:
assumes *Coplanar A B C D* **and**
¬ *Col C A B* **and**
¬ *Col D A B*
shows *A B TS C D* ∨ *A B OS C D*
proof −
obtain *X* **where** *P1*: *Col A B X* ∧ *Col C D X* ∨ *Col A C X* ∧ *Col B D X* ∨ *Col A D X* ∧ *Col B C X*
using *Coplanar-def assms*(*1*) **by** *auto*
have *P2*: *Col A B X* ∧ *Col C D X* ⟶ *A B TS C D* ∨ *A B OS C D*
by (*metis TS-def Tarski-neutral-dimensionless.l9-19-R2 Tarski-neutral-dimensionless-axioms assms*(*2*) *assms*(*3*) *not-col-permutation-3 not-col-permutation-5 not-out-bet*)
have *P3*: *Col A C X* ∧ *Col B D X* ⟶ *A B TS C D* ∨ *A B OS C D*
using *assms*(*2*) *assms*(*3*) *one-or-two-sides-aux* **by** *blast*
have *Col A D X* ∧ *Col B C X* ⟶ *A B TS C D* ∨ *A B OS C D*
using *assms*(*2*) *assms*(*3*) *l9-2 one-or-two-sides-aux one-side-symmetry* **by** *blast*
then show *?thesis*
using *P1 P2 P3* **by** *blast*
qed

lemma *os--coplanar*:
  assumes *A B OS C D*
  shows *Coplanar A B C D*
proof −
  have *P1*: ¬ *Col A B C*
    using *assms one-side-not-col123* **by** *blast*
  obtain *C'* **where** *P2*: *Bet C B C'* ∧ *Cong B C' B C*
    using *segment-construction* **by** *presburger*
  have *P3*: *A B TS D C'*
    by (*metis* (*no-types*) *Cong-perm OS-def P2 TS-def assms bet--ts bet-cong-eq invert-one-side l9-10 l9-8-2 one-side-not-col123 ts-distincts*)
  obtain *T* **where** *P4*: *Col T A B* ∧ *Bet D T C'*
    using *P3 TS-def* **by** *blast*
  have *P5*: *C'* ≠ *T*
    using *P3 P4 TS-def* **by** *blast*
  have *P6*: *Col T B C* ⟶ *Coplanar A B C D*
    by (*metis Col-def Coplanar-def P2 P4 P5 col-trivial-2 l6-16-1*)
  {
    assume *Q0*: ¬ *Col T B C*
    {
      assume *R0*: *Bet T B A*
      have *S1*: *B C TS T A*
        by (*metis P1 Q0 R0 bet--ts col-permutation-2 not-col-distincts*)
      have *C' Out T D*
        using *P4 P5 bet-out-1* **by** *auto*
      then have *B C OS T D*
        using *P2 Q0 bet-col invert-one-side not-col-permutation-3 out-one-side-1* **by** *blast*
      then have *R1*: *B C TS D A*
        using *S1 l9-8-2* **by** *blast*
      then have *Coplanar A B C D*
        using *ncoplanar-perm-9 ts--coplanar* **by** *blast*
    }
    then have *Q1*: *Bet T B A* ⟶ *Coplanar A B C D* **by** *blast*
    {
      assume *R0*: ¬ *Bet T B A*
      {
        have *R2*: *B C OS D T*
        proof −
          have *S1*: ¬ *Col B C D*
            by (*metis Col-perm P2 P3 P4 Q0 bet-col colx ts-distincts*)
          have *S2*: *Col B C C'*
            by (*simp add*: *P2 bet-col col-permutation-4*)
          have *S3*: *C' Out D T*
            using *P4 P5 bet-out-1 l6-6* **by** *auto*
          then show *?thesis*
            using *S1 S2 out-one-side-1* **by** *blast*
        qed

        have *R3*: *B C OS T A*
          using *P4 Q0 R0 col-permutation-2 col-permutation-5 not-bet-out out-one-side* **by** *blast*
      }
      then have *R1*: *B C OS D A*
        by (*metis P2 P4 Q0 bet-col bet-out-1 col-permutation-2 col-permutation-5 os-out-os*)
      then have *Coplanar A B C D*
        by (*simp add*: *R1 assms coplanar-perm-19 invert-one-side l9-31 one-side-symmetry ts--coplanar*)
    }
    then have ¬ *Bet T B A* ⟶ *Coplanar A B C D* **by** *blast*
    then have *Coplanar A B C D* **using** *Q1* **by** *blast*
  }
  then have ¬ *Col T B C* ⟶ *Coplanar A B C D* **by** *blast*
  then show *?thesis* **using** *P6* **by** *blast*
qed

lemma *coplanar-trans-1*:
  assumes ¬ *Col P Q R* **and**
    Coplanar P Q R A **and**

Coplanar P Q R B
shows *Coplanar Q R A B*
proof −
have *P1*: *Col Q R A* ⟶ *Coplanar Q R A B*
by (*simp add*: *col--coplanar*)
{
assume *T1*: ¬ *Col Q R A*
{
assume *T2*: ¬ *Col Q R B*
{
have *Col Q A B* ⟶ *Coplanar Q R A B*
using *ncop--ncols* **by** *blast*
{
assume *S1*: ¬ *Col Q A B*
have *U1*: *Q R TS P A* ∨ *Q R OS P A*
by (*simp add*: *T1 assms*(*1*) *assms*(*2*) *cop--one-or-two-sides coplanar-perm-8 not-col-permutation-2*)
have *U2*: *Q R TS P B* ∨ *Q R OS P B*
using *T2 assms*(*1*) *assms*(*3*) *col-permutation-1 cop--one-or-two-sides coplanar-perm-8* **by** *blast*
have *W1*: *Q R TS P A* ∧ *Q R OS P A* ⟶ *Q R TS A B* ∨ *Q R OS A B*
using *l9-9* **by** *blast*
have *W2*: *Q R TS P A* ∧ *Q R OS P B* ⟶ *Q R TS A B* ∨ *Q R OS A B*
using *l9-2 l9-8-2* **by** *blast*
have *W3*: *Q R TS P B* ∧ *Q R OS P A* ⟶ *Q R TS A B* ∨ *Q R OS A B*
using *l9-8-2* **by** *blast*
have *Q R TS P B* ∧ *Q R OS P B* ⟶ *Q R TS A B* ∨ *Q R OS A B*
using *l9-9* **by** *blast*
then have *S2*: *Q R TS A B* ∨ *Q R OS A B* **using** *U1 U2 W1 W2 W3*
using *OS-def l9-2 one-side-transitivity* **by** *blast*
have *Coplanar Q R A B*
using *S2 os--coplanar ts--coplanar* **by** *blast*
}
then have ¬ *Col Q A B* ⟶ *Coplanar Q R A B* **by** *blast*
}
then have *Coplanar Q R A B*
using *ncop--ncols* **by** *blast*
}
then have ¬ *Col Q R B* ⟶ *Coplanar Q R A B*
by *blast*
}
then have ¬ *Col Q R A* ⟶ *Coplanar Q R A B*
using *ncop--ncols* **by** *blast*
then show *?thesis* **using** *P1* **by** *blast*
qed

lemma *col-cop--cop*:
assumes *Coplanar A B C D* **and**
$C \neq D$ **and**
Col C D E
shows *Coplanar A B C E*
proof −
have *Col D A C* ⟶ *Coplanar A B C E*
by (*meson assms*(*2*) *assms*(*3*) *col-permutation-1 l6-16-1 ncop--ncols*)
moreover
{
assume ¬ *Col D A C*
then have *Coplanar A C B E*
by (*meson assms*(*1*) *assms*(*3*) *col--coplanar coplanar-trans-1 ncoplanar-perm-11 ncoplanar-perm-13*)
then have *Coplanar A B C E*
using *ncoplanar-perm-2* **by** *blast*
}
ultimately show *?thesis*
by *blast*
qed

lemma *bet-cop--cop*:
assumes *Coplanar A B C E* **and**

Bet C D E
shows *Coplanar A B C D*
by (*metis NCol-perm Tarski-neutral-dimensionless.col-cop--cop Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*2*) *bet-col bet-neq12--neq*)

lemma *col2-cop--cop*:
assumes *Coplanar A B C D* **and**
C ≠ D **and**
Col C D E **and**
Col C D F
shows *Coplanar A B E F*
proof *cases*
assume *C = E*
then show *?thesis*
using *assms*(*1*) *assms*(*2*) *assms*(*4*) *col-cop--cop* **by** *blast*
next
assume *C ≠ E*
then show *?thesis*
by (*metis assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *col-cop--cop col-transitivity-1 ncoplanar-perm-1 not-col-permutation-4*)
qed

lemma *col-cop2--cop*:
assumes *U ≠ V* **and**
Coplanar A B C U **and**
Coplanar A B C V **and**
Col U V P
shows *Coplanar A B C P*
proof *cases*
assume *Col A B C*
then show *?thesis*
using *ncop--ncol* **by** *blast*
next
assume ¬ *Col A B C*
then show *?thesis*
by (*smt Col-perm assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *col-cop--cop coplanar-trans-1 ncoplanar-perm-1 ncoplanar-perm-14 ncoplanar-perm-15 ncoplanar-perm-23*)
qed

lemma *bet-cop2--cop*:
assumes *Coplanar A B C U* **and**
Coplanar A B C W **and**
Bet U V W
shows *Coplanar A B C V*
proof −
have *Col U V W*
using *assms*(*3*) *bet-col* **by** *blast*
then have *Col U W V*
by (*meson not-col-permutation-5*)
then show *?thesis*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *bet-neq23--neq col-cop2--cop* **by** *blast*
qed

lemma *coplanar-pseudo-trans*:
assumes ¬ *Col P Q R* **and**
Coplanar P Q R A **and**
Coplanar P Q R B **and**
Coplanar P Q R C **and**
Coplanar P Q R D
shows *Coplanar A B C D*
proof *cases*
have *LEM1*: (¬ *Col P Q R* ∧ *Coplanar P Q R A* ∧ *Coplanar P Q R B* ∧ *Coplanar P Q R C*) ⟶ *Coplanar A B C R*
by (*smt col-transitivity-2 coplanar-trans-1 ncop--ncols ncoplanar-perm-19 ncoplanar-perm-21*)
assume *P2*: *Col P Q D*
have *P3*: *P ≠ Q*
using *assms*(*1*) *col-trivial-1* **by** *blast*
have *P4*: *Coplanar A B C Q*

by (*smt assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *col2-cop--cop coplanar-trans-1 ncoplanar-perm-9 not-col-distincts*)
have *P5*: ¬ *Col Q R P*
using *Col-cases assms*(*1*) **by** *blast*
have *P6*: *Coplanar Q R P A*
using *assms*(*2*) *ncoplanar-perm-12* **by** *blast*
have *P7*: *Coplanar Q R P B*
using *assms*(*3*) *ncoplanar-perm-12* **by** *blast*
have *P8*: *Coplanar Q R P C*
using *assms*(*4*) *ncoplanar-perm-12* **by** *blast*
then have *Coplanar A B C P* **using** *LEM1 P5 P6 P7*
by (*smt col-transitivity-2 coplanar-trans-1 ncop--ncols ncoplanar-perm-19*)
then show *?thesis*
using *LEM1 P2 P3 P4 col-cop2--cop* **by** *blast*
next
assume *P9*: ¬ *Col P Q D*
have *P10*: *Coplanar P Q D A*
using *NCol-cases assms*(*1*) *assms*(*2*) *assms*(*5*) *coplanar-trans-1 ncoplanar-perm-8* **by** *blast*
have *P11*: *Coplanar P Q D B*
using *assms*(*1*) *assms*(*3*) *assms*(*5*) *col-permutation-1 coplanar-perm-12 coplanar-trans-1* **by** *blast*
have *Coplanar P Q D C*
by (*meson assms*(*1*) *assms*(*4*) *assms*(*5*) *coplanar-perm-7 coplanar-trans-1 ncoplanar-perm-14 not-col-permutation-3*)
then show *?thesis* **using** *P9 P10 P11*
by (*smt P10 P11 P9 col3 coplanar-trans-1 ncop--ncol ncoplanar-perm-20 ncoplanar-perm-23 not-col-distincts*)
qed

lemma *l9-30*:
assumes ¬ *Coplanar A B C P* **and**
¬ *Col D E F* **and**
Coplanar D E F P **and**
Coplanar A B C X **and**
Coplanar A B C Y **and**
Coplanar A B C Z **and**
Coplanar D E F X **and**
Coplanar D E F Y **and**
Coplanar D E F Z
shows *Col X Y Z*
proof −
{
assume *P1*: ¬ *Col X Y Z*
have *P2*: ¬ *Col A B C*
using *assms*(*1*) *col--coplanar* **by** *blast*
have *Coplanar A B C P*
proof −
have *Q2*: *Coplanar X Y Z A*
by (*smt P2 assms*(*4*) *assms*(*5*) *assms*(*6*) *col2-cop--cop coplanar-trans-1 ncoplanar-perm-18 not-col-distincts*)
have *Q3*: *Coplanar X Y Z B*
using *P2 assms*(*4*) *assms*(*5*) *assms*(*6*) *col-trivial-3 coplanar-pseudo-trans ncop--ncols* **by** *blast*
have *Q4*: *Coplanar X Y Z C*
using *P2 assms*(*4*) *assms*(*5*) *assms*(*6*) *col-trivial-2 coplanar-pseudo-trans ncop--ncols* **by** *blast*
have *Coplanar X Y Z P*
using *assms*(*2*) *assms*(*3*) *assms*(*7*) *assms*(*8*) *assms*(*9*) *coplanar-pseudo-trans* **by** *blast*
then show *?thesis* **using** *P1 Q2 Q3 Q4*
using *assms*(*2*) *assms*(*3*) *assms*(*7*) *assms*(*8*) *assms*(*9*) *coplanar-pseudo-trans* **by** *blast*
qed
then have *False* **using** *assms*(*1*) **by** *blast*
}
then show *?thesis* **by** *blast*
qed

lemma *cop-per2--col*:
assumes *Coplanar A X Y Z* **and**
$A \neq Z$ **and**
Per X Z A **and**
Per Y Z A
shows *Col X Y Z*
proof *cases*

assume *X* = *Y* ∨ *X* = *Z* ∨ *Y* = *Z*
then show *?thesis*
using *not-col-distincts* **by** *blast*
next
assume *H1*:¬ (*X* = *Y* ∨ *X* = *Z* ∨ *Y* = *Z*)
obtain *B* **where** *P1*: *Cong X A X B* ∧ *Z Midpoint A B* ∧ *Cong Y A Y B*
using *Per-def assms*(*3*) *assms*(*4*) *per-double-cong* **by** *blast*
have *P2*: *X* ≠ *Y*
using *H1* **by** *blast*
have *P3*: *X* ≠ *Z*
using *H1* **by** *blast*
have *P4*: *Y* ≠ *Z*
using *H1* **by** *blast*
obtain *I* **where** *P5*: *Col A X I* ∧ *Col Y Z I* ∨
Col A Y I ∧ *Col X Z I* ∨ *Col A Z I* ∧ *Col X Y I*
using *Coplanar-def assms*(*1*) **by** *auto*
have *P6*: *Col A X I* ∧ *Col Y Z I* ⟶ *Col X Y Z*
by (*smt P1 P4 assms*(*2*) *l4-17 l4-18 l7-13 l7-2 l7-3-2 midpoint-distinct-2 not-col-permutation-1*)
have *P7*: *Col A Y I* ∧ *Col X Z I* ⟶ *Col X Y Z*
by (*smt P1 P3 assms*(*2*) *col-permutation-1 col-permutation-5 l4-17 l4-18 l7-13 l7-2 l7-3-2 midpoint-distinct-2*)
have *Col A Z I* ∧ *Col X Y I* ⟶ *Col X Y Z*
by (*metis P1 P2 assms*(*2*) *col-permutation-1 l4-17 l4-18 l7-13 l7-2 l7-3-2 midpoint-distinct-2*)
then show *?thesis*
using *P5 P6 P7* **by** *blast*
qed

lemma *cop-perp2--col*:
assumes *Coplanar A B Y Z* **and**
X Y Perp A B **and**
X Z Perp A B
shows *Col X Y Z*
proof *cases*
assume *P1*: *Col A B X*
{
assume *Q0*: *X* = *A*
then have *Q1*: *X* ≠ *B*
using *assms*(*3*) *perp-not-eq-2* **by** *blast*
have *Q2*: *Coplanar B Y Z X*
by (*simp add*: *Q0 assms*(*1*) *coplanar-perm-9*)
have *Q3*: *Per Y X B*
using *Q0 assms*(*2*) *perp-per-2* **by** *blast*
have *Per Z X B*
using *Q0 assms*(*3*) *perp-per-2* **by** *blast*
then have *Col X Y Z*
using *Q1 Q2 Q3 cop-per2--col not-col-permutation-1* **by** *blast*
}
then have *P2*: *X* = *A* ⟶ *Col X Y Z* **by** *blast*
{
assume *Q0*: *X* ≠ *A*
have *Q1*: *A X Perp X Y*
by (*metis P1 Perp-perm Q0 assms*(*2*) *perp-col1*)
have *Q2*: *A X Perp X Z*
by (*metis P1 Perp-perm Q0 assms*(*3*) *perp-col1*)
have *Q3*: *Coplanar A Y Z X*
by (*smt P1 assms*(*1*) *assms*(*2*) *col2-cop--cop col-trivial-3 coplanar-perm-12 coplanar-perm-16 perp-distinct*)
have *Q4*: *Per Y X A*
using *Perp-perm Q1 perp-per-2* **by** *blast*
have *Per Z X A*
using *P1 Q0 assms*(*3*) *perp-col1 perp-per-1* **by** *auto*
then have *Col X Y Z*
using *Q0 Q3 Q4 cop-per2--col not-col-permutation-1* **by** *blast*
}
then have *X* ≠ *A* ⟶ *Col X Y Z* **by** *blast*
then show *?thesis*
using *P2* **by** *blast*
next

assume *P1*: ¬ *Col A B X*
obtain *Y0* **where** *P2*: *Y0 PerpAt X Y A B*
using *Perp-def assms*(*2*) **by** *blast*
obtain *Z0* **where** *P3*: *Z0 PerpAt X Z A B*
using *Perp-def assms*(*3*) **by** *auto*
have *P4*: *X Y0 Perp A B*
by (*metis P1 P2 assms*(*2*) *perp-col perp-in-col*)
have *P5*: *X Z0 Perp A B*
by (*metis P1 P3 assms*(*3*) *perp-col perp-in-col*)
have *P6*: *Y0* = *Z0*
by (*meson P1 P2 P3 P4 P5 Perp-perm l8-18-uniqueness perp-in-col*)
have *P7*: *X* ≠ *Y0*
using *P4 perp-not-eq-1* **by** *blast*
have *P8*: *Col X Y0 Y*
using *P2 col-permutation-5 perp-in-col* **by** *blast*
have *Col X Y0 Z*
using *P3 P6 col-permutation-5 perp-in-col* **by** *blast*
then show *?thesis*
using *P7 P8 col-transitivity-1* **by** *blast*
qed

lemma *two-sides-dec*:
shows *A B TS C D* ∨ ¬ *A B TS C D*
by *simp*

lemma *cop-nts--os*:
assumes *Coplanar A B C D* **and**
¬ *Col C A B* **and**
¬ *Col D A B* **and**
¬ *A B TS C D*
shows *A B OS C D*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *cop--one-or-two-sides* **by** *blast*

lemma *cop-nos--ts*:
assumes *Coplanar A B C D* **and**
¬ *Col C A B* **and**
¬ *Col D A B* **and**
¬ *A B OS C D*
shows *A B TS C D*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *cop-nts--os* **by** *blast*

lemma *one-side-dec*:
A B OS C D ∨ ¬ *A B OS C D*
by *simp*

lemma *cop-dec*:
Coplanar A B C D ∨ ¬ *Coplanar A B C D*
by *simp*

lemma *ex-diff-cop*:
∃ *E*. *Coplanar A B C E* ∧ *D* ≠ *E*
by (*metis col-trivial-2 diff-col-ex ncop--ncols*)

lemma *ex-ncol-cop*:
assumes *D* ≠ *E*
shows ∃ *F*. *Coplanar A B C F* ∧ ¬ *Col D E F*
proof *cases*
assume *Col A B C*
then show *?thesis*
using *assms ncop--ncols not-col-exists* **by** *blast*
next
assume *P1*: ¬ *Col A B C*
then show *?thesis*
proof −
have *P2*: (*Col D E A* ∧ *Col D E B*) ⟶ (∃ *F*. *Coplanar A B C F* ∧ ¬ *Col D E F*)
by (*meson P1 assms col3 col-trivial-2 ncop--ncols*)

have *P3*: (¬*Col D E A* ∧ *Col D E B*) ⟶ (∃ *F*. *Coplanar A B C F* ∧ ¬ *Col D E F*)
  using *col-trivial-3 ncop--ncols* **by** *blast*
have *P4*: (*Col D E A* ∧ ¬*Col D E B*) ⟶ (∃ *F*. *Coplanar A B C F* ∧ ¬ *Col D E F*)
  using *col-trivial-2 ncop--ncols* **by** *blast*
have (¬*Col D E A* ∧ ¬*Col D E B*) ⟶ (∃ *F*. *Coplanar A B C F* ∧ ¬ *Col D E F*)
  using *col-trivial-3 ncop--ncols* **by** *blast*
then show *?thesis* **using** *P2 P3 P4* **by** *blast*
qed
qed

lemma *ex-ncol-cop2*:
∃ *E F*. (*Coplanar A B C E* ∧ *Coplanar A B C F* ∧ ¬ *Col D E F*)
proof −
have *f1*: ∀ *p pa pb*. *Coplanar pb pa p pb*
  by (*meson col-trivial-3 ncop--ncols*)
have *f2*: ∀ *p pa pb*. *Coplanar pb pa p p*
  by (*meson Col-perm col-trivial-3 ncop--ncols*)
obtain *pp* :: ′*p* ⇒ ′*p* ⇒ ′*p* **where**
  f3: ∀ *p pa*. *p* = *pa* ∨ ¬ *Col p pa* (*pp p pa*)
  using *not-col-exists* **by** *moura*
have *f4*: ∀ *p pa pb*. *Coplanar pb pb pa p*
  by (*meson Col-perm col-trivial-3 ncop--ncols*)
have ∃ *p*. *A* ≠ *p*
  by (*meson col-trivial-3 diff-col-ex3*)
moreover
{ **assume** *B* ≠ *A*
  then have *D* = *B* ⟶ (∃ *p*. ¬ *Col D p A* ∧ *Coplanar A B C p*)
    using *f3 f2* **by** (*metis* (*no-types*) *Col-perm ncop--ncols*)
  then have *D* = *B* ⟶ (∃ *p pa*. *Coplanar A B C p* ∧ *Coplanar A B C pa* ∧ ¬ *Col D p pa*)
    using *f1* **by** *blast* }
moreover
{ **assume** *D* ≠ *B*
  moreover
  { **assume** ∃ *p*. *D* ≠ *B* ∧ ¬ *Coplanar A B C p*
    then have *D* ≠ *B* ∧ ¬ *Col A B C*
      using *ncop--ncols* **by** *blast*
    then have ∃ *p*. ¬ *Col D p B* ∧ *Coplanar A B C p*
      using *f2 f1* **by** (*metis* (*no-types*) *Col-perm col-transitivity-1*) }
  ultimately have *?thesis*
    using *f3* **by** (*metis* (*no-types*) *col-trivial-3 ncop--ncols*) }
ultimately show *?thesis*
  using *f4 f3* **by** *blast*
qed

lemma *col2-cop2--eq*:
assumes ¬ *Coplanar A B C U* **and**
  U ≠ *V* **and**
  Coplanar A B C P **and**
  Coplanar A B C Q **and**
  Col U V P **and**
  Col U V Q
shows *P* = *Q*
proof −
have *Col U Q P*
  by (*meson assms*(*2*) *assms*(*5*) *assms*(*6*) *col-transitivity-1*)
then have *Col P Q U*
  using *not-col-permutation-3* **by** *blast*
then show *?thesis*
  using *assms*(*1*) *assms*(*3*) *assms*(*4*) *col-cop2--cop* **by** *blast*
qed

lemma *cong3-cop2--col*:
assumes *Coplanar A B C P* **and**
  Coplanar A B C Q **and**
  P ≠ *Q* **and**
  Cong A P A Q **and**

Cong B P B Q **and**
Cong C P C Q
shows *Col A B C*
proof *cases*
assume *Col A B C*
then show *?thesis* **by** *blast*
next
assume *P1*: ¬ *Col A B C*
obtain *M* **where** *P2*: *M Midpoint P Q*
using *assms*(*6*) *l7-25* **by** *blast*
have *P3*: *Per A M P*
using *P2 Per-def assms*(*4*) **by** *blast*
have *P4*: *Per B M P*
using *P2 Per-def assms*(*5*) **by** *blast*
have *P5*: *Per C M P*
using *P2 Per-def assms*(*6*) **by** *blast*
have *False*
proof *cases*
assume *Q1*: *A* = *M*
have *Q2*: *Coplanar P B C A*
using *assms*(*1*) *ncoplanar-perm-21* **by** *blast*
have *Q3*: *P* ≠ *A*
by (*metis assms*(*3*) *assms*(*4*) *cong-diff-4*)
have *Q4*: *Per B A P*
by (*simp add*: *P4 Q1*)
have *Q5*: *Per C A P*
by (*simp add*: *P5 Q1*)
then show *?thesis* **using** *Q1 Q2 Q3 Q4 cop-per2--col*
using *P1 not-col-permutation-1* **by** *blast*
next
assume *Q0*: *A* ≠ *M*
have *Q1*: *Col A B M*
proof −
have *R1*: *Coplanar A B P Q*
using *P1 assms*(*1*) *assms*(*2*) *coplanar-trans-1 ncoplanar-perm-8 not-col-permutation-2* **by** *blast*
then have *R2*: *Coplanar P A B M*
using *P2 bet-cop--cop coplanar-perm-14 midpoint-bet ncoplanar-perm-6* **by** *blast*
have *R3*: *P* ≠ *M*
using *P2 assms*(*3*) *l7-3-2 l7-9-bis* **by** *blast*
have *R4*: *Per A M P*
by (*simp add*: *P3*)
have *R5*: *Per B M P*
by (*simp add*: *P4*)
then show *?thesis*
using *R2 R3 R4 cop-per2--col* **by** *blast*
qed
have *Col A C M*
proof −
have *R1*: *Coplanar P A C M*
using *P1 Q1 assms*(*1*) *col2-cop--cop coplanar-perm-22 ncoplanar-perm-3 not-col-distincts* **by** *blast*
have *R2*: *P* ≠ *M*
using *P2 assms*(*3*) *l7-3-2 symmetric-point-uniqueness* **by** *blast*
have *R3*: *Per A M P*
by (*simp add*: *P3*)
have *Per C M P*
by (*simp add*: *P5*)
then show *?thesis*
using *R1 R2 R3 cop-per2--col* **by** *blast*
qed
then show *?thesis*
using *NCol-perm P1 Q0 Q1 col-trivial-3 colx* **by** *blast*
qed
then show *?thesis* **by** *blast*
qed

lemma *l9-38*:

assumes *A B C TSP P Q*
shows *A B C TSP Q P*
using *Bet-perm TSP-def assms* **by** *blast*

lemma *l9-39*:
assumes *A B C TSP P R* **and**
Coplanar A B C D **and**
D Out P Q
shows *A B C TSP Q R*
proof −
have *P1*: ¬ *Col A B C*
using *TSP-def assms*(*1*) *ncop--ncol* **by** *blast*
have *P2*: ¬ *Coplanar A B C Q*
by (*metis TSP-def assms*(*1*) *assms*(*2*) *assms*(*3*) *col-cop2--cop l6-6 out-col out-diff2*)
have *P3*: ¬ *Coplanar A B C R*
using *TSP-def assms*(*1*) **by** *blast*
obtain *T* **where** *P3A*: *Coplanar A B C T* ∧ *Bet P T R*
using *TSP-def assms*(*1*) **by** *blast*
have *W1*: *D* = *T* ⟶ *A B C TSP Q R*
using *P2 P3 P3A TSP-def assms*(*3*) *bet-out--bet* **by** *blast*
{
assume *V1*: *D* ≠ *T*
have *V1A*: ¬ *Col P D T* **using** *P3A col-cop2--cop*
by (*metis TSP-def V1 assms*(*1*) *assms*(*2*) *col2-cop2--eq col-trivial-2*)
have *V1B*: *D T TS P R*
by (*metis P3 P3A V1A bet--ts invert-two-sides not-col-permutation-3*)
have *D T OS P Q*
using *V1A assms*(*3*) *not-col-permutation-1 out-one-side* **by** *blast*
then have *V2*: *D T TS Q R*
using *V1B l9-8-2* **by** *blast*
then obtain *T′* **where** *V3*: *Col T′ D T* ∧ *Bet Q T′ R*
using *TS-def* **by** *blast*
have *V4*: *Coplanar A B C T′*
using *Col-cases P3A V1 V3 assms*(*2*) *col-cop2--cop* **by** *blast*
then have *A B C TSP Q R*
using *P2 P3 TSP-def V3* **by** *blast*
}
then have *D* ≠ *T* ⟶ *A B C TSP Q R* **by** *blast*
then show *?thesis* **using** *W1* **by** *blast*
qed

lemma *l9-41-1*:
assumes *A B C TSP P R* **and**
A B C TSP Q R
shows *A B C OSP P Q*
using *OSP-def assms*(*1*) *assms*(*2*) **by** *blast*

lemma *l9-41-2*:
assumes *A B C TSP P R* **and**
A B C OSP P Q
shows *A B C TSP Q R*
proof −
have *P1*: ¬ *Coplanar A B C P*
using *TSP-def assms*(*1*) **by** *blast*
obtain *S* **where** *P2*: *A B C TSP P S* ∧ *A B C TSP Q S*
using *OSP-def assms*(*2*) **by** *blast*
obtain *X* **where** *P3*: *Coplanar A B C X* ∧ *Bet P X S*
using *P2 TSP-def* **by** *blast*
have *P4*: ¬ *Coplanar A B C P* ∧ ¬ *Coplanar A B C S*
using *P2 TSP-def* **by** *blast*
obtain *Y* **where** *P5*: *Coplanar A B C Y* ∧ *Bet Q Y S*
using *P2 TSP-def* **by** *blast*
have *P6*: ¬ *Coplanar A B C Q* ∧ ¬ *Coplanar A B C S*
using *P2 TSP-def* **by** *blast*
have *P7*: *X* ≠ *P* ∧ *S* ≠ *X* ∧ *Q* ≠ *Y* ∧ *S* ≠ *Y*
using *P3 P4 P5 P6* **by** *blast*

{
assume *Q1*: *Col P Q S*
have *Q2*: *X* = *Y*
proof −
have *R2*: *Q* ≠ *S*
using *P5 P6 bet-neq12--neq* by *blast*
have *R5*: *Col Q S X*
by (*smt Col-def P3 Q1 between-inner-transitivity between-symmetry col-transitivity-2*)
have *Col Q S Y*
by (*simp add*: *P5 bet-col col-permutation-5*)
then show *?thesis*
using *P2 P3 P5 R2 R5 TSP-def col2-cop2--eq* by *blast*
qed
then have *X Out P Q*
by (*metis P3 P5 P7 l6-2*)
then have *A B C TSP Q R*
using *P3 assms*(*1*) *l9-39* by *blast*
}
then have *P7*: *Col P Q S* ⟶ *A B C TSP Q R* by *blast*
{
assume *Q1*: ¬ *Col P Q S*
obtain *Z* where *Q2*: *Bet X Z Q* ∧ *Bet Y Z P*
using *P3 P5 inner-pasch* by *blast*
{
assume *X* = *Z*
then have *False*
by (*metis P2 P3 P5 Q1 Q2 TSP-def bet-col col-cop2--cop l6-16-1 not-col-permutation-5*)
}
then have *Q3*: *X* ≠ *Z* by *blast*
have *Y* ≠ *Z*
proof −
have *X* ≠ *Z*
by (*meson* ‹*X* = *Z* ⟹ *False*›)
then have *Z* ≠ *Y*
by (*metis* (*no-types*) *P2 P3 P5 Q2 TSP-def bet-col col-cop2--cop*)
then show *?thesis*
by *meson*
qed
then have *Y Out P Z*
using *Q2 bet-out l6-6* by *auto*
then have *Q4*: *A B C TSP Z R*
using *assms*(*1*) *P5 l9-39* by *blast*
have *X Out Z Q*
using *Q2 Q3 bet-out* by *auto*
then have *A B C TSP Q R*
using *Q4 P3 l9-39* by *blast*
}
then have ¬ *Col P Q S* ⟶ *A B C TSP Q R* by *blast*
then show *?thesis* using *P7* by *blast*
qed

lemma *tsp-exists*:
assumes ¬ *Coplanar A B C P*
shows ∃ *Q*. *A B C TSP P Q*
proof −
obtain *Q* where *P1*: *Bet P A Q* ∧ *Cong A Q A P*
using *segment-construction* by *blast*
have *P2*: *Coplanar A B C A*
using *coplanar-trivial ncoplanar-perm-5* by *blast*
have *P3*: ¬ *Coplanar A B C P*
by (*simp add*: *assms*)
have *P4*: ¬ *Coplanar A B C Q*
by (*metis P1 P2 Tarski-neutral-dimensionless.col-cop2--cop Tarski-neutral-dimensionless-axioms assms bet-col cong-diff-4 not-col-permutation-2*)
then show *?thesis*
using *P1 P2 TSP-def assms* by *blast*

qed

lemma *osp-reflexivity*:
  assumes ¬ *Coplanar A B C P*
  shows *A B C OSP P P*
  by (*meson assms l9-41-1 tsp-exists*)

lemma *osp-symmetry*:
  assumes *A B C OSP P Q*
  shows *A B C OSP Q P*
  using *OSP-def assms* **by** *auto*

lemma *osp-transitivity*:
  assumes *A B C OSP P Q* **and**
    A B C OSP Q R
  shows *A B C OSP P R*
  using *OSP-def assms*(*1*) *assms*(*2*) *l9-41-2* **by** *blast*

lemma *cop3-tsp--tsp*:
  assumes ¬ *Col D E F* **and**
    Coplanar A B C D **and**
    Coplanar A B C E **and**
    Coplanar A B C F **and**
    A B C TSP P Q
  shows *D E F TSP P Q*
proof −
  obtain *T* **where** *P1*: *Coplanar A B C T* ∧ *Bet P T Q*
    using *TSP-def assms*(*5*) **by** *blast*
  have *P2*: ¬ *Col A B C*
    using *TSP-def assms*(*5*) *ncop--ncols* **by** *blast*
  have *P3*: *Coplanar D E F A* ∧ *Coplanar D E F B* ∧ *Coplanar D E F C* ∧ *Coplanar D E F T*
  proof −
    have *P3A*: *Coplanar D E F A*
      using *P2 assms*(*2*) *assms*(*3*) *assms*(*4*) *col-trivial-3 coplanar-pseudo-trans ncop--ncols* **by** *blast*
    have *P3B*: *Coplanar D E F B*
      using *P2 assms*(*2*) *assms*(*3*) *assms*(*4*) *col-trivial-2 coplanar-pseudo-trans ncop--ncols* **by** *blast*
    have *P3C*: *Coplanar D E F C*
      by (*meson P2 assms*(*2*) *assms*(*3*) *assms*(*4*) *coplanar-perm-16 coplanar-pseudo-trans coplanar-trivial*)
    have *Coplanar D E F T*
      using *P1 P2 assms*(*2*) *assms*(*3*) *assms*(*4*) *coplanar-pseudo-trans* **by** *blast*
    then show *?thesis* **using** *P3A P3B P3C* **by** *simp*
  qed
  have *P4*: ¬ *Coplanar D E F P*
    using *P3 TSP-def assms*(*1*) *assms*(*5*) *coplanar-pseudo-trans* **by** *auto*
  have *P5*: ¬ *Coplanar D E F Q*
    by (*metis P1 P3 P4 TSP-def assms*(*5*) *bet-col bet-col1 col2-cop2--eq*)
  have *P6*: *Coplanar D E F T*
    by (*simp add*: *P3*)
  have *Bet P T Q*
    by (*simp add*: *P1*)
  then show *?thesis*
    using *P4 P5 P6 TSP-def* **by** *blast*
qed

lemma *cop3-osp--osp*:
  assumes ¬ *Col D E F* **and**
    Coplanar A B C D **and**
    Coplanar A B C E **and**
    Coplanar A B C F **and**
    A B C OSP P Q
  shows *D E F OSP P Q*
proof −
  obtain *R* **where** *P1*: *A B C TSP P R* ∧ *A B C TSP Q R*
    using *OSP-def assms*(*5*) **by** *blast*
  then show *?thesis*

using *OSP-def assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *cop3-tsp--tsp* **by** *blast*
qed

lemma *ncop-distincts*:
assumes ¬ *Coplanar A B C D*
shows $A \neq B \wedge A \neq C \wedge A \neq D \wedge B \neq C \wedge B \neq D \wedge C \neq D$
using *Coplanar-def assms col-trivial-1 col-trivial-2* **by** *blast*

lemma *tsp-distincts*:
assumes *A B C TSP P Q*
shows $A \neq B \wedge A \neq C \wedge B \neq C \wedge A \neq P \wedge B \neq P \wedge C \neq P \wedge A \neq Q \wedge B \neq Q \wedge C \neq Q \wedge P \neq Q$
proof −
obtain *pp* :: $'p \Rightarrow 'p \Rightarrow 'p \Rightarrow 'p \Rightarrow 'p \Rightarrow 'p$ **where**
∀ *x0 x1 x2 x3 x4*. (∃ *v5*. *Coplanar x4 x3 x2 v5* ∧ *Bet x1 v5 x0*) = (*Coplanar x4 x3 x2* (*pp x0 x1 x2 x3 x4*) ∧ *Bet x1* (*pp x0 x1 x2 x3 x4*) *x0*)
by *moura*
then have *f1*: ¬ *Coplanar A B C P* ∧ ¬ *Coplanar A B C Q* ∧ *Coplanar A B C* (*pp Q P C B A*) ∧ *Bet P* (*pp Q P C B A*) *Q*
using *TSP-def assms* **by** *presburger*
then have *Q* ≠ *pp Q P C B A*
by *force*
then show *?thesis*
using *f1* **by** (*meson bet-neq32--neq ncop-distincts*)
qed

lemma *osp-distincts*:
assumes *A B C OSP P Q*
shows $A \neq B \wedge A \neq C \wedge B \neq C \wedge A \neq P \wedge B \neq P \wedge C \neq P \wedge A \neq Q \wedge B \neq Q \wedge C \neq Q$
using *OSP-def assms tsp-distincts* **by** *blast*

lemma *tsp--ncop1*:
assumes *A B C TSP P Q*
shows ¬ *Coplanar A B C P*
using *TSP-def assms* **by** *blast*

lemma *tsp--ncop2*:
assumes *A B C TSP P Q*
shows ¬ *Coplanar A B C Q*
using *TSP-def assms* **by** *auto*

lemma *osp--ncop1*:
assumes *A B C OSP P Q*
shows ¬ *Coplanar A B C P*
using *OSP-def TSP-def assms* **by** *blast*

lemma *osp--ncop2*:
assumes *A B C OSP P Q*
shows ¬ *Coplanar A B C Q*
using *assms osp--ncop1 osp-symmetry* **by** *blast*

lemma *tsp--nosp*:
assumes *A B C TSP P Q*
shows ¬ *A B C OSP P Q*
using *assms l9-41-2 tsp-distincts* **by** *blast*

lemma *osp--ntsp*:
assumes *A B C OSP P Q*
shows ¬ *A B C TSP P Q*
using *assms tsp--nosp* **by** *blast*

lemma *osp-bet--osp*:
assumes *A B C OSP P R* **and**
Bet P Q R
shows *A B C OSP P Q*
proof −
obtain *S* **where** *P1*: *A B C TSP P S*

```
    using OSP-def assms(1) by blast
  then obtain Y where P2: Coplanar A B C Y ∧ Bet R Y S
    using TSP-def assms(1) l9-41-2 by blast
  obtain X where Q1: Coplanar A B C X ∧ Bet P X S
    using P1 TSP-def by blast
  have Q2: P ≠ X ∧ S ≠ X ∧ R ≠ Y
    using P1 P2 Q1 TSP-def assms(1) osp--ncop2 by auto
  {
    assume P3: Col P R S
    have P5: A B C TSP Q S
    proof −
      have Q3: X = Y
      proof −
        have R1: ¬ Coplanar A B C R
          using assms(1) osp--ncop2 by blast
        have R2: R ≠ S
          using P1 assms(1) osp--ntsp by blast
        have R5: Col R S X
          by (smt Col-def P3 Q1 bet-col1 between-exchange4 between-symmetry)
        have Col R S Y
          using P2 bet-col col-permutation-5 by blast
        then show ?thesis
          using R1 R2 Q1 P2 R5 col2-cop2--eq by blast
      qed
      then have Y Out P Q
         by (smt P2 P3 Q1 Q2 assms(2) bet-col1 between-exchange4 between-symmetry l6-3-2 l6-4-2 not-bet-and-out
third-point)
      then show ?thesis
        using P1 P2 l9-39 by blast
    qed
    then have A B C OSP P Q
      using OSP-def P1 P2 l9-39 by blast
  }
  then have H1: Col P R S ⟶ A B C OSP P Q by blast
  {
    assume T1: ¬ Col P R S
    have T2: X Y OS P R
    proof −
      have T3: P ≠ X ∧ S ≠ X ∧ R ≠ Y ∧ S ≠ Y
        using P1 P2 Q2 TSP-def by auto
      have T4: ¬ Col S X Y
        by (metis P2 Q1 T1 T3 bet-out-1 col-out2-col col-permutation-5 not-col-permutation-4)
      have T5: X Y TS P S
        by (metis Col-perm Q1 Q2 T4 bet--ts bet-col col-transitivity-2)
      have T6: X Y TS R S
        by (metis P2 Q1 T4 assms(1) bet--ts col-cop2--cop invert-two-sides not-col-distincts osp--ncop2)
      then show ?thesis
        using T5 l9-8-1 by auto
    qed
    then have T7: X Y OS P Q
      using assms(2) l9-17 by blast
    then obtain S' where T7A: X Y TS P S' ∧ X Y TS Q S'
      using OS-def by blast
    have T7B: ¬ Col P X Y ∧ ¬ Col S' X Y ∧ (∃ T::'p. Col T X Y ∧ Bet P T S')
      using T7A TS-def by auto
    have T7C: ¬ Col Q X Y ∧ ¬ Col S' X Y ∧ (∃ T::'p. Col T X Y ∧ Bet Q T S')
      using T7A TS-def by blast
    obtain X' where T9: Col X' X Y ∧ Bet P X' S' ∧ X Y TS Q S'
      using T7A T7B by blast
    obtain Y' where T10: Col Y' X Y ∧ Bet Q Y' S'
      using T7C by blast
    have T11: Coplanar A B C X'
      using Col-cases P2 Q1 T9 col-cop2--cop ts-distincts by blast
    have T12: Coplanar A B C Y'
      using Col-cases P2 Q1 T10 T9 col-cop2--cop ts-distincts by blast
    have T13: ¬ Coplanar A B C S'
```

    using *T11 T7C T9 assms*(*1*) *bet-col bet-col1 col2-cop2--eq osp--ncop1* **by** *fastforce*
  then have *A B C OSP P Q*
  proof −
    have *R1*: *A B C TSP P S′*
      using *P1 T11 T13 T9 TSP-def* **by** *blast*
    have *A B C TSP Q S′*
      by (*metis T10 T12 T13 T7C TSP-def bet-col col-cop2--cop*)
    then show *?thesis* **using** *R1* **by** (*smt l9-41-1*)
  qed
 }
 then show *?thesis* **using** *H1* **by** *blast*
qed

lemma *l9-18-3*:
 assumes *Coplanar A B C P* **and**
  Col X Y P
 shows *A B C TSP X Y* ⟷ (*Bet X P Y* ∧ ¬ *Coplanar A B C X* ∧ ¬ *Coplanar A B C Y*)
 by (*meson TSP-def assms*(*1*) *assms*(*2*) *l9-39 not-bet-out not-col-permutation-5 tsp-distincts*)

lemma *bet-cop--tsp*:
 assumes ¬ *Coplanar A B C X* **and**
  $P \neq Y$ **and**
  Coplanar A B C P **and**
  Bet X P Y
 shows *A B C TSP X Y*
 using *TSP-def assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *bet-col bet-col1 col2-cop2--eq* **by** *fastforce*

lemma *cop-out--osp*:
 assumes ¬ *Coplanar A B C X* **and**
  Coplanar A B C P **and**
  P Out X Y
 shows *A B C OSP X Y*
 by (*meson OSP-def assms*(*1*) *assms*(*2*) *assms*(*3*) *l9-39 tsp-exists*)

lemma *l9-19-3*:
 assumes *Coplanar A B C P* **and**
  Col X Y P
 shows *A B C OSP X Y* ⟷ (*P Out X Y* ∧ ¬ *Coplanar A B C X*)
 by (*meson assms*(*1*) *assms*(*2*) *cop-out--osp l6-4-2 l9-18-3 not-col-permutation-5 osp--ncop1 osp--ncop2 tsp--nosp*)

lemma *cop2-ts--tsp*:
 assumes ¬ *Coplanar A B C X* **and** *Coplanar A B C D* **and**
  Coplanar A B C E **and** *D E TS X Y*
 shows *A B C TSP X Y*
proof −
 obtain *T* **where** *P1*: *Col T D E* ∧ *Bet X T Y*
  using *TS-def assms*(*4*) **by** *blast*
 have *P2*: *Coplanar A B C T*
  using *P1 assms*(*2*) *assms*(*3*) *assms*(*4*) *col-cop2--cop not-col-permutation-2 ts-distincts* **by** *blast*
 then show *?thesis*
  by (*metis P1 TS-def assms*(*1*) *assms*(*4*) *bet-cop--tsp*)
qed

lemma *cop2-os--osp*:
 assumes ¬ *Coplanar A B C X* **and**
  Coplanar A B C D **and**
  Coplanar A B C E **and**
  D E OS X Y
 shows *A B C OSP X Y*
proof −
 obtain *Z* **where** *P1*: *D E TS X Z* ∧ *D E TS Y Z*
  using *OS-def assms*(*4*) **by** *blast*
 then have *P2*: *A B C TSP X Z*
  using *assms*(*1*) *assms*(*2*) *assms*(*3*) *cop2-ts--tsp* **by** *blast*
 then have *P3*: *A B C TSP Y Z*
  by (*meson P1 assms*(*2*) *assms*(*3*) *cop2-ts--tsp l9-2 tsp--ncop2*)

then show *?thesis*
using *P2 l9-41-1* **by** *blast*
qed

lemma *cop3-tsp--ts*:
assumes $D \neq E$ **and**
Coplanar A B C D **and**
Coplanar A B C E **and**
Coplanar D E X Y **and**
A B C TSP X Y
shows *D E TS X Y*
by (*meson assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *col-cop2--cop cop2-os--osp cop-nts--os not-col-permutation-2 tsp--ncop1 tsp--ncop2 tsp--nosp*)

lemma *cop3-osp--os*:
assumes $D \neq E$ **and**
Coplanar A B C D **and**
Coplanar A B C E **and**
Coplanar D E X Y **and**
A B C OSP X Y
shows *D E OS X Y*
by (*meson assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *col-cop2--cop cop2-ts--tsp cop-nts--os not-col-permutation-2 osp--ncop1 osp--ncop2 tsp--nosp*)

lemma *cop-tsp--ex-cop2*:
assumes
A B C TSP D E
shows ∃ *Q*. (*Coplanar A B C Q* ∧ *Coplanar D E P Q* ∧ $P \neq Q$)
proof *cases*
assume *Col D E P*
then show *?thesis*
by (*meson ex-diff-cop ncop--ncols*)
next
assume ¬ *Col D E P*
then obtain *Q* **where** *Coplanar A B C Q* ∧ *Bet D Q E* ∧ ¬ *Col D E P*
using *TSP-def assms*(*1*) **by** *blast*
then show *?thesis*
using *Col-perm bet-col ncop--ncols* **by** *blast*
qed

lemma *cop-osp--ex-cop2*:
assumes *Coplanar A B C P* **and**
A B C OSP D E
shows ∃ *Q*. *Coplanar A B C Q* ∧ *Coplanar D E P Q* ∧ $P \neq Q$
proof *cases*
assume *Col D E P*
then show *?thesis*
by (*metis col-trivial-3 diff-col-ex ncop--ncols*)
next
assume *P1*: ¬ *Col D E P*
obtain E' **where** *P2*: *Bet E P* E' ∧ *Cong P* E' *P E*
using *segment-construction* **by** *blast*
have *P3*: ¬ *Col D* E' *P*
by (*metis P1 P2 bet-col bet-cong-eq between-symmetry col-permutation-5 l5-2 l6-16-1*)
have *P4*: *A B C TSP D* E'
by (*metis P2 P3 assms*(*1*) *assms*(*2*) *bet-cop--tsp l9-41-2 not-col-distincts osp--ncop2 osp-symmetry*)
then have ¬ *Coplanar A B C D* ∧ ¬ *Coplanar A B C* E' ∧ (∃ *T*. *Coplanar A B C T* ∧ *Bet D T* E')
by (*simp add*: *TSP-def*)
then obtain *Q* **where** *P7*: *Coplanar A B C Q* ∧ *Bet D Q* E'
by *blast*
then have *Coplanar D* E' *P Q*
using *bet-col ncop--ncols ncoplanar-perm-5* **by** *blast*
then have *Coplanar D E P Q*
using *Col-perm P2 P3 bet-col col-cop--cop ncoplanar-perm-5 not-col-distincts* **by** *blast*
then show *?thesis*
using *P3 P7 bet-col col-permutation-5* **by** *blast*

qed

lemma *sac--coplanar*:
  assumes *Saccheri A B C D*
  shows *Coplanar A B C D*
  using *Saccheri-def assms ncoplanar-perm-4 os--coplanar* **by** *blast*

3.9 Line reflexivity

3.9.1 Dimensionless

lemma *Ch10-Goal1*:
  assumes ¬ *Coplanar D C B A*
  shows ¬ *Coplanar A B C D*
  by (*simp add*: *assms ncoplanar-perm-23*)

lemma *ex-sym*:
  ∃ *Y*. (*A B Perp X Y* ∨ *X* = *Y*) ∧ (∃ *M*. *Col A B M* ∧ *M Midpoint X Y*)
proof *cases*
  assume *Col A B X*
  thus *?thesis*
    using *l7-3-2* **by** *blast*
next
  assume ¬ *Col A B X*
  then obtain *M0* **where** *P1*: *Col A B M0* ∧ *A B Perp X M0*
    using *l8-18-existence* **by** *blast*
  obtain *Z* **where** *P2*: *M0 Midpoint X Z*
    using *symmetric-point-construction* **by** *blast*
  thus *?thesis*
    by (*metis* (*full-types*) *P1 Perp-cases bet-col midpoint-bet perp-col*)
qed

lemma *is-image-is-image-spec*:
  assumes *A* ≠ *B*
  shows *P′ P Reflect A B* ⟷ *P′ P ReflectL A B*
  by (*simp add*: *Reflect-def assms*)

lemma *ex-sym1*:
  assumes *A* ≠ *B*
  shows ∃ *Y*. (*A B Perp X Y* ∨ *X* = *Y*) ∧ (∃ *M*. *Col A B M* ∧ *M Midpoint X Y* ∧ *X Y Reflect A B*)
proof *cases*
  assume *Col A B X*
  thus *?thesis*
    by (*meson ReflectL-def Reflect-def assms l7-3-2*)
next
  assume *P0*: ¬ *Col A B X*
  then obtain *M0* **where** *P1*: *Col A B M0* ∧ *A B Perp X M0*
    using *l8-18-existence* **by** *blast*
  obtain *Z* **where** *P2*: *M0 Midpoint X Z*
    using *symmetric-point-construction* **by** *blast*
  have *P3*: *A B Perp X Z*
  proof *cases*
    assume *X* = *Z*
    thus *?thesis*
      using *P1 P2 P0 midpoint-distinct* **by** *blast*
  next
    assume *X* ≠ *Z*
    then have *P2*: *X Z Perp A B*
      using *P1 P2 Perp-cases bet-col midpoint-bet perp-col* **by** *blast*
    show *?thesis*
      by (*simp add*: *Tarski-neutral-dimensionless.Perp-perm Tarski-neutral-dimensionless-axioms P2*)
  qed
  have *P10*: (*A B Perp X Z* ∨ *X* = *Z*)
    by (*simp add*: *P3*)
  have ∃ *M*. *Col A B M* ∧ *M Midpoint X Z* ∧ *X Z Reflect A B*
    using *P1 P2 P3 ReflectL-def assms is-image-is-image-spec l7-2 perp-right-comm* **by** *blast*
  thus *?thesis*

using *P3* **by** *blast*
qed

lemma *l10-2-uniqueness*:
assumes *P1 P Reflect A B* **and**
P2 P Reflect A B
shows *P1 = P2*
proof *cases*
assume *A = B*
thus *?thesis*
using *Reflect-def assms(1) assms(2) symmetric-point-uniqueness* **by** *auto*
next
assume *P1*: *A ≠ B*
have *P1A*: *P1 P ReflectL A B*
using *P1 assms(1) is-image-is-image-spec* **by** *auto*
then have *P1B*: *A B Perp P P1 ∨ P = P1*
using *ReflectL-def* **by** *blast*
have *P2A*: *P2 P ReflectL A B*
using *P1 assms(2) is-image-is-image-spec* **by** *auto*
then have *P2B*: *A B Perp P P2 ∨ P = P2*
using *ReflectL-def* **by** *blast*
obtain *X* **where** *R1*: *X Midpoint P P1 ∧ Col A B X*
by (*metis ReflectL-def assms(1) col-trivial-1 is-image-is-image-spec midpoint-existence*)
obtain *Y* **where** *R2*: *Y Midpoint P P2 ∧ Col A B Y*
by (*metis ReflectL-def assms(2) col-trivial-1 is-image-is-image-spec midpoint-existence*)
{
assume *Q1*:(*A B Perp P P1 ∧ A B Perp P P2*)
have *S1*: *P ≠ X*
proof −
{
assume *P = X*
then have *P = P1*
using *R1 is-midpoint-id* **by** *blast*
then have *A B Perp P P*
using *Q1* **by** *blast*
then have *False*
using *perp-distinct* **by** *blast*
}
thus *?thesis* **by** *blast*
qed
then have *P1 = P2*
by (*smt Perp-cases Q1* ‹⋀*thesis*. (⋀*X*. *X Midpoint P P1 ∧ Col A B X ⟹ thesis*) ⟹ *thesis*› ‹⋀*thesis*. (⋀*Y*. *Y Midpoint P P2 ∧ Col A B Y ⟹ thesis*) ⟹ *thesis*› *col-permutation-1 l7-2 l7-9 l8-18-uniqueness midpoint-col perp-col perp-not-col2*)
}
then have *T1*: (*A B Perp P P1 ∧ A B Perp P P2*) ⟶ *P1 = P2* **by** *blast*
have *T2*: (*P = P1 ∧ A B Perp P P2*) ⟶ *P1 = P2*
by (*metis R1 R2 col3 col-trivial-2 col-trivial-3 midpoint-col midpoint-distinct-1 midpoint-distinct-2 perp-not-col2*)
have *T3*: (*P = P2 ∧ A B Perp P P1*) ⟶ *P1 = P2*
by (*metis R1 R2 col-trivial-2 midpoint-col midpoint-distinct-3 perp-col2 perp-not-col2*)
thus *?thesis*
using *T1 T2 T3 P1B P2B* **by** *blast*
qed

lemma *l10-2-uniqueness-spec*:
assumes *P1 P ReflectL A B* **and**
P2 P ReflectL A B
shows *P1 = P2*
proof −
have *A B Perp P P1 ∨ P = P1*
using *ReflectL-def assms(1)* **by** *blast*
moreover obtain *X1* **where** *X1 Midpoint P P1 ∧ Col A B X1*
using *ReflectL-def assms(1)* **by** *blast*
moreover have *A B Perp P P2 ∨ P = P2*
using *ReflectL-def assms(2)* **by** *blast*
moreover obtain *X2* **where** *X2 Midpoint P P2 ∧ Col A B X2*

using *ReflectL-def assms*(*2*) **by** *blast*
ultimately show *?thesis*
by (*smt col-permutation-1 l8-16-1 l8-18-uniqueness midpoint-col midpoint-distinct-3 perp-col1 symmetric-point-uniqueness*)
qed

lemma *l10-2-existence-spec*:
∃ P'. P' *P ReflectL A B*
proof *cases*
assume *Col A B P*
thus *?thesis*
using *ReflectL-def l7-3-2* **by** *blast*
next
assume ¬ *Col A B P*
then obtain *X* **where** *Col A B X* ∧ *A B Perp P X*
using *l8-18-existence* **by** *blast*
moreover obtain P' **where** *X Midpoint P* P'
using *symmetric-point-construction* **by** *blast*
ultimately show *?thesis*
using *ReflectL-def bet-col midpoint-bet perp-col1* **by** *blast*
qed

lemma *l10-2-existence*:
∃ P'. P' *P Reflect A B*
by (*metis Reflect-def l10-2-existence-spec symmetric-point-construction*)

lemma *l10-4-spec*:
assumes *P* P' *ReflectL A B*
shows P' *P ReflectL A B*
proof −
obtain *X* **where** *X Midpoint P* P' ∧ *Col A B X*
using *ReflectL-def assms l7-2* **by** *blast*
thus *?thesis*
using *Perp-cases ReflectL-def assms* **by** *auto*
qed

lemma *l10-4*:
assumes *P* P' *Reflect A B*
shows P' *P Reflect A B*
using *Reflect-def Tarski-neutral-dimensionless.l7-2 Tarski-neutral-dimensionless-axioms assms l10-4-spec* **by** *fastforce*

lemma *l10-5*:
assumes P' *P Reflect A B* **and**
P'' P' *Reflect A B*
shows *P* = P''
by (*meson assms*(*1*) *assms*(*2*) *l10-2-uniqueness l10-4*)

lemma *l10-6-uniqueness*:
assumes *P P1 Reflect A B* **and**
P P2 Reflect A B
shows *P1* = *P2*
using *assms*(*1*) *assms*(*2*) *l10-4 l10-5* **by** *blast*

lemma *l10-6-uniqueness-spec*:
assumes *P P1 ReflectL A B* **and**
P P2 ReflectL A B
shows *P1* = *P2*
using *assms*(*1*) *assms*(*2*) *l10-2-uniqueness-spec l10-4-spec* **by** *blast*

lemma *l10-6-existence-spec*:
assumes *A* ≠ *B*
shows ∃ *P*. P' *P ReflectL A B*
using *l10-2-existence-spec l10-4-spec* **by** *blast*

lemma *l10-6-existence*:
∃ *P*. P' *P Reflect A B*
using *l10-2-existence l10-4* **by** *blast*

```
lemma l10-7:
  assumes P' P Reflect A B and
    Q' Q Reflect A B and
    P' = Q'
  shows P = Q
  using assms(1) assms(2) assms(3) l10-6-uniqueness by blast

lemma l10-8:
  assumes P P Reflect A B
  shows Col P A B
  by (metis Col-perm assms col-trivial-2 ex-sym1 l10-6-uniqueness l7-3)

lemma col--refl:
  assumes Col P A B
  shows P P ReflectL A B
  using ReflectL-def assms col-permutation-1 l7-3-2 by blast

lemma is-image-col-cong:
  assumes A ≠ B and
    P P' Reflect A B and
    Col A B X
  shows Cong P X P' X
proof −
  have P1: P P' ReflectL A B
    using assms(1) assms(2) is-image-is-image-spec by blast
  obtain M0 where P2: M0 Midpoint P' P ∧ Col A B M0
    using P1 ReflectL-def by blast
  have A B Perp P' P ∨ P' = P
    using P1 ReflectL-def by auto
  moreover
  {
    assume S1: A B Perp P' P
    then have A ≠ B ∧ P' ≠ P
      using perp-distinct by blast
    have S2: M0 = X ⟶ Cong P X P' X
      using P2 cong-4312 midpoint-cong by blast
    {
      assume M0 ≠ X
      then have M0 X Perp P' P
        using P2 S1 assms(3) perp-col2 by blast
      then have ¬ Col A B P ∧ Per P M0 X
        by (metis Col-perm P2 S1 colx l8-2 midpoint-col midpoint-distinct-1 per-col perp-col1 perp-not-col2 perp-per-1)
      then have Cong P X P' X
        using P2 cong-commutativity l7-2 l8-2 per-double-cong by blast
    }
    then have Cong P X P' X
      using S2 by blast
  }
  then have A B Perp P' P ⟶ Cong P X P' X by blast
  moreover
  {
    assume P = P'
    then have Cong P X P' X
      by (simp add: cong-reflexivity)
  }
  ultimately show ?thesis by blast
qed

lemma is-image-spec-col-cong:
  assumes P P' ReflectL A B and
    Col A B X
  shows Cong P X P' X
 by (metis Col-def Reflect-def assms(1) assms(2) between-trivial col--refl cong-reflexivity is-image-col-cong l10-6-uniqueness-spec)

lemma image-id:
```

assumes $A \neq B$ **and**
Col A B T **and**
T T′ Reflect A B
shows $T = T'$
using *assms(1) assms(2) assms(3) cong-diff-4 is-image-col-cong* **by** *blast*

lemma *osym-not-col*:
assumes *P P′ Reflect A B* **and**
¬ *Col A B P*
shows ¬ *Col A B P′*
using *assms(1) assms(2) l10-4 local.image-id not-col-distincts* **by** *blast*

lemma *midpoint-preserves-image*:
assumes $A \neq B$ **and**
Col A B M **and**
P P′ Reflect A B **and**
M Midpoint P Q **and**
M Midpoint P′ Q′
shows *Q Q′ Reflect A B*
proof −
obtain *X* **where** *P1*: *X Midpoint P′ P* ∧ *Col A B X*
using *ReflectL-def assms(1) assms(3) is-image-is-image-spec* **by** *blast*
{
assume *S1*: *A B Perp P′ P*
obtain *Y* **where** *S2*: *M Midpoint X Y*
using *symmetric-point-construction* **by** *blast*
have *S3*: *Y Midpoint Q Q′*
proof −
have *R4*: *X Midpoint P P′*
by (*simp add*: *P1 l7-2*)
thus *?thesis*
using *assms(4) assms(5) S2 symmetry-preserves-midpoint* **by** *blast*
qed
have *S4*: $P \neq P'$
using *S1 perp-not-eq-2* **by** *blast*
then have *S5*: $Q \neq Q'$
using *Tarski-neutral-dimensionless.l7-9 Tarski-neutral-dimensionless-axioms assms(4) assms(5)* **by** *fastforce*
have *S6*: *Y Midpoint Q′ Q* ∧ *Col A B Y*
by (*metis P1 S2 S3 assms(2) colx l7-2 midpoint-col midpoint-distinct-1*)
have *S7*: *A B Perp Q′ Q* ∨ $Q = Q'$
proof −
have *R3*: *Per M Y Q*
proof −
have *S1*: *Y Midpoint Q Q′*
using *S3* **by** *auto*
have *Cong M Q M Q′*
using *assms(1) assms(2) assms(3) assms(4) assms(5) cong-commutativity is-image-col-cong l7-16 l7-3-2* **by** *blast*
thus *?thesis*
using *Per-def S1* **by** *blast*
qed
{
have $X = Y \longrightarrow$ (*A B Perp Q′ Q* ∨ $Q = Q'$)
by (*metis P1 Perp-cases S1 S2 S6 assms(5) l7-3 l7-9-bis*)
{
assume $X \neq Y$
then have *Y PerpAt M Y Y Q*
using *R3 S2 S3 S5 midpoint-distinct-1 per-perp-in* **by** *blast*
then have *V1*: *Y Y Perp Y Q* ∨ *M Y Perp Y Q*
by (*simp add*: *perp-in-perp-bis*)
{
have *Y Y Perp Y Q* $\longrightarrow$ *A B Perp Q′ Q* ∨ $Q = Q'$
using *perp-not-eq-1* **by** *blast*
{
assume *T1*: *M Y Perp Y Q*
have *T2*: *Y Q Perp A B*

proof *cases*
assume $A = M$
thus *?thesis*
using *Perp-cases S6 T1 assms(1) col-permutation-5 perp-col* **by** *blast*
next
assume $A \neq M$
thus *?thesis*
by (*smt S6 T1 assms(1) assms(2) col2--eq col-transitivity-2 perp-col0 perp-not-eq-1*)
qed
have $A\ B\ Perp\ Q'\ Q \lor Q = Q'$
by (*metis S3 T2 midpoint-col not-col-distincts perp-col0*)
}
then have $M\ Y\ Perp\ Y\ Q \longrightarrow A\ B\ Perp\ Q'\ Q \lor Q = Q'$ **by** *blast*
}
then have $A\ B\ Perp\ Q'\ Q \lor Q = Q'$
using *V1 perp-distinct* **by** *blast*
}
then have $X \neq Y \longrightarrow (A\ B\ Perp\ Q'\ Q \lor Q = Q')$ **by** *blast*
}
thus *?thesis*
by (*metis P1 Perp-cases S1 S2 S6 assms(5) l7-3 l7-9-bis*)
qed
then have $Q\ Q'\ ReflectL\ A\ B$
using *ReflectL-def S6* **by** *blast*
}
then have $A\ B\ Perp\ P'\ P \longrightarrow Q\ Q'\ ReflectL\ A\ B$ **by** *blast*
moreover
{
assume $P = P'$
then have $Q\ Q'\ ReflectL\ A\ B$
by (*metis P1 assms(2) assms(4) assms(5) col--refl col-permutation-2 colx midpoint-col midpoint-distinct-3 symmetric-point-uniqueness*)
}
ultimately show *?thesis*
using *ReflectL-def assms(1) assms(3) is-image-is-image-spec* **by** *auto*
qed

lemma *image-in-is-image-spec*:
assumes $M\ ReflectLAt\ P\ P'\ A\ B$
shows $P\ P'\ ReflectL\ A\ B$
proof –
have *P1*: $M\ Midpoint\ P'\ P$
using *ReflectLAt-def assms* **by** *blast*
have *P2*: $Col\ A\ B\ M$
using *ReflectLAt-def assms* **by** *blast*
have $A\ B\ Perp\ P'\ P \lor P' = P$
using *ReflectLAt-def assms* **by** *blast*
thus *?thesis* **using** *P1 P2*
using *ReflectL-def* **by** *blast*
qed

lemma *image-in-gen-is-image*:
assumes $M\ ReflectAt\ P\ P'\ A\ B$
shows $P\ P'\ Reflect\ A\ B$
using *ReflectAt-def Reflect-def assms image-in-is-image-spec* **by** *auto*

lemma *image-image-in*:
assumes $P \neq P'$ **and**
$P\ P'\ ReflectL\ A\ B$ **and**
$Col\ A\ B\ M$ **and**
$Col\ P\ M\ P'$
shows $M\ ReflectLAt\ P\ P'\ A\ B$
proof –
obtain M' **where** *P1*: $M'\ Midpoint\ P'\ P \land Col\ A\ B\ M'$
using *ReflectL-def assms(2)* **by** *blast*
have *Q1*: $P\ M'\ Perp\ A\ B$

```
    by (metis Col-cases P1 Perp-perm ReflectL-def assms(1) assms(2) bet-col cong-diff-3 midpoint-bet midpoint-cong
not-cong-4321 perp-col1)
  {
    assume R1: A B Perp P′ P
    have R3: P ≠ M′
      using Q1 perp-not-eq-1 by auto
    have R4: A B Perp P′ P
      by (simp add: R1)
    have R5: Col P P′ M′
      using P1 midpoint-col not-col-permutation-3 by blast
    have R6: M′ Midpoint P′ P
      by (simp add: P1)
    have R7: ¬ Col A B P
      using assms(1) assms(2) col--refl col-permutation-2 l10-2-uniqueness-spec l10-4-spec by blast
    have R8: P ≠ P′
      by (simp add: assms(1))
    have R9: Col A B M′
      by (simp add: P1)
    have R10: Col A B M
      by (simp add: assms(3))
    have R11: Col P P′ M′
      by (simp add: R5)
    have R12: Col P P′ M
      using Col-perm assms(4) by blast
    have M = M′
    proof cases
      assume S1: A = M′
      have Per P M′ A
        by (simp add: S1 l8-5)
      thus ?thesis using l6-21 R8 R9 R10 R11 R12
        using R7 by blast
    next
      assume A ≠ M′
      thus ?thesis
        using R10 R12 R5 R7 R8 R9 l6-21 by blast
    qed
    then have M Midpoint P′ P
      using R6 by blast
  }
  then have Q2: A B Perp P′ P ⟶ M Midpoint P′ P by blast
  have Q3: P′ = P ⟶ M Midpoint P′ P
    using assms(1) by auto
  have Q4: A B Perp P′ P ∨ P′ = P
    using ReflectL-def assms(2) by auto
  then have M Midpoint P′ P
    using Q2 Q3 by blast
  thus ?thesis
    by (simp add: ReflectLAt-def Q4 assms(3))
qed

lemma image-in-col:
  assumes Y ReflectLAt P P′ A B
  shows Col P P′ Y
  using Col-perm ReflectLAt-def assms midpoint-col by blast

lemma is-image-spec-rev:
  assumes P P′ ReflectL A B
  shows P P′ ReflectL B A
proof −
  obtain M0 where P1: M0 Midpoint P′ P ∧ Col A B M0
    using ReflectL-def assms by blast
  have P2: Col B A M0
    using Col-cases P1 by blast
  have A B Perp P′ P ∨ P′ = P
    using ReflectL-def assms by blast
  thus ?thesis
```

using *P1 P2 Perp-cases ReflectL-def* **by** *auto*
qed

lemma *is-image-rev*:
assumes *P P′ Reflect A B*
shows *P P′ Reflect B A*
using *Reflect-def assms is-image-spec-rev* **by** *auto*

lemma *midpoint-preserves-per*:
assumes *Per A B C* **and**
M Midpoint A A1 **and**
M Midpoint B B1 **and**
M Midpoint C C1
shows *Per A1 B1 C1*
proof −
obtain *C′* **where** *P1*: *B Midpoint C C′* $\wedge$ *Cong A C A C′*
using *Per-def assms*(*1*) **by** *blast*
obtain *C1′* **where** *P2*: *M Midpoint C′ C1′*
using *symmetric-point-construction* **by** *blast*
thus *?thesis*
by (*meson P1 Per-def assms*(*2*) *assms*(*3*) *assms*(*4*) *l7-16 symmetry-preserves-midpoint*)
qed

lemma *col--image-spec*:
assumes *Col A B X*
shows *X X ReflectL A B*
by (*simp add*: *assms col--refl col-permutation-2*)

lemma *image-triv*:
A A Reflect A B
by (*simp add*: *Reflect-def col--refl col-trivial-1 l7-3-2*)

lemma *cong-midpoint--image*:
assumes *Cong A X A Y* **and**
B Midpoint X Y
shows *Y X Reflect A B*
proof *cases*
assume $A = B$
thus *?thesis*
by (*simp add*: *Reflect-def assms*(*2*))
next
assume *S0*: $A \neq B$
{
assume *S1*: $X \neq Y$
then have *X Y Perp A B*
proof −
have *T1*: $B \neq X$
using *S1 assms*(*2*) *midpoint-distinct-1* **by** *blast*
have *T2*: $B \neq Y$
using *S1 assms*(*2*) *midpoint-not-midpoint* **by** *blast*
have *Per A B X*
using *Per-def assms*(*1*) *assms*(*2*) **by** *auto*
thus *?thesis*
using *S0 S1 T1 T2 assms*(*2*) *col-per-perp midpoint-col* **by** *auto*
qed
then have *A B Perp X Y* $\vee$ $X = Y$
using *Perp-perm* **by** *blast*
then have *Y X Reflect A B*
using *ReflectL-def S0 assms*(*2*) *col-trivial-2 is-image-is-image-spec* **by** *blast*
}
then have $X \neq Y \longrightarrow$ *Y X Reflect A B* **by** *blast*
thus *?thesis*
using *assms*(*2*) *image-triv is-image-rev l7-3* **by** *blast*
qed

lemma *col-image-spec--eq*:
  assumes *Col A B P* **and**
    P P′ ReflectL A B
  shows *P = P′*
  using *assms(1) assms(2) col--image-spec l10-2-uniqueness-spec l10-4-spec* **by** *blast*

lemma *image-spec-triv*:
  A A ReflectL B B
  using *col--image-spec not-col-distincts* **by** *blast*

lemma *image-spec--eq*:
  assumes *P P′ ReflectL A A*
  shows *P = P′*
  using *assms col-image-spec--eq not-col-distincts* **by** *blast*

lemma *image--midpoint*:
  assumes *P P′ Reflect A A*
  shows *A Midpoint P′ P*
  using *Reflect-def assms* **by** *auto*

lemma *is-image-spec-dec*:
  A B ReflectL C D ∨ ¬ A B ReflectL C D
  by *simp*

lemma *l10-14*:
  assumes *P ≠ P′* **and**
    A ≠ B **and**
    P P′ Reflect A B
  shows *A B TS P P′*
proof −
  have *P1*: *P P′ ReflectL A B*
    using *assms(2) assms(3) is-image-is-image-spec* **by** *blast*
  then obtain *M0* **where** *M0 Midpoint P′ P ∧ Col A B M0*
    using *ReflectL-def* **by** *blast*
  then have *A B Perp P′ P ⟶ A B TS P P′*
    by (*meson TS-def assms(1) assms(2) assms(3) between-symmetry col-permutation-2 local.image-id midpoint-bet osym-not-col*)
  thus *?thesis*
    using *assms(1) P1 ReflectL-def* **by** *blast*
qed

lemma *l10-15*:
  assumes *Col A B C* **and**
    ¬ Col A B P
  shows *∃ Q. A B Perp Q C ∧ A B OS P Q*
proof −
  have *P1*: *A ≠ B*
    using *assms(2) col-trivial-1* **by** *auto*
  obtain *X* **where** *P2*: *A B TS P X*
    using *assms(2) col-permutation-1 l9-10* **by** *blast*
  {
    assume *Q1*: *A = C*
    obtain *Q* **where** *Q2*: *∃ T. A B Perp Q A ∧ Col A B T ∧ Bet X T Q*
      using *P1 l8-21* **by** *blast*
    then obtain *T* **where** *A B Perp Q A ∧ Col A B T ∧ Bet X T Q* **by** *blast*
    then have *A B TS Q X*
      by (*meson P2 TS-def between-symmetry col-permutation-2 perp-not-col*)
    then have *Q5*: *A B OS P Q*
      using *P2 l9-8-1* **by** *blast*
    then have *∃ Q. A B Perp Q C ∧ A B OS P Q*
      using *Q1 Q2* **by** *blast*
  }
  then have *P3*: *A = C ⟶ (∃ Q. A B Perp Q C ∧ A B OS P Q)* **by** *blast*
  {
    assume *Q1*: *A ≠ C*
    then obtain *Q* **where** *Q2*: *∃ T. C A Perp Q C ∧ Col C A T ∧ Bet X T Q*

```
      using l8-21 by presburger
    then obtain T where Q3: C A Perp Q C ∧ Col C A T ∧ Bet X T Q by blast
    have Q4: A B Perp Q C
      using NCol-perm P1 Q2 assms(1) col-trivial-2 perp-col2 by blast
    have A B TS Q X
    proof −
      have R1: ¬ Col Q A B
        using Col-perm P1 Q2 assms(1) col-trivial-2 colx perp-not-col by blast
      have R2: ¬ Col X A B
        using P2 TS-def by auto
      have R3: Col T A B
        by (metis Q1 Q3 assms(1) col-trivial-2 colx not-col-permutation-1)
      have Bet Q T X
        using Bet-cases Q3 by blast
      then have ∃ T. Col T A B ∧ Bet Q T X
        using R3 by blast
      thus ?thesis using R1 R2
        by (simp add: TS-def)
    qed
    then have A B OS P Q
      using P2 l9-8-1 by blast
    then have ∃ Q. A B Perp Q C ∧ A B OS P Q
      using Q4 by blast
  }
  thus ?thesis using P3 by blast
qed

lemma ex-per-cong:
  assumes A ≠ B and
    X ≠ Y and
    Col A B C and
    ¬ Col A B D
  shows ∃ P. Per P C A ∧ Cong P C X Y ∧ A B OS P D
proof −
  obtain Q where P1: A B Perp Q C ∧ A B OS D Q
    using assms(3) assms(4) l10-15 by blast
  obtain P where P2: C Out Q P ∧ Cong C P X Y
    by (metis P1 assms(2) perp-not-eq-2 segment-construction-3)
  have P3: Per P C A
    using P1 P2 assms(3) col-trivial-3 l8-16-1 l8-3 out-col by blast
  have A B OS P D
    using P1 P2 assms(3) one-side-symmetry os-out-os by blast
  thus ?thesis
    using P2 P3 cong-left-commutativity by blast
qed

lemma exists-cong-per:
  ∃ C. Per A B C ∧ Cong B C X Y
proof cases
  assume A = B
  thus ?thesis
    by (meson Tarski-neutral-dimensionless.l8-5 Tarski-neutral-dimensionless-axioms l8-2 segment-construction)
next
  assume A ≠ B
  thus ?thesis
    by (metis Perp-perm bet-col between-trivial l8-16-1 l8-21 segment-construction)
qed
```

3.9.2 Upper dim 2

```
lemma upper-dim-implies-per2--col:
  assumes upper-dim-axiom
  shows ∀ A B C X. (Per A X C ∧ X ≠ C ∧ Per B X C) ⟶ Col A B X
proof −
  {
    fix A B C X
```

assume *Per A X C* ∧ *X* ≠ *C* ∧ *Per B X C*
moreover then obtain *C'* **where** *X Midpoint C C'* ∧ *Cong A C A C'*
using *Per-def* **by** *blast*
ultimately have *Col A B X*
by (*smt Col-def assms midpoint-cong midpoint-distinct-2 not-cong-2134 per-double-cong upper-dim-axiom-def*)
}
then show *?thesis* **by** *blast*
qed

lemma *upper-dim-implies-col-perp2--col*:
assumes *upper-dim-axiom*
shows ∀ *A B X Y P*. (*Col A B P* ∧ *A B Perp X P* ∧ *P A Perp Y P*) ⟶ *Col Y X P*
proof −
{
fix *A B X Y P*
assume *H1*: *Col A B P* ∧ *A B Perp X P* ∧ *P A Perp Y P*
then have *H2*: *P* ≠ *A*
using *perp-not-eq-1* **by** *blast*
have *Col Y X P*
proof −
have *T1*: *Per Y P A*
using *H1 l8-2 perp-per-1* **by** *blast*
moreover have *Per X P A*
using *H1 col-trivial-3 l8-16-1* **by** *blast*
then show *?thesis* **using** *T1 H2*
using *assms upper-dim-implies-per2--col* **by** *blast*
qed
}
then show *?thesis* **by** *blast*
qed

lemma *upper-dim-implies-perp2--col*:
assumes *upper-dim-axiom*
shows ∀ *X Y Z A B*. (*X Y Perp A B* ∧ *X Z Perp A B*) ⟶ *Col X Y Z*
proof −
{
fix *X Y Z A B*
assume *H1*: *X Y Perp A B* ∧ *X Z Perp A B*
then have *H1A*: *X Y Perp A B* **by** *blast*
have *H1B*: *X Z Perp A B* **using** *H1* **by** *blast*
obtain *C* **where** *H2*: *C PerpAt X Y A B*
using *H1 Perp-def* **by** *blast*
obtain *C'* **where** *H3*: *C' PerpAt X Z A B*
using *H1 Perp-def* **by** *blast*
have *Col X Y Z*
proof *cases*
assume *H2*: *Col A B X*
{
assume *X* = *A*
then have *Col X Y Z* **using** *upper-dim-implies-col-perp2--col*
by (*metis H1 H2 Perp-cases assms col-permutation-1*)
}
then have *P1*: *X* = *A* ⟶ *Col X Y Z* **by** *blast*
{
assume *P2*: *X* ≠ *A*
then have *P3*: *A B Perp X Y* **using** *perp-sym*
using *H1 Perp-perm* **by** *blast*
have *Col A B X*
by (*simp add*: *H2*)
then have *P4*: *A X Perp X Y* **using** *perp-col*
using *P2 P3* **by** *auto*
have *P5*: *A X Perp X Z*
by (*metis H1 H2 P2 Perp-perm col-trivial-3 perp-col0*)
have *P6*: *Col Y Z X*
proof −
have *Q1*: *upper-dim-axiom*

```
          by (simp add: assms)
        have Q2: Per Y X A
          using P4 Perp-cases perp-per-2 by blast
        have Per Z X A
          by (meson P5 Tarski-neutral-dimensionless.Perp-cases Tarski-neutral-dimensionless-axioms perp-per-2)
        then show ?thesis using Q1 Q2 P2
          using upper-dim-implies-per2--col by blast
      qed
      then have Col X Y Z
        using Col-perm by blast
    }
    then show ?thesis
      using P1 by blast
  next
    assume T1: ¬ Col A B X
    obtain Y0 where Q3: Y0 PerpAt X Y A B
      using H1 Perp-def by blast
    obtain Z0 where Q4: Z0 PerpAt X Z A B
      using Perp-def H1 by blast
    have Q5: X Y0 Perp A B
    proof −
      have R1: X ≠ Y0
        using Q3 T1 perp-in-col by blast
      have R2: X Y Perp A B
        by (simp add: H1A)
      then show ?thesis using R1
        using Q3 perp-col perp-in-col by blast
    qed
    have X Z0 Perp A B
      by (metis H1B Q4 T1 perp-col perp-in-col)
    then have Q7: Y0 = Z0
    by (meson Q3 Q4 Q5 T1 Tarski-neutral-dimensionless.Perp-perm Tarski-neutral-dimensionless-axioms l8-18-uniqueness
perp-in-col)
    have Col X Y Z
    proof −
      have X ≠ Y0
        using Q5 perp-distinct by auto
      moreover have Col X Y0 Y
        using Q3 not-col-permutation-5 perp-in-col by blast
      moreover have Col X Y0 Z
        using Q4 Q7 col-permutation-5 perp-in-col by blast
      ultimately show ?thesis
        using col-transitivity-1 by blast
    qed
    then show ?thesis using l8-18-uniqueness
      by (smt H1 H2 Perp-cases T1 col-trivial-3 perp-col1 perp-in-col perp-not-col)
  qed
 }
 then show ?thesis by blast
qed

lemma upper-dim-implies-not-two-sides-one-side-aux:
 assumes upper-dim-axiom
 shows ∀ A B X Y PX. (A ≠ B ∧ PX ≠ A ∧ A B Perp X PX ∧ Col A B PX ∧ ¬ Col X A B ∧ ¬ Col Y A B ∧ ¬
A B TS X Y) ⟶ A B OS X Y
proof −
 {
  fix A B X Y PX
  assume H1: A ≠ B ∧ PX ≠ A ∧ A B Perp X PX ∧ Col A B PX ∧ ¬ Col X A B ∧ ¬ Col Y A B ∧ ¬ A B TS X Y
  have H1A: A ≠ B using H1 by simp
  have H1B: PX ≠ A using H1 by simp
  have H1C: A B Perp X PX using H1 by simp
  have H1D: Col A B PX using H1 by simp
  have H1E: ¬ Col X A B using H1 by simp
  have H1F: ¬ Col Y A B using H1 by simp
  have H1G: ¬ A B TS X Y using H1 by simp
```

have ∃ *P T*. *PX A Perp P PX* ∧ *Col PX A T* ∧ *Bet Y T P*
using *H1B l8-21* **by** *blast*
then obtain *P T* **where** *T1*: *PX A Perp P PX* ∧ *Col PX A T* ∧ *Bet Y T P*
by *blast*
have *J1*: *PX A Perp P PX* **using** *T1* **by** *blast*
have *J2*: *Col PX A T* **using** *T1* **by** *blast*
have *J3*: *Bet Y T P* **using** *T1* **by** *blast*
have *P9*: *Col P X PX* **using** *upper-dim-implies-col-perp2--col*
using *H1C H1D J1 assms* **by** *blast*
have *J4*: ¬ *Col P A B*
using *H1A H1D T1 col-trivial-2 colx not-col-permutation-3 perp-not-col* **by** *blast*
have *J5*: *PX A TS P Y*
proof −
have *f1*: *Col PX A B*
using *H1D not-col-permutation-1* **by** *blast*
then have *f2*: *Col B PX A*
using *not-col-permutation-1* **by** *blast*
have *f3*: ∀ *p*. (*T* = *A* ∨ *Col p A PX*) ∨ ¬ *Col p A T*
by (*metis J2 l6-16-1*)
have *f4*: *Col T PX A*
using *J2 not-col-permutation-1* **by** *blast*
have *f5*: ∀ *p*. *Col p PX B* ∨ ¬ *Col p PX A*
using *f2* **by** (*meson H1B l6-16-1*)
have *f6*: ∀ *p*. (*B* = *PX* ∨ *Col p B A*) ∨ ¬ *Col p B PX*
using *H1D l6-16-1* **by** *blast*
have *f7*: ∀ *p pa*. ((*B* = *PX* ∨ *Col p PX pa*) ∨ ¬ *Col p PX B*) ∨ ¬ *Col pa PX A*
using *f5* **by** (*metis l6-16-1*)
have *f8*: ∀ *p*. ((*T* = *A* ∨ *B* = *PX*) ∨ *Col p A B*) ∨ ¬ *Col p A PX*
using *f2* **by** (*metis H1B l6-16-1 not-col-permutation-1*)
have *Col B T PX*
using *f5 f4 not-col-permutation-1* **by** *blast*
then have *f9*: ∀ *p*. (*T* = *PX* ∨ *Col p T B*) ∨ ¬ *Col p T PX*
using *l6-16-1* **by** *blast*
have *B* = *PX* ⟶ ¬ *Col Y PX A* ∧ ¬ *Col P PX A*
using *f1* **by** (*metis* (*no-types*) *H1B H1F J4 l6-16-1 not-col-permutation-1*)
then show *?thesis*
using *f9 f8 f7 f6 f5 f4 f3* **by** (*metis* (*no-types*) *H1B H1F J3 J4 TS-def l9-2 not-col-permutation-1*)
qed
have *J6*: *X* ≠ *PX*
using *H1 perp-not-eq-2* **by** *blast*
have *J7*: *P* ≠ *X*
using *H1A H1D H1G J5 col-preserves-two-sides col-trivial-2 not-col-permutation-1* **by** *blast*
have *J8*: *Bet X PX P* ∨ *PX Out X P* ∨ ¬ *Col X PX P*
using *l6-4-2* **by** *blast*
have *J9*: *PX A TS P X*
by (*metis H1A H1D H1G J5 J6 J8 Out-cases P9 TS-def bet--ts between-symmetry col-permutation-1 col-preserves-two-sides col-trivial-2 l9-5*)
then have *A B OS X Y*
by (*meson H1A H1D J5 col2-os--os col-trivial-2 l9-2 l9-8-1 not-col-permutation-1*)
}
then show *?thesis* **by** *blast*
qed

lemma *upper-dim-implies-not-two-sides-one-side*:
assumes *upper-dim-axiom*
shows ∀ *A B X Y*. (¬ *Col X A B* ∧ ¬ *Col Y A B* ∧ ¬ *A B TS X Y*) ⟶ *A B OS X Y*
proof −
{
fix *A B X Y*
assume *H1*: ¬ *Col X A B* ∧ ¬ *Col Y A B* ∧ ¬ *A B TS X Y*
have *H1A*: ¬ *Col X A B* **using** *H1* **by** *simp*
have *H1B*: ¬ *Col Y A B* **using** *H1* **by** *simp*
have *H1C*: ¬ *A B TS X Y* **using** *H1* **by** *simp*
have *P1*: *A* ≠ *B*
using *H1A col-trivial-2* **by** *blast*
obtain *PX* **where** *P2*: *Col A B PX* ∧ *A B Perp X PX*

```
      using Col-cases H1 l8-18-existence by blast
    have A B OS X Y
    proof cases
      assume H5: PX = A
      have B A OS X Y
      proof -
        have F1: B A Perp X A
          using P2 Perp-perm H5 by blast
        have F2: Col B A A
          using not-col-distincts by blast
        have F3: ¬ Col X B A
          using Col-cases H1A by blast
        have F4: ¬ Col Y B A
          using Col-cases H1B by blast
        have ¬ B A TS X Y
          using H1C invert-two-sides by blast
        then show ?thesis
          by (metis F1 F3 F4 assms col-trivial-2 upper-dim-implies-not-two-sides-one-side-aux)
      qed
      then show ?thesis
        by (simp add: invert-one-side)
    next
      assume PX ≠ A
      then show ?thesis
        using H1 P1 P2 assms upper-dim-implies-not-two-sides-one-side-aux by blast
    qed
  }
  then show ?thesis by blast
qed

lemma upper-dim-implies-not-one-side-two-sides:
  assumes upper-dim-axiom
  shows ∀ A B X Y. (¬ Col X A B ∧ ¬ Col Y A B ∧ ¬ A B OS X Y) ⟶ A B TS X Y
  using assms upper-dim-implies-not-two-sides-one-side by blast

lemma upper-dim-implies-one-or-two-sides:
  assumes upper-dim-axiom
  shows ∀ A B X Y. (¬ Col X A B ∧ ¬ Col Y A B) ⟶ (A B TS X Y ∨ A B OS X Y)
  using assms upper-dim-implies-not-two-sides-one-side by blast

lemma upper-dim-implies-all-coplanar:
  assumes upper-dim-axiom
  shows all-coplanar-axiom
  using all-coplanar-axiom-def assms upper-dim-axiom-def by auto

lemma all-coplanar-implies-upper-dim:
  assumes all-coplanar-axiom
  shows upper-dim-axiom
  using all-coplanar-axiom-def assms upper-dim-axiom-def by auto

lemma all-coplanar-upper-dim:
  shows all-coplanar-axiom ⟷ upper-dim-axiom
  using all-coplanar-implies-upper-dim upper-dim-implies-all-coplanar by auto

lemma upper-dim-stab:
  shows ¬ ¬ upper-dim-axiom ⟶ upper-dim-axiom by blast

lemma cop--cong-on-bissect:
  assumes Coplanar A B X P and
    M Midpoint A B and
    M PerpAt A B P M and
    Cong X A X B
  shows Col M P X
proof -
  have P1: X = M ∨ ¬ Col A B X ∧ M PerpAt X M A B
    using assms(2) assms(3) assms(4) cong-commutativity cong-perp-or-mid perp-in-distinct by blast
```

{
assume *H1*: ¬ *Col A B X* ∧ *M PerpAt X M A B*
then have *Q1*: *X M Perp A B*
using *perp-in-perp* **by** *blast*
have *Q2*: *A B Perp P M*
using *assms*(*3*) *perp-in-perp* **by** *blast*
have *P2*: *Col M A B*
by (*simp add*: *assms*(*2*) *midpoint-col*)
then have *Col M P X* **using** *cop-perp2--col*
by (*meson Perp-perm Q1 Q2 assms*(*1*) *coplanar-perm-1*)
}
then show *?thesis*
using *P1 not-col-distincts* **by** *blast*
qed

lemma *cong-cop-mid-perp--col*:
assumes *Coplanar A B X P* **and**
Cong A X B X **and**
M Midpoint A B **and**
A B Perp P M
shows *Col M P X*
proof −
have *M PerpAt A B P M*
using *Col-perm assms*(*3*) *assms*(*4*) *bet-col l8-15-1 midpoint-bet* **by** *blast*
then show *?thesis*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *cop--cong-on-bissect not-cong-2143* **by** *blast*
qed

lemma *cop-image-in2--col*:
assumes *Coplanar A B P Q* **and**
M ReflectLAt P P′ A B **and**
M ReflectLAt Q Q′ A B
shows *Col M P Q*
proof −
have *P1*: *P P′ ReflectL A B*
using *assms*(*2*) *image-in-is-image-spec* **by** *auto*
then have *P2*: *A B Perp P′ P* ∨ *P′* = *P*
using *ReflectL-def* **by** *auto*
have *P3*: *Q Q′ ReflectL A B*
using *assms*(*3*) *image-in-is-image-spec* **by** *blast*
then have *P4*: *A B Perp Q′ Q* ∨ *Q′* = *Q*
using *ReflectL-def* **by** *auto*
{
assume *S1*: *A B Perp P′ P* ∧ *A B Perp Q′ Q*
{
assume *T1*: *A* = *M*
have *T2*: *Per B A P*
by (*metis P1 Perp-perm S1 T1 assms*(*2*) *image-in-col is-image-is-image-spec l10-14 perp-col1 perp-distinct perp-per-1 ts-distincts*)
have *T3*: *Per B A Q*
by (*metis S1 T1 assms*(*3*) *image-in-col l8-5 perp-col1 perp-per-1 perp-right-comm*)
have *T4*: *Coplanar B P Q A*
using *assms*(*1*) *ncoplanar-perm-18* **by** *blast*
have *T5*: *B* ≠ *A*
using *S1 perp-distinct* **by** *blast*
have *T6*: *Per P A B*
by (*simp add*: *T2 l8-2*)
have *T7*: *Per Q A B*
using *Per-cases T3* **by** *blast*
then have *Col P Q A* **using** *T4 T5 T6*
using *cop-per2--col* **by** *blast*
then have *Col A P Q*
using *not-col-permutation-1* **by** *blast*
then have *Col M P Q*
using *T1* **by** *blast*
}

```
    then have S2: A = M ⟶ Col M P Q by blast
    {
      assume D0: A ≠ M
      have D1: Per A M P
      proof -
        have E1: M Midpoint P P′
          using ReflectLAt-def assms(2) l7-2 by blast
        have Cong P A P′ A
          using P1 col-trivial-3 is-image-spec-col-cong by blast
        then have Cong A P A P′
          using Cong-perm by blast
        then show ?thesis
          using E1 Per-def by blast
      qed
      have D2: Per A M Q
      proof -
        have E2: M Midpoint Q Q′
          using ReflectLAt-def assms(3) l7-2 by blast
        have Cong A Q A Q′
          using P3 col-trivial-3 cong-commutativity is-image-spec-col-cong by blast
        then show ?thesis
          using E2 Per-def by blast
      qed
      have Col P Q M
      proof -
        have W1: Coplanar P Q A B
          using assms(1) ncoplanar-perm-16 by blast
        have W2: A ≠ B
          using S1 perp-distinct by blast
        have Col A B M
          using ReflectLAt-def assms(2) by blast
        then have Coplanar P Q A M
          using W1 W2 col2-cop--cop col-trivial-3 by blast
        then have V1: Coplanar A P Q M
          using ncoplanar-perm-8 by blast
        have V3: Per P M A
          by (simp add: D1 l8-2)
        have Per Q M A
          using D2 Per-perm by blast
        then show ?thesis
          using V1 D0 V3 cop-per2--col by blast
      qed
      then have Col M P Q
        using Col-perm by blast
    }
    then have A ≠ M ⟶ Col M P Q by blast
    then have Col M P Q
      using S2 by blast
  }
  then have P5: (A B Perp P′ P ∧ A B Perp Q′ Q) ⟶ Col M P Q by blast
  have P6: (A B Perp P′ P ∧ (Q′ = Q)) ⟶ Col M P Q
    using ReflectLAt-def assms(3) l7-3 not-col-distincts by blast
  have P7: (P′ = P ∧ A B Perp Q′ Q) ⟶ Col M P Q
    using ReflectLAt-def assms(2) l7-3 not-col-distincts by blast
  have (P′ = P ∧ Q′ = Q) ⟶ Col M P Q
    using ReflectLAt-def assms(3) col-trivial-3 l7-3 by blast
  then show ?thesis
    using P2 P4 P5 P6 P7 by blast
qed

lemma l10-10-spec:
  assumes P′ P ReflectL A B and
    Q′ Q ReflectL A B
  shows Cong P Q P′ Q′
proof cases
  assume A = B
```

then show *?thesis*
using *assms(1) assms(2) cong-reflexivity image-spec--eq* **by** *blast*
next
assume *H1*: $A \neq B$
obtain X **where** *P1*: X *Midpoint* $P\ P' \wedge$ *Col* $A\ B\ X$
using *ReflectL-def assms(1)* **by** *blast*
obtain Y **where** *P2*: Y *Midpoint* $Q\ Q' \wedge$ *Col* $A\ B\ Y$
using *ReflectL-def assms(2)* **by** *blast*
obtain Z **where** *P3*: Z *Midpoint* $X\ Y$
using *midpoint-existence* **by** *blast*
have *P4*: *Col* $A\ B\ Z$
proof *cases*
assume $X = Y$
then show *?thesis*
by (*metis P2 P3 midpoint-distinct-3*)
next
assume $X \neq Y$
then show *?thesis*
by (*metis P1 P2 P3 l6-21 midpoint-col not-col-distincts*)
qed
obtain R **where** *P5*: Z *Midpoint* $P\ R$
using *symmetric-point-construction* **by** *blast*
obtain R' **where** *P6*: Z *Midpoint* $P'\ R'$
using *symmetric-point-construction* **by** *blast*
have *P7*: $A\ B$ *Perp* $P\ P' \vee P = P'$
using *ReflectL-def assms(1)* **by** *auto*
have *P8*: $A\ B$ *Perp* $Q\ Q' \vee Q = Q'$
using *ReflectL-def assms(2)* **by** *blast*
{
assume *Q1*: $A\ B$ *Perp* $P\ P' \wedge A\ B$ *Perp* $Q\ Q'$
have *Q2*: $R\ R'$ *ReflectL* $A\ B$
proof −
have $P\ P'$ *Reflect* $A\ B$
by (*simp add*: *H1 assms(1) is-image-is-image-spec l10-4-spec*)
then have $R\ R'$ *Reflect* $A\ B$
using *H1 P4 P5 P6 midpoint-preserves-image* **by** *blast*
then show *?thesis*
using *H1 is-image-is-image-spec* **by** *blast*
qed
have *Q3*: $R \neq R'$
using *P5 P6 Q1 l7-9 perp-not-eq-2* **by** *blast*
have *Q4*: Y *Midpoint* $R\ R'$
using *P1 P3 P5 P6 symmetry-preserves-midpoint* **by** *blast*
have *Q5*: *Cong* $Q'\ R'\ Q\ R$
using *P2 Q4 l7-13* **by** *blast*
have *Q6*: *Cong* $P'\ Z\ P\ Z$
using *P4 assms(1) is-image-spec-col-cong* **by** *auto*
have *Q7*: *Cong* $Q'\ Z\ Q\ Z$
using *P4 assms(2) is-image-spec-col-cong* **by** *blast*
then have *Cong* $P\ Q\ P'\ Q'$
proof −
have *S1*: *Cong* $R\ Z\ R'\ Z$
using *P5 P6 Q6 cong-symmetry l7-16 l7-3-2* **by** *blast*
have $R \neq Z$
using *Q3 S1 cong-reverse-identity* **by** *blast*
then show *?thesis*
by (*meson P5 P6 Q5 Q6 Q7 S1 between-symmetry five-segment midpoint-bet not-cong-2143 not-cong-3412*)
qed
}
then have *P9*: $(A\ B$ *Perp* $P\ P' \wedge A\ B$ *Perp* $Q\ Q') \longrightarrow$ *Cong* $P\ Q\ P'\ Q'$ **by** *blast*
have *P10*: $(A\ B$ *Perp* $P\ P' \wedge Q = Q') \longrightarrow$ *Cong* $P\ Q\ P'\ Q'$
using *P2 Tarski-neutral-dimensionless.l7-3 Tarski-neutral-dimensionless-axioms assms(1) cong-symmetry is-image-spec-col-cong* **by** *fastforce*
have *P11*: $(P = P' \wedge A\ B$ *Perp* $Q\ Q') \longrightarrow$ *Cong* $P\ Q\ P'\ Q'$
using *P1 Tarski-neutral-dimensionless.l7-3 Tarski-neutral-dimensionless.not-cong-4321 Tarski-neutral-dimensionless-axioms assms(2) is-image-spec-col-cong* **by** *fastforce*

have $(P = P' \wedge Q = Q') \longrightarrow$ *Cong P Q P' Q'*
  using *cong-reflexivity* **by** *blast*
then show *?thesis*
  using *P10 P11 P7 P8 P9* **by** *blast*
qed

lemma *l10-10*:
assumes *P' P Reflect A B* **and**
  Q' Q Reflect A B
shows *Cong P Q P' Q'*
using *Reflect-def assms(1) assms(2) cong-4321 l10-10-spec l7-13* **by** *auto*

lemma *image-preserves-bet*:
assumes *A A' ReflectL X Y* **and**
  B B' ReflectL X Y **and**
  C C' ReflectL X Y **and**
  Bet A B C
shows *Bet A' B' C'*
proof −
have *P3*: *A B C Cong3 A' B' C'*
  using *Cong3-def assms(1) assms(2) assms(3) l10-10-spec l10-4-spec* **by** *blast*
then show *?thesis*
  using *assms(4) l4-6* **by** *blast*
qed

lemma *image-gen-preserves-bet*:
assumes *A A' Reflect X Y* **and**
  B B' Reflect X Y **and**
  C C' Reflect X Y **and**
  Bet A B C
shows *Bet A' B' C'*
proof *cases*
assume $X = Y$
then show *?thesis*
  by *(metis (full-types) assms(1) assms(2) assms(3) assms(4) image--midpoint l7-15 l7-2)*
next
assume *P1*: $X \neq Y$
then have *P2*: *A A' ReflectL X Y*
  using *assms(1) is-image-is-image-spec* **by** *blast*
have *P3*: *B B' ReflectL X Y*
  using *P1 assms(2) is-image-is-image-spec* **by** *auto*
have *C C' ReflectL X Y*
  using *P1 assms(3) is-image-is-image-spec* **by** *blast*
then show *?thesis* **using** *image-preserves-bet*
  using *assms(4) P2 P3* **by** *blast*
qed

lemma *image-preserves-col*:
assumes *A A' ReflectL X Y* **and**
  B B' ReflectL X Y **and**
  C C' ReflectL X Y **and**
  Col A B C
shows *Col A' B' C'* **using** *image-preserves-bet*
using *Col-def assms(1) assms(2) assms(3) assms(4)* **by** *auto*

lemma *image-gen-preserves-col*:
assumes *A A' Reflect X Y* **and**
  B B' Reflect X Y **and**
  C C' Reflect X Y **and**
  Col A B C
shows *Col A' B' C'*
using *Col-def assms(1) assms(2) assms(3) assms(4) image-gen-preserves-bet* **by** *auto*

lemma *image-gen-preserves-ncol*:
assumes *A A' Reflect X Y* **and**
  B B' Reflect X Y **and**

C C′ Reflect X Y **and**
¬ *Col A B C*
shows ¬ *Col A′ B′ C′*
using *assms(1) assms(2) assms(3) assms(4)image-gen-preserves-col l10-4* **by** *blast*

lemma *image-gen-preserves-inter*:
assumes *A A′ Reflect X Y* **and**
B B′ Reflect X Y **and**
C C′ Reflect X Y **and**
D D′ Reflect X Y **and**
¬ *Col A B C* **and**
$C \neq D$ **and**
Col A B I **and**
Col C D I **and**
Col A′ B′ I′ **and**
Col C′ D′ I′
shows *I I′ Reflect X Y*
proof −
obtain *I0* **where** *P1*: *I I0 Reflect X Y*
using *l10-6-existence* **by** *blast*
then show *?thesis*
by (*smt Tarski-neutral-dimensionless.image-gen-preserves-col Tarski-neutral-dimensionless-axioms assms(1) assms(10) assms(2) assms(3) assms(4) assms(5) assms(6) assms(7) assms(8) assms(9) l10-4 l10-7 l6-21*)
qed

lemma *intersection-with-image-gen*:
assumes *A A′ Reflect X Y* **and**
B B′ Reflect X Y **and**
¬ *Col A B A′* **and**
Col A B C **and**
Col A′ B′ C
shows *Col C X Y*
by (*smt assms(1) assms(2) assms(3) assms(4) assms(5) image-gen-preserves-inter l10-2-uniqueness l10-4 l10-8 not-col-distincts*)

lemma *image-preserves-midpoint* :
assumes *A A′ ReflectL X Y* **and**
B B′ ReflectL X Y **and**
C C′ ReflectL X Y **and**
A Midpoint B C
shows *A′ Midpoint B′ C′*
proof −
have *P1*: *Bet B′ A′ C′* **using** *image-preserves-bet*
using *assms(1) assms(2) assms(3) assms(4) midpoint-bet* **by** *auto*
have *Cong B′ A′ A′ C′*
by (*metis Cong-perm Tarski-neutral-dimensionless.l10-10-spec Tarski-neutral-dimensionless.l7-13 Tarski-neutral-dimensionless-axioms assms(1) assms(2) assms(3) assms(4) cong-transitivity l7-3-2*)
then show *?thesis*
by (*simp add*: *Midpoint-def P1*)
qed

lemma *image-spec-preserves-per*:
assumes *A A′ ReflectL X Y* **and**
B B′ ReflectL X Y **and**
C C′ ReflectL X Y **and**
Per A B C
shows *Per A′ B′ C′*
proof *cases*
assume $X = Y$
then show *?thesis*
using *assms(1) assms(2) assms(3) assms(4) image-spec--eq* **by** *blast*
next
assume *P1*: $X \neq Y$
obtain *C1* **where** *P2*: *B Midpoint C C1*
using *symmetric-point-construction* **by** *blast*
obtain *C1′* **where** *P3*: *C1 C1′ ReflectL X Y*

by (*meson P1 l10-6-existence-spec*)
then have *P4*: *B′ Midpoint C′ C1′*
using *P2 assms*(*2*) *assms*(*3*) *image-preserves-midpoint* **by** *blast*
have *Cong A′ C′ A′ C1′*
proof −
have *Q1*: *Cong A′ C′ A C*
using *assms*(*1*) *assms*(*3*) *l10-10-spec* **by** *auto*
have *Cong A C A′ C1′*
by (*metis P2 P3 Tarski-neutral-dimensionless.l10-10-spec Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*4*) *cong-inner-transitivity cong-symmetry per-double-cong*)
then show *?thesis*
using *Q1 cong-transitivity* **by** *blast*
qed
then show *?thesis*
using *P4 Per-def* **by** *blast*
qed

lemma *image-preserves-per*:
assumes *A A′ Reflect X Y* **and**
B B′ Reflect X Y **and**
C C′ Reflect X Y **and**
Per A B C
shows *Per A′ B′ C′*
proof *cases*
assume $X = Y$
then show *?thesis* **using** *midpoint-preserves-per*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *image--midpoint l7-2* **by** *blast*
next
assume *P1*: $X \neq Y$
have *P2*: $X \neq Y \wedge$ *A A′ ReflectL X Y*
using *P1 assms*(*1*) *is-image-is-image-spec* **by** *blast*
have *P3*: $X \neq Y \wedge$ *B B′ ReflectL X Y*
using *P1 assms*(*2*) *is-image-is-image-spec* **by** *blast*
have *P4*: $X \neq Y \wedge$ *C C′ ReflectL X Y*
using *P1 assms*(*3*) *is-image-is-image-spec* **by** *blast*
then show *?thesis* **using** *image-spec-preserves-per*
using *P2 P3 assms*(*4*) **by** *blast*
qed

lemma *l10-12*:
assumes *Per A B C* **and**
Per A′ B′ C′ **and**
Cong A B A′ B′ **and**
Cong B C B′ C′
shows *Cong A C A′ C′*
proof *cases*
assume *P1*: $B = C$
then have $B' = C'$
using *assms*(*4*) *cong-reverse-identity* **by** *blast*
then show *?thesis*
using *P1 assms*(*3*) **by** *blast*
next
assume *P2*: $B \neq C$
have *Cong A C A′ C′*
proof *cases*
assume $A = B$
then show *?thesis*
using *assms*(*3*) *assms*(*4*) *cong-diff-3* **by** *force*
next
assume *P3*: $A \neq B$
obtain *X* **where** *P4*: *X Midpoint B B′*
using *midpoint-existence* **by** *blast*
obtain *A1* **where** *P5*: *X Midpoint A′ A1*
using *Mid-perm symmetric-point-construction* **by** *blast*
obtain *C1* **where** *P6*: *X Midpoint C′ C1*
using *Mid-perm symmetric-point-construction* **by** *blast*

have *Q1*: *A′ B′ C′ Cong3 A1 B C1*
  using *Cong3-def P4 P5 P6 l7-13 l7-2* **by** *blast*
have *Q2*: *Per A1 B C1*
  using *assms*(*2*)*Q1 l8-10* **by** *blast*
have *Q3*: *Cong A B A1 B*
  by (*metis Cong3-def Q1 Tarski-neutral-dimensionless.cong-symmetry Tarski-neutral-dimensionless-axioms assms*(*3*) *cong-inner-transitivity*)
have *Q4*: *Cong B C B C1*
  by (*metis Cong3-def Q1 Tarski-neutral-dimensionless.cong-symmetry Tarski-neutral-dimensionless-axioms assms*(*4*) *cong-inner-transitivity*)
obtain *Y* **where** *P7*: *Y Midpoint C C1*
  using *midpoint-existence* **by** *auto*
then have *R1*: *C1 C Reflect B Y* **using** *cong-midpoint--image*
  using *Q4* **by** *blast*
obtain *A2* **where** *R2*: *A1 A2 Reflect B Y*
  using *l10-6-existence* **by** *blast*
have *R3*: *Cong C A2 C1 A1*
  using *R1 R2 l10-10* **by** *blast*
have *R5*: *B B Reflect B Y*
  using *image-triv* **by** *blast*
have *R6*: *Per A2 B C* **using** *image-preserves-per*
  using *Q2 R1 R2 image-triv* **by** *blast*
have *R7*: *Cong A B A2 B*
  using *l10-10 Cong-perm Q3 R2 cong-transitivity image-triv* **by** *blast*
obtain *Z* **where** *R7A*: *Z Midpoint A A2*
  using *midpoint-existence* **by** *blast*
have *Cong B A B A2*
  using *Cong-perm R7* **by** *blast*
then have *T1*: *A2 A Reflect B Z* **using** *R7A cong-midpoint--image*
  by *blast*
obtain *C0* **where** *T2*: *B Midpoint C C0*
  using *symmetric-point-construction* **by** *blast*
have *T3*: *Cong A C A C0*
  using *T2 assms*(*1*) *per-double-cong* **by** *blast*
have *T4*: *Cong A2 C A2 C0*
  using *R6 T2 per-double-cong* **by** *blast*
have *T5*: *C0 C Reflect B Z*
proof −
  have *C0 C Reflect Z B*
  proof *cases*
    assume *A* = *A2*
    then show *?thesis*
      by (*metis R7A T2 T3 cong-midpoint--image midpoint-distinct-3*)
  next
    assume *A* ≠ *A2*
    then show *?thesis* **using** *l4-17 cong-midpoint--image*
      by (*metis R7A T2 T3 T4 midpoint-col not-col-permutation-3*)
  qed
  then show *?thesis*
    using *is-image-rev* **by** *blast*
qed
have *T6*: *Cong A C A2 C0*
  using *T1 T5 l10-10* **by** *auto*
have *R4*: *Cong A C A2 C*
  by (*metis T4 T6 Tarski-neutral-dimensionless.cong-symmetry Tarski-neutral-dimensionless-axioms cong-inner-transitivity*)
then have *Q5*: *Cong A C A1 C1*
  by (*meson R3 cong-inner-transitivity not-cong-3421*)
then show *?thesis*
  using *Cong3-def Q1 Q5 cong-symmetry cong-transitivity* **by** *blast*
qed
then show *?thesis* **by** *blast*
qed

lemma *l10-16*:
  assumes ¬ *Col A B C* **and**
    ¬ *Col A′ B′ P* **and**

```
    Cong A B A' B'
  shows ∃ C'. A B C Cong3 A' B' C' ∧ A' B' OS P C'
proof cases
  assume A = B
  then show ?thesis
    using assms(1) not-col-distincts by auto
next
  assume P1: A ≠ B
  obtain X where P2: Col A B X ∧ A B Perp C X
    using assms(1) l8-18-existence by blast
  obtain X' where P3: A B X Cong3 A' B' X'
    using P2 assms(3) l4-14 by blast
  obtain Q where P4: A' B' Perp Q X' ∧ A' B' OS P Q
    using P2 P3 assms(2) l10-15 l4-13 by blast
  obtain C' where P5: X' Out C' Q ∧ Cong X' C' X C
    by (metis P2 P4 l6-11-existence perp-distinct)
  have P6: Cong A C A' C'
  proof cases
    assume A = X
    then show ?thesis
      by (metis Cong3-def P3 P5 cong-4321 cong-commutativity cong-diff-3)
  next
    assume A ≠ X
    have P7: Per A X C
      using P2 col-trivial-3 l8-16-1 l8-2 by blast
    have P8: Per A' X' C'
    proof -
      have X' PerpAt A' X' X' C'
      proof -
        have Z1: A' X' Perp X' C'
        proof -
          have W1: X' ≠ C'
            using P5 out-distinct by blast
          have W2: X' Q Perp A' B'
            using P4 Perp-perm by blast
          then have X' C' Perp A' B'
            by (metis P5 Perp-perm W1 col-trivial-3 not-col-permutation-5 out-col perp-col2-bis)
          then show ?thesis
            by (metis Cong3-def P2 P3 Perp-perm ‹A ≠ X› col-trivial-3 cong-identity l4-13 perp-col2-bis)
        qed
        have Z2: Col X' A' X'
          using not-col-distincts by blast
        have Col X' X' C'
          by (simp add: col-trivial-1)
        then show ?thesis
          by (simp add: Z1 Z2 l8-14-2-1b-bis)
      qed
      then show ?thesis
        by (simp add: perp-in-per)
    qed
    have P9: Cong A X A' X'
      using Cong3-def P3 by auto
    have Cong X C X' C'
      using Cong-perm P5 by blast
    then show ?thesis using l10-12
      using P7 P8 P9 by blast
  qed
  have P10: Cong B C B' C'
  proof cases
    assume B = X
    then show ?thesis
      by (metis Cong3-def P3 P5 cong-4321 cong-commutativity cong-diff-3)
  next
    assume B ≠ X
    have Q1: Per B X C
      using P2 col-trivial-2 l8-16-1 l8-2 by blast
```

```
    have X' PerpAt B' X' X' C'
    proof -
      have Q2: B' X' Perp X' C'
      proof -
        have R1: B' ≠ X'
          using Cong3-def P3 ‹B ≠ X› cong-identity by blast
        have X' C' Perp B' A'
        proof -
          have S1: X' ≠ C'
            using Out-def P5 by blast
          have S2: X' Q Perp B' A'
            using P4 Perp-perm by blast
          have Col X' Q C'
            using Col-perm P5 out-col by blast
          then show ?thesis
            using S1 S2 perp-col by blast
        qed
        then have R2: B' A' Perp X' C'
          using Perp-perm by blast
        have R3: Col B' A' X'
          using Col-perm P2 P3 l4-13 by blast
        then show ?thesis
          using R1 R2 perp-col by blast
      qed
      have Q3: Col X' B' X'
        by (simp add: col-trivial-3)
      have Col X' X' C'
        by (simp add: col-trivial-1)
      then show ?thesis using l8-14-2-1b-bis
        using Q2 Q3 by blast
    qed
    then have Q2: Per B' X' C'
      by (simp add: perp-in-per)
    have Q3: Cong B X B' X'
      using Cong3-def P3 by blast
    have Q4: Cong X C X' C'
      using P5 not-cong-3412 by blast
    then show ?thesis
      using Q1 Q2 Q3 l10-12 by blast
  qed
  have P12: A' B' OS C' Q ⟷ X' Out C' Q ∧ ¬ Col A' B' C' using l9-19 l4-13
    by (meson P2 P3 P5 one-side-not-col123 out-one-side-1)
  then have P13: A' B' OS C' Q using l4-13
    by (meson P2 P3 P4 P5 l6-6 one-side-not-col124 out-one-side-1)
  then show ?thesis
    using Cong3-def P10 P4 P6 assms(3) one-side-symmetry one-side-transitivity by blast
qed

lemma cong-cop-image--col:
  assumes P ≠ P' and
    P P' Reflect A B and
    Cong P X P' X and
    Coplanar A B P X
  shows Col A B X
proof -
  have P1: (A ≠ B ∧ P P' ReflectL A B) ∨ (A = B ∧ A Midpoint P' P)
    by (metis assms(2) image--midpoint is-image-is-image-spec)
  {
    assume Q1: A ≠ B ∧ P P' ReflectL A B
    then obtain M where Q2: M Midpoint P' P ∧ Col A B M
      using ReflectL-def by blast
    have Col A B X
    proof cases
      assume R1: A = M
      have R2: P A Perp A B
      proof -
```

```
      have S1: P ≠ A
        using Q2 R1 assms(1) midpoint-distinct-2 by blast
      have S2: P P' Perp A B
        using Perp-perm Q1 ReflectL-def assms(1) by blast
      have Col P P' A
        using Q2 R1 midpoint-col not-col-permutation-3 by blast
      then show ?thesis using perp-col
        using S1 S2 by blast
    qed
    have R3: Per P A B
      by (simp add: R2 perp-comm perp-per-1)
    then have R3A: Per B A P using l8-2
      by blast
    have A Midpoint P P' ∧ Cong X P X P'
      using Cong-cases Q2 R1 assms(3) l7-2 by blast
    then have R4: Per X A P
      using Per-def by blast
    have R5: Coplanar P B X A
      using assms(4) ncoplanar-perm-20 by blast
    have P ≠ A
      using R2 perp-not-eq-1 by auto
    then show ?thesis using R4 R5 R3A
      using cop-per2--col not-col-permutation-1 by blast
  next
    assume T1: A ≠ M
    have T3: P ≠ M
      using Q2 assms(1) l7-3-2 sym-preserve-diff by blast
    have T2: P M Perp M A
    proof -
      have T4: P P' Perp M A
        using Perp-perm Q1 Q2 ReflectL-def T1 assms(1) col-trivial-3 perp-col0 by blast
      have Col P P' M
        by (simp add: Col-perm Q2 midpoint-col)
      then show ?thesis using T3 T4 perp-col by blast
    qed
    then have M P Perp A M
      using perp-comm by blast
    then have M PerpAt M P A M
      using perp-perp-in by blast
    then have M PerpAt P M M A
      by (simp add: perp-in-comm)
    then have U1: Per P M A
      by (simp add: perp-in-per)
    have U2: Per X M P using l7-2 cong-commutativity
      using Per-def Q2 assms(3) by blast
    have Col A X M
    proof -
      have W2: Coplanar P A X M
        by (meson Q1 Q2 assms(4) col-cop2--cop coplanar-perm-13 ncop-distincts)
      have Per A M P
        by (simp add: U1 l8-2)
      then show ?thesis using cop-per2--col
        using U2 T3 W2 by blast
    qed
    then show ?thesis
      using Q2 T1 col2--eq not-col-permutation-4 by blast
  qed
}
then have P2: (A ≠ B ∧ P P' ReflectL A B) ⟶ Col A B X by blast
have (A = B ∧ A Midpoint P' P) ⟶ Col A B X
  using col-trivial-1 by blast
then show ?thesis using P1 P2 by blast
qed

lemma cong-cop-per2-1:
  assumes A ≠ B and
```

Per A B X **and**
Per A B Y **and**
Cong B X B Y **and**
Coplanar A B X Y
shows *X = Y ∨ B Midpoint X Y*
by (*meson Per-cases assms(1) assms(2) assms(3) assms(4) assms(5) cop-per2--col coplanar-perm-3 l7-20-bis not-col-permutation-5*)

lemma *cong-cop-per2*:
assumes *A ≠ B* **and**
Per A B X **and**
Per A B Y **and**
Cong B X B Y **and**
Coplanar A B X Y
shows *X = Y ∨ X Y ReflectL A B*
proof −
have *X = Y ∨ B Midpoint X Y*
using *assms(1) assms(2) assms(3) assms(4) assms(5) cong-cop-per2-1* **by** *blast*
then show *?thesis*
by (*metis Mid-perm Per-def Reflect-def assms(1) assms(3) cong-midpoint--image symmetric-point-uniqueness*)
qed

lemma *cong-cop-per2-gen*:
assumes *A ≠ B* **and**
Per A B X **and**
Per A B Y **and**
Cong B X B Y **and**
Coplanar A B X Y
shows *X = Y ∨ X Y Reflect A B*
proof −
have *X = Y ∨ B Midpoint X Y*
using *assms(1) assms(2) assms(3) assms(4) assms(5) cong-cop-per2-1* **by** *blast*
then show *?thesis*
using *assms(2) cong-midpoint--image l10-4 per-double-cong* **by** *blast*
qed

lemma *ex-perp-cop*:
assumes *A ≠ B*
shows ∃ *Q. A B Perp Q C ∧ Coplanar A B P Q*
proof −
{
assume *Col A B C ∧ Col A B P*
then have ∃ *Q. A B Perp Q C ∧ Coplanar A B P Q*
using *assms ex-ncol-cop l10-15 ncop--ncols* **by** *blast*
}
then have *T1*: (*Col A B C ∧ Col A B P*) ⟶
(∃ *Q. A B Perp Q C ∧ Coplanar A B P Q*) **by** *blast*
{
assume ¬*Col A B C ∧ Col A B P*
then have ∃ *Q. A B Perp Q C ∧ Coplanar A B P Q*
by (*metis Perp-cases ncop--ncols not-col-distincts perp-exists*)
}
then have *T2*: (¬*Col A B C ∧ Col A B P*) ⟶
(∃ *Q. A B Perp Q C ∧ Coplanar A B P Q*) **by** *blast*

{
assume *Col A B C ∧* ¬*Col A B P*
then have ∃ *Q. A B Perp Q C ∧ Coplanar A B P Q*
using *l10-15 os--coplanar* **by** *blast*
}
then have *T3*: (*Col A B C ∧* ¬*Col A B P*) ⟶
(∃ *Q. A B Perp Q C ∧ Coplanar A B P Q*) **by** *blast*
{
assume ¬*Col A B C ∧* ¬*Col A B P*
then have ∃ *Q. A B Perp Q C ∧ Coplanar A B P Q*
using *l8-18-existence ncop--ncols perp-right-comm* **by** *blast*
}

then have ($\neg$*Col A B C* $\wedge$ $\neg$*Col A B P*) $\longrightarrow$
($\exists$ *Q*. *A B Perp Q C* $\wedge$ *Coplanar A B P Q*) **by** *blast*
then show *?thesis* **using** *T1 T2 T3* **by** *blast*
qed

lemma *hilbert-s-version-of-pasch-aux*:
assumes *Coplanar A B C P* **and**
$\neg$ *Col A I P* **and**
$\neg$ *Col B C P* **and**
Bet B I C **and**
$B \neq I$ **and**
$I \neq C$ **and**
$B \neq C$
shows $\exists$ *X*. *Col I P X* $\wedge$ ((*Bet A X B* $\wedge$ $A \neq X$ $\wedge$ $X \neq B$ $\wedge$ $A \neq B$) $\vee$ (*Bet A X C* $\wedge$ $A \neq X$ $\wedge$ $X \neq C$ $\wedge$ $A \neq C$))
proof −
have *T1*: *I P TS B C*
using *Col-perm assms*(*3*) *assms*(*4*) *assms*(*5*) *assms*(*6*) *bet--ts bet-col col-transitivity-1* **by** *blast*
have *T2*: *Coplanar A P B I*
using *assms*(*1*) *assms*(*4*) *bet-cop--cop coplanar-perm-6 ncoplanar-perm-9* **by** *blast*
have *T3*: *I P TS A B* $\vee$ *I P TS A C*
by (*meson T1 T2 TS-def assms*(*2*) *cop-nos--ts coplanar-perm-21 l9-2 l9-8-2*)
have *T4*: *I P TS A B* $\longrightarrow$
($\exists$ *X*. *Col I P X* $\wedge$
((*Bet A X B* $\wedge$ $A \neq X$ $\wedge$ $X \neq B$ $\wedge$ $A \neq B$) $\vee$
(*Bet A X C* $\wedge$ $A \neq X$ $\wedge$ $X \neq C$ $\wedge$ $A \neq C$)))
by (*metis TS-def not-col-permutation-2 ts-distincts*)
have *I P TS A C* $\longrightarrow$
($\exists$ *X*. *Col I P X* $\wedge$
((*Bet A X B* $\wedge$ $A \neq X$ $\wedge$ $X \neq B$ $\wedge$ $A \neq B$) $\vee$
(*Bet A X C* $\wedge$ $A \neq X$ $\wedge$ $X \neq C$ $\wedge$ $A \neq C$)))
by (*metis TS-def not-col-permutation-2 ts-distincts*)

then show *?thesis* **using** *T3 T4* **by** *blast*
qed

lemma *hilbert-s-version-of-pasch*:
assumes *Coplanar A B C P* **and**
$\neg$ *Col C Q P* **and**
$\neg$ *Col A B P* **and**
BetS A Q B
shows $\exists$ *X*. *Col P Q X* $\wedge$ (*BetS A X C* $\vee$ *BetS B X C*)
proof −
obtain *X* **where** *Col Q P X* $\wedge$
(*Bet C X A* $\wedge$ $C \neq X$ $\wedge$ $X \neq A$ $\wedge$ $C \neq A$ $\vee$
Bet C X B $\wedge$ $C \neq X$ $\wedge$ $X \neq B$ $\wedge$ $C \neq B$)
using *BetSEq assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *coplanar-perm-12 hilbert-s-version-of-pasch-aux* **by** *fastforce*
then show *?thesis*
by (*metis BetS-def Bet-cases Col-perm*)
qed

lemma *two-sides-cases*:
assumes $\neg$ *Col PO A B* **and**
PO P OS A B
shows *PO A TS P B* $\vee$ *PO B TS P A*
by (*meson assms*(*1*) *assms*(*2*) *cop-nts--os l9-31 ncoplanar-perm-3 not-col-permutation-4 one-side-not-col124 one-side-symmetry os--coplanar*)

lemma *not-par-two-sides*:
assumes $C \neq D$ **and**
Col A B I **and**
Col C D I **and**
$\neg$ *Col A B C*
shows $\exists$ *X Y*. *Col C D X* $\wedge$ *Col C D Y* $\wedge$ *A B TS X Y*
proof −
obtain *pp* :: $'p \Rightarrow 'p \Rightarrow 'p$ **where**
f1: $\forall$ *p pa*. *Bet p pa* (*pp p pa*) $\wedge$ *pa* $\neq$ (*pp p pa*)

```
    by (meson point-construction-different)
  then have f2: ∀ p pa pb pc. (Col pc pb p ∨ ¬ Col pc pb (pp p pa)) ∨ ¬ Col pc pb pa
    by (meson Col-def colx)
  have f3: ∀ p pa. Col pa p pa
    by (meson Col-def between-trivial)
  have f4: ∀ p pa. Col pa p p
    by (meson Col-def between-trivial)
  have f5: Col I D C
    by (meson Col-perm assms(3))
  have f6: ∀ p pa. Col (pp pa p) p pa
    using f4 f3 f2 by blast
  then have f7: ∀ p pa. Col pa (pp pa p) p
    by (meson Col-perm)
  then have f8: ∀ p pa pb pc. (pc pb TS p (pp p pa) ∨ Col pc pb p) ∨ ¬ Col pc pb pa
    using f2 f1 by (meson l9-18)
  have I = D ∨ Col D (pp D I) C
    using f7 f5 f3 colx by blast
  then have I = D ∨ Col C D (pp D I)
    using Col-perm by blast
  then show ?thesis
    using f8 f6 f3 by (metis Col-perm assms(2) assms(4))
qed

lemma cop-not-par-other-side:
  assumes C ≠ D and
    Col A B I and
    Col C D I and
    ¬ Col A B C and
    ¬ Col A B P and
    Coplanar A B C P
  shows ∃ Q. Col C D Q ∧ A B TS P Q
proof −
  obtain X Y where P1:Col C D X ∧ Col C D Y ∧ A B TS X Y
    using assms(1) assms(2) assms(3) assms(4) not-par-two-sides by blast
  then have Coplanar C A B X
    using Coplanar-def assms(1) assms(2) assms(3) col-transitivity-1 by blast
  then have Coplanar A B P X
    using assms(4) assms(6) col-permutation-3 coplanar-trans-1 ncoplanar-perm-2 ncoplanar-perm-6 by blast
  then show ?thesis
    by (meson P1 l9-8-2 TS-def assms(5) cop-nts--os not-col-permutation-2 one-side-symmetry)
qed


lemma cop-not-par-same-side:
  assumes C ≠ D and
    Col A B I and
    Col C D I and
    ¬ Col A B C and
    ¬ Col A B P and
    Coplanar A B C P
  shows ∃ Q. Col C D Q ∧ A B OS P Q
proof −
  obtain X Y where P1: Col C D X ∧ Col C D Y ∧ A B TS X Y
    using assms(1) assms(2) assms(3) assms(4) not-par-two-sides by blast
  then have Coplanar C A B X
    using Coplanar-def assms(1) assms(2) assms(3) col-transitivity-1 by blast
  then have Coplanar A B P X
    using assms(4) assms(6) col-permutation-1 coplanar-perm-2 coplanar-trans-1 ncoplanar-perm-14 by blast
  then show ?thesis
    by (meson P1 TS-def assms(5) cop-nts--os l9-2 l9-8-1 not-col-permutation-2)
qed

end
```

3.9.3 Line reflexivity: 2D

context *Tarski-2D*

begin

lemma *all-coplanar*:
Coplanar A B C D
proof −
have ∀ *A B C P Q*. $P \neq Q$ ⟶ *Cong A P A Q* ⟶ *Cong B P B Q*⟶ *Cong C P C Q* ⟶
(*Bet A B C* ∨ *Bet B C A* ∨ *Bet C A B*)
using *upper-dim* **by** *blast*
then show *?thesis* **using** *upper-dim-implies-all-coplanar*
by (*smt Tarski-neutral-dimensionless.not-col-permutation-2 Tarski-neutral-dimensionless-axioms all-coplanar-axiom-def all-coplanar-implies-upper-dim coplanar-perm-9 ncop--ncol os--coplanar ts--coplanar upper-dim-implies-not-one-side-two-sides*)
qed

lemma *per2--col*:
assumes *Per A X C* **and**
$X \neq C$ **and**
Per B X C
shows *Col A B X*
using *all-coplanar-axiom-def all-coplanar-upper-dim assms*(*1*) *assms*(*2*) *assms*(*3*) *upper-dim upper-dim-implies-per2--col*
by *blast*

lemma *perp2--col*:
assumes *X Y Perp A B* **and**
X Z Perp A B
shows *Col X Y Z*
by (*meson Tarski-neutral-dimensionless.cop-perp2--col Tarski-neutral-dimensionless-axioms all-coplanar assms*(*1*) *assms*(*2*))
end

3.10 Angles

3.10.1 Some generalites

context *Tarski-neutral-dimensionless*

begin

lemma *l11-3*:
assumes *A B C CongA D E F*
shows ∃ *A′ C′ D′ F′*. *B Out A′ A* ∧ *B Out C C′* ∧ *E Out D′ D* ∧ *E Out F F′* ∧ *A′ B C′ Cong3 D′ E F′*
proof −
obtain *A′ C′ D′ F′* **where** *P1*: *Bet B A A′* ∧ *Cong A A′ E D* ∧ *Bet B C C′* ∧ *Cong C C′ E F* ∧ *Bet E D D′* ∧ *Cong D D′ B A* ∧ *Bet E F F′* ∧ *Cong F F′ B C* ∧ *Cong A′ C′ D′ F′* **using** *CongA-def*
using *assms* **by** *auto*
then have *A′ B C′ Cong3 D′ E F′*
by (*meson Cong3-def between-symmetry l2-11-b not-cong-1243 not-cong-4312*)
thus *?thesis*
by (*metis CongA-def P1 assms bet-out l6-6*)
qed

lemma *l11-aux*:
assumes *B Out A A′* **and**
E Out D D′ **and**
Cong B A′ E D′ **and**
Bet B A A0 **and**
Bet E D D0 **and**
Cong A A0 E D **and**
Cong D D0 B A
shows *Cong B A0 E D0* ∧ *Cong A′ A0 D′ D0*
proof −
have *P2*: *Cong B A0 E D0*
by (*meson Bet-cases assms*(*4*) *assms*(*5*) *assms*(*6*) *assms*(*7*) *l2-11-b not-cong-1243 not-cong-4312*)
have *P3*: *Bet B A A′* ∨ *Bet B A′ A*
using *Out-def assms*(*1*) **by** *auto*

have *P4*: *Bet E D D′* ∨ *Bet E D′ D*
  using *Out-def assms*(*2*) **by** *auto*
have *P5*: *Bet B A A′* ⟶ *Cong A′ A0 D′ D0*
  by (*smt P2 assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *bet-out l6-6 l6-7 out-cong-cong out-diff1*)
have *P6*: *Bet B A′ A* ⟶ *Cong A′ A0 D′ D0*
proof −
  have *E Out D D0*
    using *assms*(*2*) *assms*(*5*) *bet-out out-diff1* **by** *blast*
  thus *?thesis*
    by (*meson P2 assms*(*2*) *assms*(*3*) *assms*(*4*) *between-exchange4 cong-preserves-bet l4-3-1 l6-6 l6-7*)
qed
have *P7*: *Bet E D D′* ⟶ *Cong A′ A0 D′ D0*
  using *P3 P5 P6* **by** *blast*
have *Bet E D′ D* ⟶ *Cong A′ A0 D′ D0*
  using *P3 P5 P6* **by** *blast*
thus *?thesis*
  using *P2 P3 P4 P5 P6 P7* **by** *blast*
qed

lemma *l11-3-bis*:
  assumes ∃ *A′ C′ D′ F′*. (*B Out A′ A* ∧ *B Out C′ C* ∧ *E Out D′ D* ∧ *E Out F′ F* ∧ *A′ B C′ Cong3 D′ E F′*)
  shows *A B C CongA D E F*
proof −
  obtain *A′ C′ D′ F′* **where** *P1*:
    B Out A′ A ∧ *B Out C′ C* ∧ *E Out D′ D* ∧ *E Out F′ F* ∧ *A′ B C′ Cong3 D′ E F′*
    using *assms* **by** *blast*
  obtain *A0* **where** *P2*: *Bet B A A0* ∧ *Cong A A0 E D*
    using *segment-construction* **by** *presburger*
  obtain *C0* **where** *P3*: *Bet B C C0* ∧ *Cong C C0 E F*
    using *segment-construction* **by** *presburger*
  obtain *D0* **where** *P4*: *Bet E D D0* ∧ *Cong D D0 B A*
    using *segment-construction* **by** *presburger*
  obtain *F0* **where** *P5*: *Bet E F F0* ∧ *Cong F F0 B C*
    using *segment-construction* **by** *presburger*
  have *P6*: *A* ≠ *B* ∧ *C* ≠ *B* ∧ *D* ≠ *E* ∧ *F* ≠ *E*
    using *P1 out-diff2* **by** *blast*
  have *Cong A0 C0 D0 F0*
  proof −
    have *Q1*: *Cong B A0 E D0* ∧ *Cong A′ A0 D′ D0*
    proof −
      have *R1*: *B Out A A′*
        by (*simp add*: *P1 l6-6*)
      have *R2*: *E Out D D′*
        by (*simp add*: *P1 l6-6*)
      have *Cong B A′ E D′*
        using *Cong3-def P1 cong-commutativity* **by** *blast*
      thus *?thesis* **using** *l11-aux*
        using *P2 P4 R1 R2* **by** *blast*
    qed
    have *Q2*: *Cong B C0 E F0* ∧ *Cong C′ C0 F′ F0*
      by (*smt Cong3-def Out-cases P1 P3 P5 Tarski-neutral-dimensionless.l11-aux Tarski-neutral-dimensionless-axioms*)
    have *Q3*: *B A′ A0 Cong3 E D′ D0*
      by (*meson Cong3-def P1 Q1 cong-3-swap*)
    have *Q4*: *B C′ C0 Cong3 E F′ F0*
      using *Cong3-def P1 Q2* **by** *blast*
    have *Cong C0 A′ F0 D′*
    proof −
      have *R1*: *B C′ C0 A′ FSC E F′ F0 D′*
      proof −
        have *S1*: *Col B C′ C0*
          by (*metis* (*no-types*) *Col-perm P1 P3 P6 bet-col col-transitivity-1 out-col*)
        have *S3*: *Cong B A′ E D′*
          using *Cong3-def Q3* **by** *blast*
        have *Cong C′ A′ F′ D′*
          using *Cong3-def P1 cong-commutativity* **by** *blast*
        thus *?thesis*

by (*simp add*: *FSC-def S1 Q4 S3*)
qed
have $B \neq C'$
using *P1 out-distinct* **by** *blast*
thus *?thesis*
using *R1 l4-16* **by** *blast*
qed
then have *Q6*: $B\ A'\ A0\ C0$ *FSC* $E\ D'\ D0\ F0$
by (*meson FSC-def P1 P2 P6 Q2 Q3 bet-out l6-7 not-cong-2143 out-col*)
have $B \neq A'$
using *Out-def P1* **by** *blast*
thus *?thesis*
using *Q6 l4-16* **by** *blast*
qed
thus *?thesis* **using** *P6 P2 P3 P4 P5 CongA-def* **by** *auto*
qed

lemma *l11-4-1*:
assumes *A B C CongA D E F* **and**

B Out A' *A* **and**
B Out C' *C* **and**
E Out D' *D* **and**
E Out F' *F* **and**
Cong B A' *E* D' **and** *Cong B* C' *E* F'
shows *Cong* $A'\ C'\ D'\ F'$
proof −
obtain *A0 C0 D0 F0* **where** *P1*: *B Out A0 A* ∧ *B Out C C0* ∧ *E Out D0 D* ∧ *E Out F F0* ∧ *A0 B C0 Cong3 D0 E F0*
using *assms*(*1*) *l11-3* **by** *blast*
have *P2*: *B Out* A' *A0*
using *P1 assms*(*2*) *l6-6 l6-7* **by** *blast*
have *P3*: *E Out* D' *D0*
by (*meson P1 assms*(*4*) *l6-6 l6-7*)
have *P4*: *Cong* A' *A0* D' *D0*
proof −
have *Cong B A0 E D0*
using *Cong3-def P1 cong-3-swap* **by** *blast*
thus *?thesis* **using** *P2 P3*
using *assms*(*6*) *out-cong-cong* **by** *blast*
qed
have *P5*: *Cong* A' *C0* D' *F0*
proof −
have *P6*: *B A0* A' *C0 FSC E D0* D' *F0*
by (*meson Cong3-def Cong-perm FSC-def P1 P2 P4 assms*(*6*) *not-col-permutation-5 out-col*)
thus *?thesis*
using *P2 Tarski-neutral-dimensionless.l4-16 Tarski-neutral-dimensionless-axioms out-diff2* **by** *fastforce*
qed
have *P6*: *B Out* C' *C0*
using *P1 assms*(*3*) *l6-7* **by** *blast*
have *E Out* F' *F0*
using *P1 assms*(*5*) *l6-7* **by** *blast*
then have *Cong* C' *C0* F' *F0*
using *Cong3-def P1 P6 assms*(*7*) *out-cong-cong* **by** *auto*
then have *P9*: *B C0* $C'\ A'$ *FSC E F0* $F'\ D'$
by (*smt Cong3-def Cong-perm FSC-def P1 P5 P6 assms*(*6*) *assms*(*7*) *not-col-permutation-5 out-col*)
then have *Cong* $C'\ A'\ F'\ D'$
using *P6 Tarski-neutral-dimensionless.l4-16 Tarski-neutral-dimensionless-axioms out-diff2* **by** *fastforce*
thus *?thesis*
using *Tarski-neutral-dimensionless.not-cong-2143 Tarski-neutral-dimensionless-axioms* **by** *fastforce*
qed

lemma *l11-4-2*:
assumes $A \neq B$ **and**
$C \neq B$ **and**
$D \neq E$ **and**

$F \neq E$ **and**
$\forall$ A′ C′ D′ F′. (B Out A′ A ∧ B Out C′ C ∧ E Out D′ D ∧ E Out F′ F ∧ Cong B A′ E D′ ∧ Cong B C′ E F′ ⟶ Cong A′ C′ D′ F′)
shows *A B C CongA D E F*
proof −
obtain *A′* **where** *P1*: *Bet B A A′ ∧ Cong A A′ E D*
using *segment-construction* **by** *fastforce*
obtain *C′* **where** *P2*: *Bet B C C′ ∧ Cong C C′ E F*
using *segment-construction* **by** *fastforce*
obtain *D′* **where** *P3*: *Bet E D D′ ∧ Cong D D′ B A*
using *segment-construction* **by** *fastforce*
obtain *F′* **where** *P4*: *Bet E F F′ ∧ Cong F F′ B C*
using *segment-construction* **by** *fastforce*
have *P5*: *Cong A′ B D′ E*
by (*meson Bet-cases P1 P3 l2-11-b not-cong-1243 not-cong-4312*)
have *P6*: *Cong B C′ E F′*
by (*meson P2 P4 between-symmetry cong-3421 cong-right-commutativity l2-11-b*)
have *B Out A′ A ∧ B Out C′ C ∧ E Out D′ D ∧ E Out F′ F ∧ A′ B C′ Cong3 D′ E F′*
by (*metis* (*no-types, lifting*) *Cong3-def P1 P2 P3 P4 P5 P6 Tarski-neutral-dimensionless.Out-def Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *bet-neq12--neq cong-commutativity*)
thus *?thesis*
using *l11-3-bis* **by** *blast*
qed

lemma *conga-refl*:
assumes $A \neq B$ **and**
$C \neq B$
shows *A B C CongA A B C*
by (*meson CongA-def assms*(*1*) *assms*(*2*) *cong-reflexivity segment-construction*)

lemma *conga-sym*:
assumes *A B C CongA A′ B′ C′*
shows *A′ B′ C′ CongA A B C*
proof −
obtain *A0 C0 D0 F0* **where**
P1: *Bet B A A0 ∧ Cong A A0 B′ A′ ∧ Bet B C C0 ∧ Cong C C0 B′ C′ ∧ Bet B′ A′ D0 ∧ Cong A′ D0 B A ∧ Bet B′ C′ F0 ∧ Cong C′ F0 B C ∧ Cong A0 C0 D0 F0*
using *CongA-def assms* **by** *auto*
thus *?thesis*
proof −
have *∃ p pa pb pc. Bet B′ A′ p ∧ Cong A′ p B A ∧ Bet B′ C′ pa ∧ Cong C′ pa B C ∧Bet B A pb ∧ Cong A pb B′ A′ ∧Bet B C pc ∧ Cong C pc B′ C′ ∧ Cong p pa pb pc*
by (*metis* (*no-types*) *Tarski-neutral-dimensionless.cong-symmetry Tarski-neutral-dimensionless-axioms P1*)
thus *?thesis*
using *CongA-def assms* **by** *auto*
qed
qed

lemma *l11-10*:
assumes *A B C CongA D E F* **and**
B Out A′ A **and**
B Out C′ C **and**
E Out D′ D **and**
E Out F′ F
shows *A′ B C′ CongA D′ E F′*
proof −
have *P1*: $A' \neq B$
using *assms*(*2*) *out-distinct* **by** *auto*
have *P2*: $C' \neq B$
using *Out-def assms*(*3*) **by** *force*
have *P3*: $D' \neq E$
using *Out-def assms*(*4*) **by** *blast*
have *P4*: $F' \neq E$
using *assms*(*5*) *out-diff1* **by** *auto*
have *P5*: *∀ A′0 C′0 D′0 F′0. (B Out A′0 A′ ∧ B Out C′0 C′ ∧ E Out D′0 D′ ∧ E Out F′0 F′ ∧ Cong B A′0 E D′0 ∧ Cong B C′0 E F′0) ⟶ Cong A′0 C′0 D′0 F′0*

by (*meson assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *l11-4-1 l6-7*)
thus *?thesis* **using** *P1 P2 P3 P4 P5 l11-4-2* **by** *blast*
qed

lemma *out2--conga*:
assumes B *Out* A' A **and**
B *Out* C' C
shows A B C *CongA* A' B C'
by (*smt assms*(*1*) *assms*(*2*) *between-trivial2 conga-refl l11-10 out2-bet-out out-distinct*)

lemma *cong3-diff*:
assumes $A \neq B$ **and**
A B C *Cong3* A' B' C'
shows $A' \neq B'$
using *Cong3-def assms*(*1*) *assms*(*2*) *cong-diff* **by** *blast*

lemma *cong3-diff2*:
assumes $B \neq C$ **and**
A B C *Cong3* A' B' C'
shows $B' \neq C'$
using *Cong3-def assms*(*1*) *assms*(*2*) *cong-diff* **by** *blast*

lemma *cong3-conga*:
assumes $A \neq B$ **and**
$C \neq B$ **and**
A B C *Cong3* A' B' C'
shows A B C *CongA* A' B' C'
by (*metis assms*(*1*) *assms*(*2*) *assms*(*3*) *cong3-diff cong3-diff2 l11-3-bis out-trivial*)

lemma *cong3-conga2*:
assumes A B C *Cong3* A' B' C' **and**
A B C *CongA* A'' B'' C''
shows A' B' C' *CongA* A'' B'' C''
proof −
obtain *A0 C0 A2 C2* **where** *P1*: *Bet B A A0* $\wedge$ *Cong A A0* B'' A'' $\wedge$ *Bet B C C0* $\wedge$ *Cong C C0* B'' $C''\wedge$ *Bet* B'' A'' *A2* $\wedge$ *Cong* A'' *A2 B A* $\wedge$ *Bet* B'' C'' *C2* $\wedge$ *Cong* C'' *C2 B C* $\wedge$ *Cong A0 C0 A2 C2*
using *CongA-def assms*(*2*) **by** *auto*
obtain *A1* **where** *P5*: *Bet* B' A' *A1* $\wedge$ *Cong* A' *A1* B'' A''
using *segment-construction* **by** *blast*
obtain *C1* **where** *P6*: *Bet* B' C' *C1* $\wedge$ *Cong* C' *C1* B'' C''
using *segment-construction* **by** *blast*
have *P7*: *Cong A A0* A' *A1*
proof −
have *Cong* B'' A'' A' *A1* **using** *P5*
using *Cong-perm* **by** *blast*
thus *?thesis*
using *Cong-perm P1 cong-inner-transitivity* **by** *blast*
qed
have *P8*: *Cong B A0* B' *A1*
using *Cong3-def P1 P5 P7 assms*(*1*) *cong-commutativity l2-11-b* **by** *blast*
have *P9*: *Cong C C0* C' *C1*
using *P1 P6 cong-inner-transitivity cong-symmetry* **by** *blast*
have *P10*: *Cong B C0* B' *C1*
using *Cong3-def P1 P6 P9 assms*(*1*) *l2-11-b* **by** *blast*
have *B A A0 C FSC* B' A' *A1* C'
using *FSC-def P1 P5 P7 P8 Tarski-neutral-dimensionless.Cong3-def Tarski-neutral-dimensionless-axioms assms*(*1*) *bet-col l4-3* **by** *fastforce*
then have *P12*: *Cong A0 C A1* C'
using *CongA-def assms*(*2*) *l4-16* **by** *auto*
then have *B C C0 A0 FSC* B' C' *C1 A1*
using *Cong3-def FSC-def P1 P10 P8 P9 assms*(*1*) *bet-col cong-commutativity* **by** *auto*
then have *P13*: *Cong C0 A0 C1 A1*
using *l4-16 CongA-def assms*(*2*) **by** *blast*
have *Q2*: *Cong* A' *A1* B'' A''
using *P1 P7 cong-inner-transitivity* **by** *blast*
have *Q5*: *Bet* B'' A'' *A2* **using** *P1* **by** *blast*

```
  have Q6: Cong A'' A2 B' A'
  proof -
    have Cong B A B' A'
      using P1 P7 P8 P5 l4-3 by blast
    thus ?thesis
      using P1 cong-transitivity by blast
  qed
  have Q7: Bet B'' C'' C2
    using P1 by blast
  have Q8: Cong C'' C2 B' C'
  proof -
    have Cong B C B' C'
      using Cong3-def assms(1) by blast
    thus ?thesis
      using P1 cong-transitivity by blast
  qed
  have R2: Cong C0 A0 C2 A2
    using Cong-cases P1 by blast
  have Cong C1 A1 A0 C0
    using Cong-cases P13 by blast
  then have Q9: Cong A1 C1 A2 C2
    using R2 P13 cong-inner-transitivity not-cong-4321 by blast
  thus ?thesis
    using CongA-def P5 Q2 P6 Q5 Q6 Q7 Q8
    by (metis assms(1) assms(2) cong3-diff cong3-diff2)
qed

lemma conga-diff1:
  assumes A B C CongA A' B' C'
  shows A ≠ B
  using CongA-def assms by blast

lemma conga-diff2:
  assumes A B C CongA A' B' C'
  shows C ≠ B
  using CongA-def assms by blast

lemma conga-diff45:
  assumes A B C CongA A' B' C'
  shows A' ≠ B'
  using CongA-def assms by blast

lemma conga-diff56:
  assumes A B C CongA A' B' C'
  shows C' ≠ B'
  using CongA-def assms by blast

lemma conga-trans:
  assumes A B C CongA A' B' C' and
    A' B' C' CongA A'' B'' C''
  shows A B C CongA A'' B'' C''
proof -
  obtain A0 C0 A1 C1 where P1: Bet B A A0 ∧ Cong A A0 B' A' ∧
Bet B C C0 ∧ Cong C C0 B' C'∧ Bet B' A' A1 ∧ Cong A' A1 B A ∧
Bet B' C' C1 ∧ Cong C' C1 B C ∧ Cong A0 C0 A1 C1
    using CongA-def assms(1) by auto
  have P2: A''≠ B'' ∧ C'' ≠ B''
    using CongA-def assms(2) by auto
  have P3: A1 B' C1 CongA A'' B'' C''
  proof -
    have L2: B' Out A1 A' using P1
      by (metis Out-def assms(2) bet-neq12--neq conga-diff1)
    have L3: B' Out C1 C' using P1
      by (metis Out-def assms(1) bet-neq12--neq conga-diff56)
    have L4: B'' Out A'' A''
      using P2 out-trivial by auto
```

```
    have B'' Out C'' C''
      by (simp add: P2 out-trivial)
    thus ?thesis
      using assms(2) L2 L3 L4 l11-10 by blast
  qed
  have L6: A0 B C0 CongA A' B' C'
   by (smt Out-cases P1 Tarski-neutral-dimensionless.conga-diff1 Tarski-neutral-dimensionless.conga-diff2 Tarski-neutral-dimensionless
Tarski-neutral-dimensionless-axioms assms(1) bet-out conga-diff56 l11-10 l5-3)
  have L7: Cong B A0 B' A1
    by (meson P1 between-symmetry cong-3421 l2-11-b not-cong-1243)
  have L8: Cong B C0 B' C1
    using P1 between-symmetry cong-3421 l2-11-b not-cong-1243 by blast
  have L10: A0 B C0 Cong3 A1 B' C1
    by (simp add: Cong3-def L7 L8 P1 cong-commutativity)
  then have L11: A0 B C0 CongA A'' B'' C''
    by (meson Tarski-neutral-dimensionless.cong3-conga2 Tarski-neutral-dimensionless-axioms P3 cong-3-sym)
  thus ?thesis using l11-10
  proof -
    have D2: B Out A A0 using P1
      using CongA-def assms(1) bet-out by auto
    have D3: B Out C C0 using P1
      using CongA-def assms(1) bet-out by auto
    have D4: B'' Out A'' A''
      using P2 out-trivial by blast
    have B'' Out C'' C''
      using P2 out-trivial by auto
    thus ?thesis using l11-10 L11 D2 D3 D4
      by blast
  qed
qed

lemma conga-pseudo-refl:
  assumes A ≠ B and
    C ≠ B
  shows A B C CongA C B A
  by (meson CongA-def assms(1) assms(2) between-trivial cong-pseudo-reflexivity segment-construction)

lemma conga-trivial-1:
  assumes A ≠ B and
    C ≠ D
  shows A B A CongA C D C
  by (meson CongA-def assms(1) assms(2) cong-trivial-identity segment-construction)

lemma l11-13:
  assumes A B C CongA D E F and
    Bet A B A' and
    A'≠ B and
    Bet D E D' and
    D'≠ E
  shows A' B C CongA D' E F
proof -
  obtain A'' C'' D'' F'' where P1:
    Bet B A A'' ∧ Cong A A'' E D ∧
Bet B C C'' ∧ Cong C C'' E F ∧ Bet E D D'' ∧
Cong D D'' B A ∧
Bet E F F'' ∧ Cong F F'' B C ∧ Cong A'' C'' D'' F''
    using CongA-def assms(1) by auto
  obtain A0 where P2:Bet B A' A0 ∧ Cong A' A0 E D'
    using segment-construction by blast
  obtain D0 where P3: Bet E D' D0 ∧ Cong D' D0 B A'
    using segment-construction by blast
  have Cong A0 C'' D0 F''
  proof -
    have L1: A'' B A0 C'' OFSC D'' E D0 F''
    proof -
      have L2: Bet A'' B A0
```

```
      proof -
        have M1: Bet A'' A B
          using Bet-perm P1 by blast
        have M2: Bet A B A0
          using P2 assms(2) assms(3) outer-transitivity-between by blast
        have A ≠ B
          using CongA-def assms(1) by blast
        thus ?thesis
          using M1 M2 outer-transitivity-between2 by blast
      qed
      have L3: Bet D'' E D0 using Bet-perm P1 P2 outer-transitivity-between CongA-def
        by (metis P3 assms(1) assms(4) assms(5))
      have L4: Cong A'' B D'' E
        by (meson P1 between-symmetry cong-3421 cong-left-commutativity l2-11-b)
      have L5: Cong B A0 E D0
        by (meson P2 P3 between-symmetry cong-3421 cong-right-commutativity l2-11-b)
      have Cong B C'' E F''
        by (meson P1 between-symmetry cong-3421 cong-right-commutativity l2-11-b)
      thus ?thesis using P1 L2 L3 L4 L5
        by (simp add: OFSC-def)
    qed
    have A'' ≠ B
      using CongA-def P1 assms(1) bet-neq12--neq by fastforce
    thus ?thesis
      using L1 five-segment-with-def by auto
  qed
  thus ?thesis
    using CongA-def P1 P2 P3 assms(1) assms(3) assms(5) by auto
qed

lemma conga-right-comm:
  assumes A B C CongA D E F
  shows A B C CongA F E D
 by (metis Tarski-neutral-dimensionless.conga-diff45 Tarski-neutral-dimensionless.conga-sym Tarski-neutral-dimensionless.conga-trans
Tarski-neutral-dimensionless-axioms assms conga-diff56 conga-pseudo-refl)

lemma conga-left-comm:
  assumes A B C CongA D E F
  shows C B A CongA D E F
  by (meson assms conga-right-comm conga-sym)

lemma conga-comm:
  assumes A B C CongA D E F
  shows C B A CongA F E D
 by (meson Tarski-neutral-dimensionless.conga-left-comm Tarski-neutral-dimensionless.conga-right-comm Tarski-neutral-dimensionless
assms)

lemma conga-line:
  assumes A ≠ B and
    B ≠ C and
    A' ≠ B' and
    B' ≠ C'
    and Bet A B C and
    Bet A' B' C'
  shows A B C CongA A' B' C'
  by (metis Bet-cases assms(1) assms(2) assms(3) assms(4) assms(5) assms(6) conga-trivial-1 l11-13)

lemma l11-14:
  assumes Bet A B A' and
    A ≠ B and
    A' ≠ B and
    Bet C B C' and
    B ≠ C and
    B ≠ C'
  shows A B C CongA A' B C'
   by (metis Bet-perm assms(1) assms(2) assms(3) assms(4) assms(5) assms(6) conga-pseudo-refl conga-right-comm
```

l11-13)

lemma *l11-16*:
assumes *Per A B C* **and**
$A \neq B$ **and**
$C \neq B$ **and**
Per A′ B′ C′ **and**
$A' \neq B'$ **and**
$C' \neq B'$
shows *A B C CongA A′ B′ C′*
proof −
obtain *C0* **where** *P1*: *Bet B C C0* ∧ *Cong C C0 B′ C′*
using *segment-construction* **by** *blast*
obtain *C1* **where** *P2*: *Bet B′ C′ C1* ∧ *Cong C′ C1 B C*
using *segment-construction* **by** *blast*
obtain *A0* **where** *P3*: *Bet B A A0* ∧ *Cong A A0 B′ A′*
using *segment-construction* **by** *blast*
obtain *A1* **where** *P4*: *Bet B′ A′ A1* ∧ *Cong A′ A1 B A*
using *segment-construction* **by** *blast*
have *Cong A0 C0 A1 C1*
proof −
have *Q1*: *Per A0 B C0*
by (*metis P1 P3 Tarski-neutral-dimensionless.l8-3 Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*2*) *assms*(*3*) *bet-col per-col*)
have *Q2*: *Per A1 B′ C1*
by (*metis P2 P4 Tarski-neutral-dimensionless.l8-3 Tarski-neutral-dimensionless-axioms assms*(*4*) *assms*(*5*) *assms*(*6*) *bet-col per-col*)
have *Q3*: *Cong A0 B A1 B′*
by (*meson P3 P4 between-symmetry cong-3421 cong-left-commutativity l2-11-b*)
have *Cong B C0 B′ C1*
using *P1 P2 between-symmetry cong-3421 l2-11-b not-cong-1243* **by** *blast*
thus *?thesis*
using *Q1 Q2 Q3 l10-12* **by** *blast*
qed
thus *?thesis*
using *CongA-def P1 P2 P3 P4 assms*(*2*) *assms*(*3*) *assms*(*5*) *assms*(*6*) **by** *auto*
qed

lemma *l11-17*:
assumes *Per A B C* **and**
A B C CongA A′ B′ C′
shows *Per A′ B′ C′*
proof −
obtain *A0 C0 A1 C1* **where** *P1*: *Bet B A A0* ∧ *Cong A A0 B′ A′* ∧ *Bet B C C0* ∧ *Cong C C0 B′ C′* ∧ *Bet B′ A′ A1* ∧ *Cong A′ A1 B A* ∧ *Bet B′ C′ C1* ∧ *Cong C′ C1 B C* ∧ *Cong A0 C0 A1 C1*
using *CongA-def assms*(*2*) **by** *auto*
have *P2*: *Per A0 B C0*
proof −
have *Q1*: $B \neq C$
using *assms*(*2*) *conga-diff2* **by** *blast*
have *Q2*: *Per A0 B C*
by (*metis P1 Tarski-neutral-dimensionless.l8-2 Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*2*) *bet-col conga-diff1 per-col*)
have *Col B C C0*
using *P1 bet-col* **by** *blast*
thus *?thesis*
using *Q1 Q2 per-col* **by** *blast*
qed
have *P3*: *Per A1 B′ C1*
proof −
have *A0 B C0 Cong3 A1 B′ C1*
by (*meson Bet-cases Cong3-def P1 l2-11-b not-cong-2134 not-cong-3421*)
thus *?thesis*
using *P2 l8-10* **by** *blast*
qed
have *P4*: $B' \neq C1$

```
    using P1 assms(2) between-identity conga-diff56 by blast
  have P5: Per A' B' C1
  proof -
    have P6: B' ≠ A1
      using P1 assms(2) between-identity conga-diff45 by blast
    have P7: Per C1 B' A1
      by (simp add: P3 l8-2)
    have Col B' A1 A'
      using P1 NCol-cases bet-col by blast
    thus ?thesis
      using P3 P6 Tarski-neutral-dimensionless.l8-3 Tarski-neutral-dimensionless-axioms by fastforce
  qed
  have Col B' C1 C'
    using P1 bet-col col-permutation-5 by blast
  thus ?thesis
    using P4 P5 per-col by blast
qed

lemma l11-18-1:
  assumes Bet C B D and
    B ≠ C and
    B ≠ D and
    A ≠ B and
    Per A B C
  shows A B C CongA A B D
  by (smt Tarski-neutral-dimensionless.l8-2 Tarski-neutral-dimensionless.l8-5 Tarski-neutral-dimensionless-axioms assms(1)
assms(2) assms(3) assms(4) assms(5) bet-col col-per2--per l11-16)

lemma l11-18-2:
  assumes Bet C B D and
    A B C CongA A B D
  shows Per A B C
proof -
  obtain A0 C0 A1 D0 where P1: Bet B A A0 ∧ Cong A A0 B A ∧ Bet B C C0 ∧
Cong C C0 B D ∧ Bet B A A1 ∧ Cong A A1 B A ∧
Bet B D D0 ∧ Cong D D0 B C ∧ Cong A0 C0 A1 D0
    using CongA-def assms(2) by auto
  have P2: A0 = A1
    by (metis P1 assms(2) conga-diff45 construction-uniqueness)
  have P3: Per A0 B C0
  proof -
    have Q1: Bet C0 B D0
    proof -
      have R1: Bet C0 C B
        using P1 between-symmetry by blast
      have R2: Bet C B D0
      proof -
        have S1: Bet C B D
          by (simp add: assms(1))
        have S2: Bet B D D0
          using P1 by blast
        have B ≠ D
          using assms(2) conga-diff56 by blast
        thus ?thesis
          using S1 S2 outer-transitivity-between by blast
      qed
      have C ≠ B
        using assms(2) conga-diff2 by auto
      thus ?thesis
        using R1 R2 outer-transitivity-between2 by blast
    qed
    have Q2: Cong C0 B B D0
      by (meson P1 between-symmetry cong-3421 l2-11-b not-cong-1243)
    have Cong A0 C0 A0 D0
      using P1 P2 by blast
    thus ?thesis
```

using *Per-def Q1 Q2 midpoint-def* **by** *blast*
qed
have *P4*: $B \neq C0$
using *P1 assms*(*2*) *bet-neq12--neq conga-diff2* **by** *blast*
have *P5*: *Per A B C0*
by (*metis P1 P3 Tarski-neutral-dimensionless.l8-3 Tarski-neutral-dimensionless-axioms assms*(*2*) *bet-col bet-col1 bet-neq21--neq col-transitivity-1 conga-diff45*)
have *Col B C0 C* **using** *P1*
using *NCol-cases bet-col* **by** *blast*
thus *?thesis*
using *P4 P5 per-col* **by** *blast*
qed

lemma *cong3-preserves-out*:
assumes *A Out B C* **and**
A B C Cong3 A′ B′ C′
shows *A′ Out B′ C′*
by (*meson assms*(*1*) *assms*(*2*) *col-permutation-4 cong3-symmetry cong-3-swap l4-13 l4-6 not-bet-and-out or-bet-out out-col*)

lemma *l11-21-a*:
assumes *B Out A C* **and**
A B C CongA A′ B′ C′
shows *B′ Out A′ C′*
proof −
obtain *A0 C0 A1 C1* **where** *P1*: *Bet B A A0* ∧
Cong A A0 B′ A′ ∧ *Bet B C C0* ∧
Cong C C0 B′ C′ ∧ *Bet B′ A′ A1* ∧
Cong A′ A1 B A ∧ *Bet B′ C′ C1* ∧
Cong C′ C1 B C ∧ *Cong A0 C0 A1 C1*
using *CongA-def assms*(*2*) **by** *auto*
have *P2*: *B Out A0 C0*
by (*metis P1 assms*(*1*) *bet-out l6-6 l6-7 out-diff1*)
have *P3*: *B′ Out A1 C1*
proof −
have *B A0 C0 Cong3 B′ A1 C1*
by (*meson Cong3-def P1 between-symmetry cong-right-commutativity l2-11-b not-cong-4312*)
thus *?thesis*
using *P2 cong3-preserves-out* **by** *blast*
qed
thus *?thesis*
by (*metis P1 assms*(*2*) *bet-out conga-diff45 conga-diff56 l6-6 l6-7*)
qed

lemma *l11-21-b*:
assumes *B Out A C* **and**
B′ Out A′ C′
shows *A B C CongA A′ B′ C′*
by (*smt assms*(*1*) *assms*(*2*) *between-trivial2 conga-trivial-1 l11-10 out2-bet-out out-distinct*)

lemma *conga-cop--or-out-ts*:
assumes *Coplanar A B C C′* **and**
A B C CongA A B C′
shows *B Out C C′* ∨ *A B TS C C′*
proof −
obtain *A0 C0 A1 C1* **where** *P1*: *Bet B A A0* ∧
Cong A A0 B A ∧*Bet B C C0* ∧
Cong C C0 B C′ ∧*Bet B A A1* ∧
Cong A A1 B A ∧*Bet B C′ C1* ∧
Cong C′ C1 B C ∧ *Cong A0 C0 A1 C1*
using *CongA-def assms*(*2*) **by** *auto*
have *P2*: *A0 = A1* **using** *P1*
by (*metis assms*(*2*) *conga-diff1 construction-uniqueness*)
have *B Out C C′* ∨ *A B TS C C′*
proof *cases*
assume *C0 = C1*

```
      thus ?thesis
        by (metis P1 assms(2) bet2--out conga-diff2 conga-diff56)
    next
      assume R1: C0 ≠ C1
      obtain M where R2: M Midpoint C0 C1
        using midpoint-existence by blast
      have R3: Cong B C0 B C1
        by (meson Bet-cases P1 l2-11-b not-cong-2134 not-cong-3421)
      have R3A: Cong A0 C0 A0 C1
        using P1 P2 by blast
      then have R4: Per A0 M C0 using R2
        using Per-def by blast
      have R5: Per B M C0
        using Per-def R2 R3 by auto
      then have R6: Per B M C1
        using R2 l8-4 by blast
      have R7: B ≠ A0
        using P1 assms(2) bet-neq12--neq conga-diff45 by blast
      then have Cong A C0 A C1
        by (meson Col-perm P1 R3 R3A bet-col l4-17)
      then have R9: Per A M C0
        using Per-def R2 by blast
      then have R10: Per A M C1
        by (meson R2 Tarski-neutral-dimensionless.l8-4 Tarski-neutral-dimensionless-axioms)
      have R11: Col B A M
      proof –
        have S1: Coplanar C0 B A M
        proof –
          have Coplanar B A C0 M
          proof –
            have T1: Coplanar B A C0 C1
            proof –
              have Coplanar A C0 B C'
              proof –
                have Coplanar A C' B C0
                proof –
                  have U1: Coplanar A C' B C
                    by (simp add: assms(1) coplanar-perm-4)
                  have U2: B ≠ C
                    using assms(2) conga-diff2 by blast
                  have Col B C C0
                    by (simp add: P1 bet-col)
                  thus ?thesis
                    by (meson Tarski-neutral-dimensionless.col-cop--cop Tarski-neutral-dimensionless-axioms U1 U2)
                qed
                thus ?thesis
                  using ncoplanar-perm-5 by blast
              qed

              thus ?thesis
               by (metis P1 Tarski-neutral-dimensionless.col-cop--cop Tarski-neutral-dimensionless-axioms assms(2) bet-col
conga-diff56 coplanar-perm-12)
            qed
            have Col C0 C1 M
              using Col-perm R2 midpoint-col by blast
            thus ?thesis
              using T1 R1 col-cop--cop by blast
          qed
          thus ?thesis
            using ncoplanar-perm-8 by blast
        qed
        have C0 ≠ M
          using R1 R2 midpoint-distinct-1 by blast
        thus ?thesis
          using R5 R9 S1 cop-per2--col by blast
      qed
```

have *B Out C C′ ∨ A B TS C C′*
proof *cases*
  assume *Q1*: *B = M*
  have *Q2*: *¬ Col A B C*
    by (*metis Col-def P1 Q1 R9 assms*(*2*) *conga-diff1 conga-diff2 l6-16-1 l8-9 not-bet-and-out out-trivial*)
  have *Q3*: *¬ Col A B C′*
    by (*metis Col-def P1 Q1 R10 assms*(*2*) *conga-diff1 conga-diff56 l6-16-1 l8-9 not-bet-and-out out-trivial*)
  have *Q4*: *Col B A B*
    by (*simp add*: *col-trivial-3*)
  have *Bet C B C′*
  proof −
    have *S1*: *Bet C1 C′ B*
      using *Bet-cases P1* **by** *blast*
    have *Bet C1 B C*
    proof −
      have *T1*: *Bet C0 C B*
        using *Bet-cases P1* **by** *blast*
      have *Bet C0 B C1*
        by (*simp add*: *Q1 R2 midpoint-bet*)
      thus *?thesis*
        using *T1 between-exchange3 between-symmetry* **by** *blast*
    qed
    thus *?thesis*
      using *S1 between-exchange3 between-symmetry* **by** *blast*
  qed
  thus *?thesis*
    by (*metis Q2 Q3 Q4 bet--ts col-permutation-4 invert-two-sides*)
next
  assume *L1*: *B ≠ M*
  have *L2*: *B M TS C0 C1*
  proof −
    have *M1*: *¬ Col C0 B M*
      by (*metis* (*no-types*) *Col-perm L1 R1 R2 R5 is-midpoint-id l8-9*)
    have *M2*: *¬ Col C1 B M*
      using *Col-perm L1 R1 R2 R6 l8-9 midpoint-not-midpoint* **by** *blast*
    have *M3*: *Col M B M*
      using *col-trivial-3* **by** *auto*
    have *Bet C0 M C1*
      by (*simp add*: *R2 midpoint-bet*)
    thus *?thesis*
      using *M1 M2 M3 TS-def* **by** *blast*
  qed
  have *A B TS C C′*
  proof −
    have *W2*: *A B TS C C1*
    proof −
      have *V1*: *A B TS C0 C1*
        using *L2 P1 R11 R7 col-two-sides cong-diff invert-two-sides not-col-permutation-5* **by** *blast*
      have *B Out C0 C*
        using *L2 Out-def P1 TS-def assms*(*2*) *col-trivial-1 conga-diff2* **by** *auto*
      thus *?thesis*
        using *V1 col-trivial-3 l9-5* **by** *blast*
    qed
    then have *W1*: *A B TS C′ C*
    proof −
      have *Z1*: *A B TS C1 C*
        by (*simp add*: *W2 l9-2*)
      have *Z2*: *Col B A B*
        using *not-col-distincts* **by** *blast*
      have *B Out C1 C′*
        using *L2 Out-def P1 TS-def assms*(*2*) *col-trivial-1 conga-diff56* **by** *auto*
      thus *?thesis*
        using *Z1 Z2 l9-5* **by** *blast*
    qed
    thus *?thesis*
      by (*simp add*: *l9-2*)

qed
thus ?thesis by blast
qed
thus ?thesis by blast
qed
thus ?thesis by blast
qed

lemma *conga-os--out*:
assumes *A B C CongA A B C′* **and**
A B OS C C′
shows *B Out C C′*
using *assms(1) assms(2) conga-cop--or-out-ts l9-9 os--coplanar* **by** *blast*

lemma *cong2-conga-cong*:
assumes *A B C CongA A′ B′ C′* **and**
Cong A B A′ B′ **and**
Cong B C B′ C′
shows *Cong A C A′ C′*
by (*smt assms(1) assms(2) assms(3) cong-4321 l11-3 l11-4-1 not-cong-3412 out-distinct out-trivial*)

lemma *angle-construction-1*:
assumes ¬ *Col A B C* **and**
¬ *Col A′ B′ P*
shows ∃ *C′*. (*A B C CongA A′ B′ C′* ∧ *A′ B′ OS C′ P*)
proof −
obtain *C0* **where** *P1*: *Col B A C0* ∧ *B A Perp C C0*
using *assms(1) col-permutation-4 l8-18-existence* **by** *blast*
have ∃ *C′*. (*A B C CongA A′ B′ C′* ∧ *A′ B′ OS C′ P*)
proof *cases*
assume *P1A*: *B* = *C0*
obtain *C′* **where** *P2*: *Per C′ B′ A′* ∧ *Cong C′ B′ C B* ∧ *A′ B′ OS C′ P*
by (*metis assms(1) assms(2) col-trivial-1 col-trivial-2 ex-per-cong*)
have *P3*: *A B C CongA A′ B′ C′*
by (*metis P1 P2 Tarski-neutral-dimensionless.l8-2 Tarski-neutral-dimensionless.os-distincts Tarski-neutral-dimensionless-axioms P1A assms(1) l11-16 not-col-distincts perp-per-1*)
thus *?thesis* **using** *P2* **by** *blast*
next
assume *P4*: *B* ≠ *C0*
have ∃ *C′*. (*A B C CongA A′ B′ C′* ∧ *A′ B′ OS C′ P*)
proof *cases*
assume *R1*: *B Out A C0*
obtain *C0′* **where** *R2*: *B′ Out A′ C0′* ∧ *Cong B′ C0′ B C0*
by (*metis P4 assms(2) col-trivial-1 segment-construction-3*)
have ∃ *C′*. *Per C′ C0′ B′* ∧ *Cong C′ C0′ C C0* ∧ *B′ C0′ OS C′ P*
proof −
have *R4*: *B′* ≠ *C0′*
using *Out-def R2* **by** *auto*
have *R5*: *C* ≠ *C0*
using *P1 perp-distinct* **by** *blast*
have *R6*: *Col B′ C0′ C0′*
by (*simp add*: *col-trivial-2*)
have ¬ *Col B′ C0′ P*
using *NCol-cases R2 R4 assms(2) col-transitivity-1 out-col* **by** *blast*
then have ∃ *C′*. *Per C′ C0′ B′* ∧
Cong C′ C0′ C C0 ∧ *B′ C0′ OS C′ P* **using** *R4 R5 R6 ex-per-cong* **by** *blast*
thus *?thesis* **by** *auto*
qed
then obtain *C′* **where** *R7*: *Per C′ C0′ B′* ∧
Cong C′ C0′ C C0 ∧ *B′ C0′ OS C′ P* **by** *auto*
then have *R8*: *C0 B C Cong3 C0′ B′ C′*
by (*meson Cong3-def P1 R2 col-trivial-2 l10-12 l8-16-1 not-col-permutation-2 not-cong-2143 not-cong-4321*)
have *R9*: *A B C CongA A′ B′ C′*
proof −
have *S1*: *C0 B C CongA C0′ B′ C′*
by (*metis P4 R8 assms(1) cong3-conga not-col-distincts*)

have *S3*: *B Out C C*
using *assms*(*1*) *not-col-distincts out-trivial* **by** *force*
have *B′* ≠ *C′*
using *R8 assms*(*1*) *cong3-diff2 not-col-distincts* **by** *blast*
then have *B′ Out C′ C′*
using *out-trivial* **by** *auto*
thus *?thesis*
using *S1 R1 S3 R2 l11-10* **by** *blast*
qed
have *B′ A′ OS C′ P*
proof −
have *T1*: *Col B′ C0′ A′*
by (*meson NCol-cases R2 Tarski-neutral-dimensionless.out-col Tarski-neutral-dimensionless-axioms*)
have *B′* ≠ *A′*
using *assms*(*2*) *col-trivial-1* **by** *auto*
thus *?thesis*
using *T1 R7 col-one-side* **by** *blast*
qed
then have *A′ B′ OS C′ P*
by (*simp add*: *invert-one-side*)
thus *?thesis*
using *R9* **by** *blast*
next
assume *U1*: ¬ *B Out A C0*
then have *U2*: *Bet A B C0*
using *NCol-perm P1 or-bet-out* **by** *blast*
obtain *C0′* **where** *U3*: *Bet A′ B′ C0′* ∧ *Cong B′ C0′ B C0*
using *segment-construction* **by** *blast*
have *U4*: ∃ *C′*. *Per C′ C0′ B′* ∧ *Cong C′ C0′ C C0* ∧ *B′ C0′ OS C′ P*
proof −
have *V2*: *C* ≠ *C0*
using *Col-cases P1 assms*(*1*) **by** *blast*
have *B′* ≠ *C0′*
using *P4 U3 cong-diff-3* **by** *blast*
then have ¬ *Col B′ C0′ P*
using *Col-def U3 assms*(*2*) *col-transitivity-1* **by** *blast*
thus *?thesis* **using** *ex-per-cong*
using *V2 not-col-distincts* **by** *blast*
qed
then obtain *C′* **where** *U5*: *Per C′ C0′ B′* ∧ *Cong C′ C0′ C C0* ∧ *B′ C0′ OS C′ P*
by *blast*
have *U98*: *A B C CongA A′ B′ C′*
proof −
have *X1*: *C0 B C Cong3 C0′ B′ C′*
proof −
have *X2*: *Cong C0 B C0′ B′*
using *Cong-cases U3* **by** *blast*
have *X3*: *Cong C0 C C0′ C′*
using *U5 not-cong-4321* **by** *blast*
have *Cong B C B′ C′*
proof −
have *Y1*: *Per C C0 B*
using *P1 col-trivial-3 l8-16-1* **by** *blast*
have *Cong C C0 C′ C0′*
using *U5 not-cong-3412* **by** *blast*
thus *?thesis*
using *Cong-perm Y1 U5 X2 l10-12* **by** *blast*
qed
thus *?thesis*
by (*simp add*: *Cong3-def X2 X3*)
qed
have *X22*: *Bet C0 B A*
using *U2 between-symmetry* **by** *blast*
have *X24*: *Bet C0′ B′ A′*
using *Bet-cases U3* **by** *blast*
have *A′* ≠ *B′*

```
        using assms(2) not-col-distincts by blast
      thus ?thesis
        by (metis P4 X1 X22 X24 assms(1) cong3-conga l11-13 not-col-distincts)
    qed
    have A' B' OS C' P
    proof -
      have B' A' OS C' P
      proof -
        have W1: Col B' C0' A'
          by (simp add: Col-def U3)
        have B' ≠ A'
          using assms(2) not-col-distincts by blast
        thus ?thesis
          using W1 U5 col-one-side by blast
      qed
      thus ?thesis
        by (simp add: invert-one-side)
    qed
    thus ?thesis
      using U98 by blast
  qed
  thus ?thesis by auto
 qed
 thus ?thesis by auto
qed

lemma angle-construction-2:
  assumes A ≠ B and
    B ≠ C and
    ¬ Col A' B' P
  shows ∃ C'. (A B C CongA A' B' C' ∧ (A' B' OS C' P ∨ Col A' B' C'))
  by (metis Col-def angle-construction-1 assms(1) assms(2) assms(3) col-trivial-3 conga-line l11-21-b or-bet-out out-trivial
point-construction-different)

lemma ex-conga-ts:
  assumes ¬ Col A B C and
    ¬ Col A' B' P
  shows ∃ C'. A B C CongA A' B' C' ∧ A' B' TS C' P
proof -
  obtain P' where P1: A' Midpoint P P'
    using symmetric-point-construction by blast
  have P2: ¬ Col A' B' P'
    by (metis P1 assms(2) col-transitivity-1 midpoint-col midpoint-distinct-2 not-col-distincts)
  obtain C' where P3:
    A B C CongA A' B' C' ∧ A' B' OS C' P'
    using P2 angle-construction-1 assms(1) by blast
  have A' B' TS P' P
    using P1 P2 assms(2) bet--ts l9-2 midpoint-bet not-col-distincts by auto
  thus ?thesis
    using P3 l9-8-2 one-side-symmetry by blast
qed

lemma l11-15:
  assumes ¬ Col A B C and
    ¬ Col D E P
  shows
    ∃ F. (A B C CongA D E F ∧ E D OS F P) ∧
       (∀ F1 F2. ((A B C CongA D E F1 ∧ E D OS F1 P) ∧
                  (A B C CongA D E F2 ∧ E D OS F2 P))
    ⟶ E Out F1 F2)
proof -
  obtain F where P1: A B C CongA D E F ∧  D E OS F P
    using angle-construction-1 assms(1) assms(2) by blast
  then have P2: A B C CongA D E F ∧ E D OS F P
    using invert-one-side by blast
  have (∀ F1 F2. ((A B C CongA D E F1 ∧ E D OS F1 P) ∧
```

(*A B C CongA D E F2* ∧ *E D OS F2 P*)) ⟶ *E Out F1 F2*)
proof −
{
fix *F1 F2*
assume *P3*: ((*A B C CongA D E F1* ∧ *E D OS F1 P*) ∧
(*A B C CongA D E F2* ∧ *E D OS F2 P*))
then have *P4*: *A B C CongA D E F1* **by** *simp*
have *P5*: *E D OS F1 P* **using** *P3* **by** *simp*
have *P6*: *A B C CongA D E F2* **using** *P3* **by** *simp*
have *P7*: *E D OS F2 P* **using** *P3* **by** *simp*
have *P8*: *D E F1 CongA D E F2*
using *P4 conga-sym P6 conga-trans* **by** *blast*
have *D E OS F1 F2*
using *P5 P7 invert-one-side one-side-symmetry one-side-transitivity* **by** *blast*
then have *E Out F1 F2* **using** *P8 conga-os--out*
by (*meson P3 conga-sym conga-trans*)
}
thus *?thesis* **by** *auto*
qed
thus *?thesis*
using *P2* **by** *blast*
qed

lemma *l11-19*:
assumes *Per A B P1* **and**
Per A B P2 **and**
A B OS P1 P2
shows *B Out P1 P2*
proof *cases*
assume *Col A B P1*
thus *?thesis*
using *assms*(*3*) *col123--nos* **by** *blast*
next
assume *P1*: ¬ *Col A B P1*
have *B Out P1 P2*
proof *cases*
assume *Col A B P2*
thus *?thesis*
using *assms*(*3*) *one-side-not-col124* **by** *blast*
next
assume *P2*: ¬ *Col A B P2*
obtain *x* **where** *A B P1 CongA A B x* ∧ *B A OS x P2* ∧
(∀ *F1 F2*. ((*A B P1 CongA A B F1* ∧ *B A OS F1 P2*) ∧
(*A B P1 CongA A B F2* ∧ *B A OS F2 P2*))⟶ *B Out F1 F2*)
using *P1 P2 l11-15* **by** *blast*
thus *?thesis*
by (*metis P1 P2 assms*(*1*) *assms*(*2*) *assms*(*3*) *conga-os--out l11-16 not-col-distincts*)
qed
thus *?thesis*
by *simp*
qed

lemma *l11-22-bet*:
assumes *Bet A B C* **and**
P′ B′ TS A′ C′ **and**
A B P CongA A′ B′ P′ **and**
P B C CongA P′ B′ C′
shows *Bet A′ B′ C′*
proof −
obtain *C′′* **where** *P1*: *Bet A′ B′ C′′* ∧ *Cong B′ C′′ B C*
using *segment-construction* **by** *blast*
have *P2*: *C B P CongA C′′ B′ P′*
by (*metis P1 assms*(*1*) *assms*(*3*) *assms*(*4*) *cong-diff-3 conga-diff2 l11-13*)
have *P3*: *C′′ B′ P′ CongA C′ B′ P′*
by (*meson P2 Tarski-neutral-dimensionless.conga-sym Tarski-neutral-dimensionless-axioms assms*(*4*) *conga-comm conga-trans*)

```
  have P4: B' Out C' C'' ∨ P' B' TS C' C''
  proof -
    have P5: Coplanar P' B' C' C''
      by (meson P1 TS-def assms(2) bet--coplanar coplanar-trans-1 ncoplanar-perm-1 ncoplanar-perm-8 ts--coplanar)
    have P' B' C' CongA P' B' C''
      using P3 conga-comm conga-sym by blast
    thus ?thesis
      by (simp add: P5 conga-cop--or-out-ts)
  qed
  have P6: B' Out C' C'' ⟶ Bet A' B' C'
  proof -
    {
      assume B' Out C' C''
      then have Bet A' B' C'
        using P1 bet-out-out-bet between-exchange4 between-trivial2 col-trivial-3 l6-6 not-bet-out by blast
    }
    thus ?thesis by simp
  qed
  have P' B' TS C' C'' ⟶ Bet A' B' C'
  proof -
    {
      assume P7: P' B' TS C' C''
      then have Bet A' B' C'
      proof cases
        assume Col C' B' P'
        thus ?thesis
          using Col-perm TS-def assms(2) by blast
      next
        assume Q1: ¬ Col C' B' P'
        then have Q2: B' ≠ P'
          using not-col-distincts by blast
        have Q3: B' P' TS A' C''
          by (metis Col-perm P1 TS-def P7 assms(2) col-trivial-3)
        have Q4: B' P' OS C' C''
          using P7 Q3 assms(2) invert-two-sides l9-8-1 l9-9 by blast
        thus ?thesis
          using P7 invert-one-side l9-9 by blast
      qed
    }
    thus ?thesis by simp
  qed
  thus ?thesis using P6 P4 by blast
qed

lemma l11-22a:
  assumes B P TS A C and
    B' P' TS A' C' and
    A B P CongA A' B' P' and
    P B C CongA P' B' C'
  shows A B C CongA A' B' C'
proof -
  have P1: A ≠ B ∧ A' ≠ B' ∧ P ≠ B ∧ P' ≠ B' ∧ C ≠ B ∧ C' ≠ B'
    using assms(3) assms(4) conga-diff1 conga-diff2 conga-diff45 conga-diff56 by auto
  have P2: A ≠ C ∧ A' ≠ C'
    using assms(1) assms(2) not-two-sides-id by blast
  obtain A'' where P3: B' Out A' A'' ∧ Cong B' A'' B A
    using P1 segment-construction-3 by force
  have P4: ¬ Col A B P
    using TS-def assms(1) by blast
  obtain T where P5: Col T B P ∧ Bet A T C
    using TS-def assms(1) by blast
  have A B C CongA A' B' C'
  proof cases
    assume B = T
    thus ?thesis
      by (metis P1 P5 assms(2) assms(3) assms(4) conga-line invert-two-sides l11-22-bet)
```

next
  assume *P6*: $B \neq T$
  have $A\ B\ C\ CongA\ A'\ B'\ C'$
  proof *cases*
    assume *P7A*: *Bet P B T*
    obtain T'' **where** *T1*: $Bet\ P'\ B'\ T'' \wedge Cong\ B'\ T''\ B\ T$
      using *segment-construction* **by** *blast*
    have $\exists\ T''.$
 $Col\ B'\ P'\ T'' \wedge (B'\ Out\ P'\ T'' \longleftrightarrow B\ Out\ P\ T) \wedge Cong\ B'\ T''\ B\ T$
    proof −
      have *T2*: $Col\ B'\ P'\ T''$ **using** *T1*
        by (*simp add*: *bet-col col-permutation-4*)
      have $(B'\ Out\ P'\ T'' \longleftrightarrow B\ Out\ P\ T) \wedge Cong\ B'\ T''\ B\ T$
        using *P7A T1 not-bet-and-out* **by** *blast*
      thus *?thesis* **using** *T2* **by** *blast*
    qed
    then obtain T'' **where** *T3*:
      $Col\ B'\ P'\ T'' \wedge (B'\ Out\ P'\ T'' \longleftrightarrow B\ Out\ P\ T) \wedge Cong\ B'\ T''\ B\ T$ **by** *blast*
    then have *T4*: $B' \neq T''$
      using *P6 cong-diff-3* **by** *blast*
    obtain C'' **where** *T5*: $Bet\ A''\ T''\ C'' \wedge Cong\ T''\ C''\ T\ C$
      using *segment-construction* **by** *blast*
    have *T6*: $A\ B\ T\ CongA\ A'\ B'\ T''$
      by (*smt Out-cases P5 P6 T3 T4 P7A assms*(*3*) *between-symmetry col-permutation-4 conga-comm l11-13 l6-4-1 or-bet-out*)
    then have *T7*: $A\ B\ T\ CongA\ A''\ B'\ T''$
      by (*smt P3 P4 P6 T3 Tarski-neutral-dimensionless.l11-10 Tarski-neutral-dimensionless-axioms bet-out col-trivial-3 cong-diff-3 l5-2 l6-6 not-col-permutation-1 or-bet-out*)
    then have *T8*: $Cong\ A\ T\ A''\ T''$
      using *P3 T3 cong2-conga-cong cong-4321 not-cong-3412* **by** *blast*
    have *T9*: $Cong\ A\ C\ A''\ C''$
      using *P5 T5 T8 cong-symmetry l2-11-b* **by** *blast*
    have *T10*: $Cong\ C\ B\ C''\ B'$
      by (*smt P3 P4 P5 T3 T5 T8 cong-commutativity cong-symmetry five-segment*)
    have $A\ B\ C\ Cong3\ A''\ B'\ C''$
      using *Cong3-def P3 T10 T9 not-cong-2143 not-cong-4321* **by** *blast*
    then have *T11*: $A\ B\ C\ CongA\ A''\ B'\ C''$
      by (*simp add*: *Tarski-neutral-dimensionless.cong3-conga Tarski-neutral-dimensionless-axioms P1*)
    have $C\ B\ T\ Cong3\ C''\ B'\ T''$
      by (*simp add*: *Cong3-def T10 T3 T5 cong-4321 cong-symmetry*)
    then have *T12*: $C\ B\ T\ CongA\ C''\ B'\ T''$
      using *P1 P6 cong3-conga* **by** *auto*
    have *T13*: $P\ B\ C\ CongA\ P'\ B'\ C''$
    proof −
      have *K4*: *Bet T B P*
        using *Bet-perm P7A* **by** *blast*
      have $Bet\ T''\ B'\ P'$
        using *Col-perm P7A T3 l6-6 not-bet-and-out or-bet-out* **by** *blast*
      thus *?thesis*
        using *K4 P1 T12 conga-comm l11-13* **by** *blast*
    qed
    have *T14*: $P'\ B'\ C'\ CongA\ P'\ B'\ C''$
    proof −
      have $P'\ B'\ C'\ CongA\ P\ B\ C$
        by (*simp add*: *assms*(*4*) *conga-sym*)
      thus *?thesis*
        using *T13 conga-trans* **by** *blast*
    qed
    have *T15*: $B'\ Out\ C'\ C'' \vee P'\ B'\ TS\ C'\ C''$
    proof −
      have *K7*: $Coplanar\ P'\ B'\ C'\ C''$
      proof −
        have *K8*: $Coplanar\ A'\ P'\ B'\ C'$
          using *assms*(*2*) *coplanar-perm-14 ts--coplanar* **by** *blast*
        have *K8A*: $Coplanar\ P'\ C''\ B'\ A''$
        proof −

have $Col\ P'\ B'\ T'' \wedge Col\ C''\ A''\ T''$
using *Col-def Col-perm T3 T5* **by** *blast*
then have $Col\ P'\ C''\ T'' \wedge Col\ B'\ A''\ T'' \vee$
$Col\ P'\ B'\ T'' \wedge Col\ C''\ A''\ T'' \vee Col\ P'\ A''\ T'' \wedge Col\ C''\ B'\ T''$ **by** *simp*
thus *?thesis*
using *Coplanar-def* **by** *auto*
qed
then have $Coplanar\ A'\ P'\ B'\ C''$
proof −
have *K9*: $B' \neq A''$
using *P3 out-distinct* **by** *blast*
have $Col\ B'\ A''\ A'$
using *Col-perm P3 out-col* **by** *blast*
thus *?thesis*
using *K8A K9 col-cop--cop coplanar-perm-19* **by** *blast*
qed
thus *?thesis*
by (*meson K8 TS-def assms*(*2*) *coplanar-perm-7 coplanar-trans-1 ncoplanar-perm-2*)
qed
thus *?thesis*
by (*simp add*: *T14 K7 conga-cop--or-out-ts*)
qed
have $A\ B\ C\ CongA\ A'\ B'\ C'$
proof *cases*
assume $B'\ Out\ C'\ C''$
thus *?thesis*
using *P1 P3 T11 l11-10 out-trivial* **by** *blast*
next
assume *W1*: $\neg\ B'\ Out\ C'\ C''$
then have *W1A*: $P'\ B'\ TS\ C'\ C''$ **using** *T15* **by** *simp*
have *W2*: $B'\ P'\ TS\ A''\ C'$
using *P3 assms*(*2*) *col-trivial-1 l9-5* **by** *blast*
then have *W3*: $B'\ P'\ OS\ A''\ C''$
using *T15 W1 invert-two-sides l9-2 l9-8-1* **by** *blast*
have *W4*: $B'\ P'\ TS\ A''\ C''$
proof −
have $\neg\ Col\ A'\ B'\ P'$
using *TS-def assms*(*2*) **by** *auto*
thus *?thesis*
using *Col-perm T3 T5 W3 one-side-chara* **by** *blast*
qed
thus *?thesis*
using *W1A W2 invert-two-sides l9-8-1 l9-9* **by** *blast*
qed
thus *?thesis* **by** *simp*
next
assume *R1*: $\neg\ Bet\ P\ B\ T$
then have *R2*: $B\ Out\ P\ T$
using *Col-cases P5 l6-4-2* **by** *blast*
have *R2A*: $\exists\ T''.\ Col\ B'\ P'\ T'' \wedge (B'\ Out\ P'\ T'' \longleftrightarrow B\ Out\ P\ T) \wedge Cong\ B'\ T''\ B\ T$
proof −
obtain T'' **where** *R3*: $B'\ Out\ P'\ T'' \wedge Cong\ B'\ T''\ B\ T$
using *P1 P6 segment-construction-3* **by** *fastforce*
thus *?thesis*
using *R2 out-col* **by** *blast*
qed
then obtain T'' **where** *R4*: $Col\ B'\ P'\ T'' \wedge (B'\ Out\ P'\ T'' \longleftrightarrow B\ Out\ P\ T) \wedge Cong\ B'\ T''\ B\ T$ **by** *auto*
have *R5*: $B' \neq T''$
using *P6 R4 cong-diff-3* **by** *blast*
obtain C'' **where** *R6*: $Bet\ A''\ T''\ C'' \wedge Cong\ T''\ C''\ T\ C$
using *segment-construction* **by** *blast*
have *R7*: $A\ B\ T\ CongA\ A'\ B'\ T''$
using *P1 R2 R4 assms*(*3*) *l11-10 l6-6 out-trivial* **by** *auto*
have *R8*: $A\ B\ T\ CongA\ A''\ B'\ T''$
using *P3 P4 R2 R4 assms*(*3*) *l11-10 l6-6 not-col-distincts out-trivial* **by** *blast*
have *R9*: $Cong\ A\ T\ A''\ T''$

using *Cong-cases P3 R4 R8 cong2-conga-cong* **by** *blast*
have *R10*: *Cong A C A'' C''*
using *P5 R6 R9 l2-11-b not-cong-3412* **by** *blast*
have *R11*: *Cong C B C'' B'*
by (*smt P3 P4 P5 R4 R6 R9 cong-commutativity cong-symmetry five-segment*)
have *A B C Cong3 A'' B' C''*
by (*simp add*: *Cong3-def P3 R10 R11 cong-4321 cong-commutativity*)
then have *R12*: *A B C CongA A'' B' C''*
using *P1* **by** (*simp add*: *cong3-conga*)
have *C B T Cong3 C'' B' T''*
using *Cong3-def R11 R4 R6 not-cong-3412 not-cong-4321* **by** *blast*
then have *R13*: *C B T CongA C'' B' T''*
using *P1 P6 Tarski-neutral-dimensionless.cong3-conga Tarski-neutral-dimensionless-axioms* **by** *fastforce*
have *R13A*: ¬ *Col A' B' P'*
using *TS-def assms*(*2*) **by** *blast*
have *R14*: *B' Out C' C''* ∨ *P' B' TS C' C''*
proof −
have *S1*: *Coplanar P' B' C' C''*
proof −
have *T1*: *Coplanar A' P' B' C'*
using *assms*(*2*) *ncoplanar-perm-14 ts--coplanar* **by** *blast*
have *Coplanar A' P' B' C''*
proof −
have *U6*: *B'* ≠ *A''*
using *P3 out-diff2* **by** *blast*
have *Coplanar P' C'' B' A''*
proof −
have *Col P' B' T''* ∧ *Col C'' A'' T''*
using *Col-def Col-perm R4 R6* **by** *blast*
thus *?thesis* **using** *Coplanar-def* **by** *auto*
qed
thus *?thesis*
by (*meson Col-cases P3 U6 col-cop--cop ncoplanar-perm-21 ncoplanar-perm-6 out-col*)
qed
thus *?thesis*
using *NCol-cases R13A T1 coplanar-trans-1* **by** *blast*
qed
have *P' B' C' CongA P' B' C''*
proof −
have *C B P CongA C'' B' P'*
using *P1 R12 R13 R2 R4 conga-diff56 l11-10 out-trivial* **by** *presburger*
then have *C' B' P' CongA C'' B' P'*
by (*meson Tarski-neutral-dimensionless.conga-trans Tarski-neutral-dimensionless-axioms assms*(*4*) *conga-comm conga-sym*)
thus *?thesis*
by (*simp add*: *conga-comm*)
qed
thus *?thesis*
by (*simp add*: *S1 conga-cop--or-out-ts*)
qed
have *S1*: *B Out A A*
using *P4 not-col-distincts out-trivial* **by** *blast*
have *S2*: *B Out C C*
using *TS-def assms*(*1*) *not-col-distincts out-trivial* **by** *auto*
have *S3*: *B' Out A' A''* **using** *P3* **by** *simp*
have *A B C CongA A' B' C'*
proof *cases*
assume *B' Out C' C''*
thus *?thesis* **using** *S1 S2 S3*
using *R12 l11-10* **by** *blast*
next
assume ¬ *B' Out C' C''*
then have *Z3*: *P' B' TS C' C''* **using** *R14* **by** *simp*
have *Q1*: *B' P' TS A'' C'*
using *S3 assms*(*2*) *l9-5 not-col-distincts* **by** *blast*
have *Q2*: *B' P' OS A'' C''*

```
      proof -
        have B' P' TS C'' C'
        proof -
          have B' P' TS C' C'' using Z3
            using invert-two-sides by blast
          thus ?thesis
            by (simp add: l9-2)
        qed
        thus ?thesis
          using Q1 l9-8-1 by blast
      qed
      have B' P' TS A'' C''
        using Col-perm Q2 R4 R6 one-side-chara by blast
      thus ?thesis
        using Q2 l9-9 by blast
    qed
    thus ?thesis using S1 S2 S3
      using R12 l11-10 by blast
  qed
  thus ?thesis by simp
 qed
 thus ?thesis by simp
qed

lemma l11-22b:
 assumes B P OS A C and
   B' P' OS A' C' and
   A B P CongA A' B' P' and
   P B C CongA P' B' C'
 shows A B C CongA A' B' C'
proof -
 obtain D where P1: Bet A B D ∧ Cong B D A B
   using segment-construction by blast
 obtain D' where P2: Bet A' B' D' ∧ Cong B' D' A' B'
   using segment-construction by blast
 have P3: D B P CongA D' B' P'
 proof -
   have Q3: D ≠ B
     by (metis P1 assms(1) col-trivial-3 cong-diff-3 one-side-not-col124 one-side-symmetry)
   have Q5: D' ≠ B'
     by (metis P2 assms(2) col-trivial-3 cong-diff-3 one-side-not-col124 one-side-symmetry)
   thus ?thesis
     using assms(3) P1 Q3 P2 l11-13 by blast
 qed
 have P5: D B C CongA D' B' C'
 proof -
   have Q1: B P TS D C
     by (metis P1 assms(1) bet--ts col-trivial-3 cong-diff-3 l9-2 l9-8-2 one-side-not-col124 one-side-symmetry)
   have B' P' TS D' C' by (metis Cong-perm P2 assms(2) bet--ts between-cong between-trivial2 l9-2 l9-8-2 one-side-not-col123
point-construction-different ts-distincts)
   thus ?thesis
     using assms(4) Q1 P3 l11-22a by blast
 qed
 have P6: Bet D B A
   using Bet-perm P1 by blast
 have P7: A ≠ B
   using assms(3) conga-diff1 by auto
 have P8: Bet D' B' A'
   using Bet-cases P2 by blast
 have A' ≠ B'
   using assms(3) conga-diff45 by blast
 thus ?thesis
   using P5 P6 P7 P8 l11-13 by blast
qed

lemma l11-22:
```

assumes $((B\ P\ TS\ A\ C \wedge B'\ P'\ TS\ A'\ C') \vee (B\ P\ OS\ A\ C \wedge B'\ P'\ OS\ A'\ C'))$ **and**
$A\ B\ P\ CongA\ A'\ B'\ P'$ **and**
$P\ B\ C\ CongA\ P'\ B'\ C'$
shows $A\ B\ C\ CongA\ A'\ B'\ C'$
by (*meson assms(1) assms(2) assms(3) l11-22a l11-22b*)

lemma *l11-24*:
assumes *P InAngle A B C*
shows *P InAngle C B A*
using *Bet-cases InAngle-def assms* **by** *auto*

lemma *col-in-angle*:
assumes $A \neq B$ **and**
$C \neq B$ **and**
$P \neq B$ **and**
$B\ Out\ A\ P \vee B\ Out\ C\ P$
shows *P InAngle A B C*
by (*meson InAngle-def assms(1) assms(2) assms(3) assms(4) between-trivial between-trivial2*)

lemma *out321--inangle*:
assumes $C \neq B$ **and**
B Out A P
shows *P InAngle A B C*
using *assms(1) assms(2) col-in-angle out-distinct* **by** *auto*

lemma *inangle1123*:
assumes $A \neq B$ **and**
$C \neq B$
shows *A InAngle A B C*
by (*simp add: assms(1) assms(2) out321--inangle out-trivial*)

lemma *out341--inangle*:
assumes $A \neq B$ **and**
B Out C P
shows *P InAngle A B C*
using *assms(1) assms(2) col-in-angle out-distinct* **by** *auto*

lemma *inangle3123*:
assumes $A \neq B$ **and**
$C \neq B$
shows *C InAngle A B C*
by (*simp add: assms(1) assms(2) inangle1123 l11-24*)

lemma *in-angle-two-sides*:
assumes $\neg$ *Col B A P* **and**
$\neg$ *Col B C P* **and**
P InAngle A B C
shows *P B TS A C*
by (*metis InAngle-def TS-def assms(1) assms(2) assms(3) not-col-distincts not-col-permutation-1 out-col*)

lemma *in-angle-out*:
assumes *B Out A C* **and**
P InAngle A B C
shows *B Out A P*
by (*metis InAngle-def assms(1) assms(2) not-bet-and-out out2-bet-out*)

lemma *col-in-angle-out*:
assumes *Col B A P* **and**
$\neg$ *Bet A B C* **and**
P InAngle A B C
shows *B Out A P*
proof −
obtain *X* **where** *P1*: $Bet\ A\ X\ C \wedge (X = B \vee B\ Out\ X\ P)$
using *InAngle-def assms(3)* **by** *auto*
have *B Out A P*
proof *cases*

```
    assume X = B
    thus ?thesis
      using P1 assms(2) by blast
  next
    assume P2: X ≠ B
    thus ?thesis
    proof −
      have f1: Bet B A P ∨ A Out B P
        by (meson assms(1) l6-4-2)
      have f2: B Out X P
        using P1 P2 by blast
      have f3: (Bet B P A ∨Bet B A P) ∨Bet P B A
        using f1 by (meson Bet-perm Out-def)
      have f4: Bet B P X ∨Bet P X B
        using f2 by (meson Bet-perm Out-def)
      then have f5: ((Bet B P X ∨Bet X B A) ∨Bet B P A) ∨Bet B A P
        using f3 by (meson between-exchange3)
      have ∀ p. Bet p X C ∨ ¬Bet A p X
        using P1 between-exchange3 by blast
      then have f6: (P = B ∨Bet B A P) ∨Bet B P A
        using f5 f3 by (meson Bet-perm P2 assms(2) outer-transitivity-between2)
      have f7: Bet C X A
        using Bet-perm P1 by blast
      have P ≠ B
        using f2 by (simp add: Out-def)
      moreover
      { assume Bet B B C
        then have A ≠ B
          using assms(2) by blast }
      ultimately have A ≠ B
        using f7 f4 f1 by (meson Bet-perm Out-def P2 between-exchange3 outer-transitivity-between2)
      thus ?thesis
        using f6 f2 by (simp add: Out-def)
    qed
  qed
  thus ?thesis by blast
qed

lemma l11-25-aux:
  assumes P InAngle A B C and
    ¬ Bet A B C and
    B Out A′ A
  shows P InAngle A′ B C
proof −
  have P1: Bet B A′ A ∨ Bet B A A′
    using Out-def assms(3) by auto
  {
    assume P2: Bet B A′ A
    obtain X where P3: Bet A X C ∧ (X = B ∨ B Out X P)
      using InAngle-def assms(1) by auto
    obtain T where P4: Bet A′ T C ∧ Bet X T B
      using Bet-perm P2 P3 inner-pasch by blast
    {
      assume X = B
      then have P InAngle A′ B C
        using P3 assms(2) by blast
    }
    {
      assume B Out X P
      then have P InAngle A′ B C
        by (metis InAngle-def P4 assms(1) assms(3) bet-out-1 l6-7 out-diff1)
    }
    then have P InAngle A′ B C
      using P3 ‹X = B ⟹ P InAngle A′ B C› by blast
  }
  {
```

```
    assume Q0: Bet B A A′
    obtain X where Q1: Bet A X C ∧ (X = B ∨ B Out X P)
      using InAngle-def assms(1) by auto
    {
      assume X = B
      then have P InAngle A′ B C
        using Q1 assms(2) by blast
    }
    {
      assume Q2: B Out X P
      obtain T where Q3: Bet A′ T C ∧ Bet B X T
        using Bet-perm Q1 Q0 outer-pasch by blast
      then have P InAngle A′ B C
        by (metis InAngle-def Q0 Q2 assms(1) bet-out l6-6 l6-7 out-diff1)
    }
    then have P InAngle A′ B C
      using ‹X = B ⟹ P InAngle A′ B C› Q1 by blast
  }
  thus ?thesis
    using P1 ‹Bet B A′ A ⟹ P InAngle A′ B C› by blast
qed

lemma l11-25:
  assumes P InAngle A B C and
    B Out A′ A and
    B Out C′ C and
    B Out P′ P
  shows P′ InAngle A′ B C′
proof cases
  assume Bet A B C
  thus ?thesis
    by (metis Bet-perm InAngle-def assms(2) assms(3) assms(4) bet-out--bet l6-6 out-distinct)
next
  assume P1: ¬ Bet A B C
  have P2: P InAngle A′ B C
    using P1 assms(1) assms(2) l11-25-aux by blast
  have P3: P InAngle A′ B C′
  proof −
    have P InAngle C′ B A′ using l11-25-aux
      using Bet-perm P1 P2 assms(2) assms(3) bet-out--bet l11-24 by blast
    thus ?thesis using l11-24 by blast
  qed
  obtain X where P4: Bet A′ X C′ ∧ (X = B ∨ B Out X P)
    using InAngle-def P3 by auto
  {
    assume X = B
    then have P′ InAngle A′ B C′
      using InAngle-def P3 P4 assms(4) out-diff1 by auto
  }
  {
    assume B Out X P
    then have P′ InAngle A′ B C′
    proof −
      have ∀ p. B Out p P′ ∨ ¬ B Out p P
        by (meson Out-cases assms(4) l6-7)
      thus ?thesis
        by (metis (no-types) InAngle-def P3 assms(4) out-diff1)
    qed
  }
  thus ?thesis
    using InAngle-def P4 assms(2) assms(3) assms(4) out-distinct by auto
qed

lemma inangle-distincts:
  assumes P InAngle A B C
  shows A ≠ B ∧ C ≠ B ∧ P ≠ B
```

using *InAngle-def assms* **by** *auto*

lemma *segment-construction-0*:
  shows $\exists\ B'.\ Cong\ A'\ B'\ A\ B$
  using *segment-construction* **by** *blast*

lemma *angle-construction-3*:
  assumes $A \neq B$ **and**
    $C \neq B$ **and**
    $A' \neq B'$
  shows $\exists\ C'.\ A\ B\ C\ CongA\ A'\ B'\ C'$
  by (*metis angle-construction-2 assms*(*1*) *assms*(*2*) *assms*(*3*) *not-col-exists*)

lemma *l11-28*:
  assumes $A\ B\ C\ Cong3\ A'\ B'\ C'$ **and**
    $Col\ A\ C\ D$
  shows $\exists\ D'.\ (Cong\ A\ D\ A'\ D' \wedge Cong\ B\ D\ B'\ D' \wedge Cong\ C\ D\ C'\ D')$
proof *cases*
  assume *P1*: $A = C$
  have $\exists\ D'.\ (Cong\ A\ D\ A'\ D' \wedge Cong\ B\ D\ B'\ D' \wedge Cong\ C\ D\ C'\ D')$
  proof *cases*
    assume $A = B$
    thus *?thesis*
      by (*metis P1 assms*(*1*) *cong3-diff cong3-symmetry cong-3-swap-2 not-cong-3421 segment-construction-0*)
  next
    assume $A \neq B$
    have $\exists\ D'.\ (Cong\ A\ D\ A'\ D' \wedge Cong\ B\ D\ B'\ D' \wedge Cong\ C\ D\ C'\ D')$
    proof *cases*
      assume $A = D$
      thus *?thesis*
        using *Cong3-def P1 assms*(*1*) *cong-trivial-identity* **by** *blast*
    next
      assume $A \neq D$
      have $\exists\ D'.\ (Cong\ A\ D\ A'\ D' \wedge Cong\ B\ D\ B'\ D' \wedge Cong\ C\ D\ C'\ D')$
      proof *cases*
        assume $B = D$
        thus *?thesis*
          using *Cong3-def assms*(*1*) *cong-3-swap-2 cong-trivial-identity* **by** *blast*
      next
        assume *Q1*: $B \neq D$
        obtain D'' **where** *Q2*: $B\ A\ D\ CongA\ B'\ A'\ D''$
          by (*metis* ‹$A \neq B$› ‹$A \neq D$› *angle-construction-3 assms*(*1*) *cong3-diff*)
        obtain D' **where** *Q3*: $A'\ Out\ D''\ D' \wedge Cong\ A'\ D'\ A\ D$
          by (*metis Q2* ‹$A \neq D$› *conga-diff56 segment-construction-3*)
        have *Q5*: $Cong\ A\ D\ A'\ D'$
          using *Q3 not-cong-3412* **by** *blast*
        have $B\ A\ D\ CongA\ B'\ A'\ D'$
          using *Q2 Q3* ‹$A \neq B$› ‹$A \neq D$› *conga-diff45 l11-10 l6-6 out-trivial* **by** *auto*
        then have $Cong\ B\ D\ B'\ D'$
          using *Cong3-def Cong-perm Q5 assms*(*1*) *cong2-conga-cong* **by** *blast*
        thus *?thesis*
          using *Cong3-def P1 Q5 assms*(*1*) *cong-reverse-identity* **by** *blast*
      qed
      thus *?thesis* **by** *simp*
    qed
    thus *?thesis* **by** *simp*
  qed
  thus *?thesis* **by** *simp*
next
  assume *Z1*: $A \neq C$
  have $\exists\ D'.\ (Cong\ A\ D\ A'\ D' \wedge Cong\ B\ D\ B'\ D' \wedge Cong\ C\ D\ C'\ D')$
  proof *cases*
    assume $A = D$
    thus *?thesis*
      using *Cong3-def Cong-perm assms*(*1*) *cong-trivial-identity* **by** *blast*
  next

```
assume A ≠ D
{
  assume Bet A C D
  obtain D′ where W1: Bet A′ C′ D′ ∧ Cong C′ D′ C D
    using segment-construction by blast
  have W2: Cong A D A′ D′
    by (meson Cong3-def W1 ‹Bet A C D› assms(1) cong-symmetry l2-11-b)
  have W3: Cong B D B′ D′
  proof −
    have X1: Cong C D C′ D′
      using W1 not-cong-3412 by blast
    have Cong C B C′ B′
      using Cong3-def assms(1) cong-commutativity by presburger
    then have W4: A C D B OFSC A′ C′ D′ B′
      using Cong3-def OFSC-def W1 X1 ‹Bet A C D› assms(1) by blast
    have Cong D B D′ B′
      using W4 ‹A ≠ C› five-segment-with-def by blast
    thus ?thesis
      using Z1 not-cong-2143 by blast
  qed
  have Cong C D C′ D′
    by (simp add: W1 cong-symmetry)
  then have ∃ D′. (Cong A D A′ D′ ∧ Cong B D B′ D′ ∧ Cong C D C′ D′)
    using W2 W3 by blast
}
{
  assume W3B: Bet C D A
  then obtain D′ where W4A: Bet A′ D′ C′ ∧ A D C Cong3 A′ D′ C′
    using Bet-perm Cong3-def assms(1) l4-5 by blast
  have W5: Cong A D A′ D′
    using Cong3-def W4A by blast
  have A D C B IFSC A′ D′ C′ B′
    by (meson Bet-perm Cong3-def Cong-perm IFSC-def W4A W3B assms(1))
  then have Cong D B D′ B′
    using l4-2 by blast
  then have W6: Cong B D B′ D′
    using Cong-perm by blast
  then have Cong C D C′ D′
    using Cong3-def W4A not-cong-2143 by blast
  then have ∃ D′. (Cong A D A′ D′ ∧ Cong B D B′ D′ ∧ Cong C D C′ D′)
    using W5 W6 by blast
}
{
  assume W7: Bet D A C
  obtain D′ where W7A: Bet C′ A′ D′ ∧ Cong A′ D′ A D
    using segment-construction by blast
  then have W8: Cong A D A′ D′
    using Cong-cases by blast
  have C A D B OFSC C′ A′ D′ B′
    by (meson Bet-perm Cong3-def Cong-perm OFSC-def W7 W7A assms(1))
  then have Cong D B D′ B′
    using Z1 five-segment-with-def by auto
  then have w9: Cong B D B′ D′
    using Cong-perm by blast
  have Cong C D C′ D′
  proof −
    have L1: Bet C A D
      using Bet-perm W7 by blast
    have L2: Bet C′ A′ D′
      using Bet-perm W7
      using W7A by blast
    have Cong C A C′ A′ using assms(1)
      using Cong3-def assms(1) not-cong-2143 by blast
    thus ?thesis using l2-11
      using L1 L2 W8 l2-11 by blast
  qed
```

```
    then have ∃ D'. (Cong A D A' D' ∧ Cong B D B' D' ∧ Cong C D C' D')
      using W8 w9 by blast
    }
    thus ?thesis
      using Bet-cases ‹Bet A C D ⟹ ∃D'. Cong A D A' D' ∧ Cong B D B' D' ∧ Cong C D C' D'› ‹Bet C D A ⟹
∃D'. Cong A D A' D' ∧ Cong B D B' D' ∧ Cong C D C' D'› assms(2) third-point by blast
  qed
  thus ?thesis
    by blast
qed

lemma bet-conga--bet:
  assumes Bet A B C and
    A B C CongA A' B' C'
  shows Bet A' B' C'
proof −
  obtain A0 C0 A1 C1 where P1:
 Bet B A A0 ∧Cong A A0 B' A' ∧
 Bet B C C0 ∧Cong C C0 B' C' ∧
 Bet B' A' A1 ∧Cong A' A1 B A ∧
 Bet B' C' C1 ∧Cong C' C1 B C ∧
 Cong A0 C0 A1 C1 using CongA-def assms(2)
    by auto
  have Bet C B A0 using P1 outer-transitivity-between
    by (metis assms(1) assms(2) between-symmetry conga-diff1)
  then have Bet A0 B C
    using Bet-cases by blast
  then have P2: Bet A0 B C0
    using P1 assms(2) conga-diff2 outer-transitivity-between by blast
  have P3: A0 B C0 Cong3 A1 B' C1
  proof −
    have Q1: Cong A0 B A1 B'
      by (meson Bet-cases P1 l2-11-b not-cong-1243 not-cong-4312)
    have Q3: Cong B C0 B' C1
      using P1 between-symmetry cong-3421 l2-11-b not-cong-1243 by blast
    thus ?thesis
      by (simp add: Cong3-def Q1 P1)
  qed
  then have P4: Bet A1 B' C1 using P2 l4-6 by blast
  then have Bet A' B' C1
    using P1 Bet-cases between-exchange3 by blast
  thus ?thesis using between-inner-transitivity P1 by blast
qed

lemma in-angle-one-side:
  assumes ¬ Col A B C and
    ¬ Col B A P and
    P InAngle A B C
  shows A B OS P C
proof −
  obtain X where P1: Bet A X C ∧ (X = B ∨ B Out X P)
    using InAngle-def assms(3) by auto
  {
    assume X = B
    then have  A B OS P C
      using P1 assms(1) bet-col by blast
  }
  {
    assume P2: B Out X P
    obtain C' where P2A: Bet C A C' ∧ Cong A C' C A
      using segment-construction by blast
    have A B TS X C'
    proof −
      have Q1: ¬ Col X A B
        by (metis Col-def P1 assms(1) assms(2) col-transitivity-2 out-col)
      have Q2 :¬ Col C' A B
```

by (*metis Col-def Cong-perm P2A assms*(*1*) *cong-diff l6-16-1*)
have $\exists$ *T*. *Col T A B* $\wedge$ *Bet X T C'*
using *Bet-cases P1 P2A between-exchange3 col-trivial-1* **by** *blast*
thus *?thesis*
by (*simp add*: *Q1 Q2 TS-def*)
qed
then have *P3*: *A B TS P C'*
using *P2 col-trivial-3 l9-5* **by** *blast*
then have *A B TS C C'*
by (*smt P1 P2 bet-out bet-ts--os between-trivial col123--nos col-trivial-3 invert-two-sides l6-6 l9-2 l9-5*)
then have *A B OS P C*
using *OS-def P3* **by** *blast*
}
thus *?thesis*
using *P1* ‹*X* = *B* $\Longrightarrow$ *A B OS P C*› **by** *blast*
qed

lemma *inangle-one-side*:
assumes ¬ *Col A B C* **and**
¬ *Col A B P* **and**
¬ *Col A B Q* **and**
P InAngle A B C **and**
Q InAngle A B C
shows *A B OS P Q*
by (*meson assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *in-angle-one-side not-col-permutation-4 one-side-symmetry one-side-transitivity*)

lemma *inangle-one-side2*:
assumes ¬ *Col A B C* **and**
¬ *Col A B P* **and**
¬ *Col A B Q* **and**
¬ *Col C B P* **and**
¬ *Col C B Q* **and**
P InAngle A B C **and**
Q InAngle A B C
shows *A B OS P Q* $\wedge$ *C B OS P Q*
by (*meson assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *assms*(*6*) *assms*(*7*) *inangle-one-side l11-24 not-col-permutation-3*)

lemma *col-conga-col*:
assumes *Col A B C* **and**
A B C CongA D E F
shows *Col D E F*
proof −
{
assume *Bet A B C*
then have *Col D E F*
using *Col-def assms*(*2*) *bet-conga--bet* **by** *blast*
}
{
assume *Bet B C A*
then have *Col D E F*
by (*meson Col-perm Tarski-neutral-dimensionless.l11-21-a Tarski-neutral-dimensionless-axioms* ‹*Bet A B C* $\Longrightarrow$ *Col D E F*› *assms*(*1*) *assms*(*2*) *or-bet-out out-col*)
}
{
assume *Bet C A B*
then have *Col D E F*
by (*meson Col-perm Tarski-neutral-dimensionless.l11-21-a Tarski-neutral-dimensionless-axioms* ‹*Bet A B C* $\Longrightarrow$ *Col D E F*› *assms*(*1*) *assms*(*2*) *or-bet-out out-col*)
}
thus *?thesis*
using *Col-def* ‹*Bet A B C* $\Longrightarrow$ *Col D E F*› ‹*Bet B C A* $\Longrightarrow$ *Col D E F*› *assms*(*1*) **by** *blast*
qed

lemma *ncol-conga-ncol*:
assumes ¬ *Col A B C* **and**

```
    A B C CongA D E F
  shows ¬ Col D E F
  using assms(1) assms(2) col-conga-col conga-sym by blast

lemma angle-construction-4:
  assumes A ≠ B and
    C ≠ B and
    A′ ≠ B′
  shows ∃ C′. (A B C CongA A′ B′ C′ ∧ Coplanar A′ B′ C′ P)
proof cases
  assume Col A′ B′ P
  thus ?thesis
    using angle-construction-3 assms(1) assms(2) assms(3) ncop--ncols by blast
next
  assume ¬ Col A′ B′ P
  {
    assume Col A B C
    then have ∃ C′. (A B C CongA A′ B′ C′ ∧ Coplanar A′ B′ C′ P)
      by (meson angle-construction-3 assms(1) assms(2) assms(3) col--coplanar col-conga-col)
  }
  {
    assume ¬ Col A B C
    then obtain C′ where A B C CongA A′ B′ C′ ∧ A′ B′ OS C′ P
      using ‹¬ Col A′ B′ P› angle-construction-1 by blast
    then have  ∃ C′. (A B C CongA A′ B′ C′ ∧ Coplanar A′ B′ C′ P)
      using os--coplanar by blast
  }
  thus ?thesis
    using ‹Col A B C ⟹ ∃ C′. A B C CongA A′ B′ C′ ∧ Coplanar A′ B′ C′ P› by blast
qed

lemma lea-distincts:
  assumes A B C LeA D E F
  shows A≠B ∧ C≠B ∧ D≠E ∧ F≠E
 by (metis (no-types) LeA-def Tarski-neutral-dimensionless.conga-diff1 Tarski-neutral-dimensionless.conga-diff2 Tarski-neutral-dimens
assms inangle-distincts)

lemma l11-29-a:
  assumes A B C LeA D E F
  shows ∃ Q. (C InAngle A B Q ∧ A B Q CongA D E F)
proof −
  obtain P where P1: P InAngle D E F ∧ A B C CongA D E P
    using LeA-def assms by blast
  then have P2: E ≠ D ∧ B ≠ A ∧ E ≠ F ∧ E ≠ P ∧ B ≠ C
    using conga-diff1 conga-diff2 inangle-distincts by blast
  then have P3: A ≠ B ∧ C ≠ B by blast
  {
    assume Q1: Bet A B C
    then have Q2: Bet D E P
      by (meson P1 Tarski-neutral-dimensionless.bet-conga--bet Tarski-neutral-dimensionless-axioms)
    have Q3: C InAngle A B C
      by (simp add: P3 inangle3123)
    obtain X where Q4: Bet D X F ∧ (X = E ∨ E Out X P)
      using InAngle-def P1 by auto
    have A B C CongA D E F
    proof −
      {
        assume R1: X = E
        have R2: Bet E F P ∨ Bet E P F
        proof −
          have R3: D ≠ E using P2 by blast
          have Bet D E F
            using Col-def Col-perm P1 Q2 col-in-angle-out not-bet-and-out by blast
          have Bet D E P using Q2 by blast
          thus ?thesis using l5-2
            using R3 ‹Bet D E F› by blast
```

qed
then have $A\ B\ C\ CongA\ D\ E\ F$
by (*smt P1 P2 bet-out l11-10 l6-6 out-trivial*)
}
{
assume *S1*: $E\ Out\ X\ P$

have *S2*: $E\ Out\ P\ F$
proof −
{
assume $Bet\ E\ X\ P$
then have $E\ Out\ P\ F$
proof −
have $Bet\ E\ X\ F$
by (*meson Bet-perm Q2 Q4* ‹$Bet\ E\ X\ P$› *between-exchange3*)
thus *?thesis*
by (*metis Bet-perm S1 bet2--out between-equality-2 between-trivial2 out2-bet-out out-diff1*)
qed
}
{
assume $Bet\ E\ P\ X$
then have $E\ Out\ P\ F$
by (*smt Bet-perm Q2 Q4 S1 bet-out-1 between-exchange3 not-bet-and-out outer-transitivity-between2*)
}
thus *?thesis*
using *Out-def S1* ‹$Bet\ E\ X\ P \Longrightarrow E\ Out\ P\ F$› by *blast*
qed

then have $A\ B\ C\ CongA\ D\ E\ F$
by (*metis Bet-perm P2 Q1 Q2 bet-out--bet conga-line*)
}
thus *?thesis*
using *Q4* ‹$X = E \Longrightarrow A\ B\ C\ CongA\ D\ E\ F$› by *blast*
qed
then have $\exists\ Q.\ (C\ InAngle\ A\ B\ Q \wedge A\ B\ Q\ CongA\ D\ E\ F)$
using *conga-diff1 conga-diff2 inangle3123* by *blast*
}
{
assume $B\ Out\ A\ C$
obtain Q where $D\ E\ F\ CongA\ A\ B\ Q$
by (*metis P2 angle-construction-3*)

then have $\exists\ Q.\ (C\ InAngle\ A\ B\ Q \wedge A\ B\ Q\ CongA\ D\ E\ F)$
by (*metis Tarski-neutral-dimensionless.conga-comm Tarski-neutral-dimensionless-axioms* ‹$B\ Out\ A\ C$› *conga-diff1 conga-sym out321--inangle*)
}
{
assume *ZZ*: $\neg\ Col\ A\ B\ C$
have *Z1*: $D \neq E$
using *P2* by *blast*
have *Z2*: $F \neq E$
using *P2* by *blast*
have *Z3*: $Bet\ D\ E\ F \vee E\ Out\ D\ F \vee \neg\ Col\ D\ E\ F$
using *not-bet-out* by *blast*
{
assume $Bet\ D\ E\ F$
obtain Q where *Z4*: $Bet\ A\ B\ Q \wedge Cong\ B\ Q\ E\ F$
using *segment-construction* by *blast*

then have $\exists\ Q.\ (C\ InAngle\ A\ B\ Q \wedge A\ B\ Q\ CongA\ D\ E\ F)$
by (*metis InAngle-def P3 Z1 Z2* ‹$Bet\ D\ E\ F$› *conga-line point-construction-different*)
}
{
assume $E\ Out\ D\ F$
then have *Z5*: $E\ Out\ D\ P$
using *P1 in-angle-out* by *blast*

have *D E P CongA A B C*
  by (*simp add*: *P1 conga-sym*)
then have *Z6*: *B Out A C* **using** *l11-21-a Z5*
  by *blast*

then have ∃ *Q*. (*C InAngle A B Q* ∧ *A B Q CongA D E F*)
  using ‹*B Out A C* ⟹ ∃ *Q*. *C InAngle A B Q* ∧ *A B Q CongA D E F*› **by** *blast*
}
{
assume *W1*: ¬ *Col D E F*
obtain *Q* **where** *W2*: *D E F CongA A B Q* ∧ *A B OS Q C*
  using *W1 ZZ angle-construction-1* **by** *blast*
obtain *DD* **where** *W3*: *E Out D DD* ∧ *Cong E DD B A*
  using *P3 Z1 segment-construction-3* **by** *force*
obtain *FF* **where** *W4*: *E Out F FF* ∧ *Cong E FF B Q*
  by (*metis P2 W2 conga-diff56 segment-construction-3*)
then have *W5*: *P InAngle DD E FF*
  by (*smt Out-cases P1 P2 W3 l11-25 out-trivial*)
obtain *X* **where** *W6*: *Bet DD X FF* ∧ (*X* = *E* ∨ *E Out X P*)
  using *InAngle-def W5* **by** *presburger*
{
  assume *W7*: *X* = *E*
  have *W8*: *Bet D E F*
  proof −
    have *W10*: *E Out DD D*
      by (*simp add*: *W3 l6-6*)
    have *E Out FF F*
      by (*simp add*: *W4 l6-6*)
    thus *?thesis* **using** *W6 W7 W10 bet-out-out-bet* **by** *blast*
  qed
  then have ∃ *Q*. (*C InAngle A B Q* ∧ *A B Q CongA D E F*)
    using ‹*Bet D E F* ⟹ ∃ *Q*. *C InAngle A B Q* ∧ *A B Q CongA D E F*› **by** *blast*
}
{
  assume *V1*: *E Out X P*
  have *B* ≠ *C* ∧ *E* ≠ *X*
    using *P3 V1 out-diff1* **by** *blast*
  then obtain *CC* **where** *V2*: *B Out C CC* ∧ *Cong B CC E X*
    using *segment-construction-3* **by** *blast*
  then have *V3*: *A B CC CongA DD E X*
    by (*smt P1 P2 V1 W3 l11-10 l6-6 out-trivial*)
  have *V4*: *Cong A CC DD X*
  proof −
    have *Cong A B DD E*
      using *W3 not-cong-4321* **by** *blast*
    thus *?thesis*
      using *V2 V3 cong2-conga-cong* **by** *blast*
  qed

  have *V5*: *A B Q CongA DD E FF*
  proof −
    have *U1*: *D E F CongA A B Q*
      by (*simp add*: *W2*)
    then have *U1A*: *A B Q CongA D E F*
      by (*simp add*: *conga-sym*)
    have *U2*: *B Out A A*
      by (*simp add*: *P3 out-trivial*)
    have *U3*: *B Out Q Q*
      using *W2 conga-diff56 out-trivial* **by** *blast*
    have *U4*: *E Out DD D*
      using *W3 l6-6* **by** *blast*
    have *E Out FF F*
      by (*simp add*: *W4 l6-6*)

    thus *?thesis* **using** *l11-10*
      using *U1A U2 U3 U4* **by** *blast*

qed
then have *V6*: *Cong A Q DD FF*
using *Cong-perm W3 W4 cong2-conga-cong* **by** *blast*
have *CC B Q CongA X E FF*
proof −
have *U1*: *B A OS CC Q*
by (*metis* (*no-types*) *V2 W2 col124--nos invert-one-side one-side-symmetry one-side-transitivity out-one-side*)
have *U2*: *E DD OS X FF*
proof −
have ¬ *Col DD E FF*
by (*metis Col-perm OS-def TS-def U1 V5 ncol-conga-ncol*)
then have ¬ *Col E DD X*
by (*metis Col-def V2 V4 W6 ZZ cong-identity l6-16-1 os-distincts out-one-side*)
then have *DD E OS X FF*
by (*metis Col-perm W6 bet-out not-col-distincts one-side-reflexivity out-out-one-side*)
thus *?thesis*
by (*simp add*: *invert-one-side*)
qed
have *CC B A CongA X E DD*
by (*simp add*: *V3 conga-comm*)
thus *?thesis*
using *U1 U2 V5 l11-22b* **by** *blast*
qed
then have *V8*: *Cong CC Q X FF*
using *V2 W4 cong2-conga-cong cong-commutativity not-cong-3412* **by** *blast*
have *V9*: *CC InAngle A B Q*
proof −
have *T2*: $Q \neq B$
using *W2 conga-diff56* **by** *blast*
have *T3*: $CC \neq B$
using *V2 out-distinct* **by** *blast*
have *Bet A CC Q*
proof −
have *T4*: *DD X FF Cong3 A CC Q*
using *Cong3-def V4 V6 V8 not-cong-3412* **by** *blast*
thus *?thesis*
using *W6 l4-6* **by** *blast*
qed
then have ∃ *X0*. *Bet A X0 Q* ∧ (*X0* = *B* ∨ *B Out X0 CC*)
using *out-trivial* **by** *blast*
thus *?thesis*
by (*simp add*: *InAngle-def P3 T2 T3*)
qed
then have *C InAngle A B Q*
using *V2 inangle-distincts l11-25 out-trivial* **by** *blast*
then have ∃ *Q*. (*C InAngle A B Q* ∧ *A B Q CongA D E F*)
using *W2 conga-sym* **by** *blast*
}
then have ∃ *Q*. (*C InAngle A B Q* ∧ *A B Q CongA D E F*)
using *W6* ‹*X* = *E* ⟹ ∃ *Q*. *C InAngle A B Q* ∧ *A B Q CongA D E F*› **by** *blast*
}
then have ∃ *Q*. (*C InAngle A B Q* ∧ *A B Q CongA D E F*)
using *Z3* ‹*E Out D F* ⟹ ∃ *Q*. *C InAngle A B Q* ∧ *A B Q CongA D E F*› ‹*Bet D E F* ⟹ ∃ *Q*. *C InAngle A B Q* ∧ *A B Q CongA D E F*› **by** *blast*
}
thus *?thesis*
using ‹*B Out A C* ⟹ ∃ *Q*. *C InAngle A B Q* ∧ *A B Q CongA D E F*› ‹*Bet A B C* ⟹ ∃ *Q*. *C InAngle A B Q* ∧ *A B Q CongA D E F*› *not-bet-out* **by** *blast*
qed

lemma *in-angle-line*:
assumes $P \neq B$ **and**
$A \neq B$ **and**
$C \neq B$ **and**
Bet A B C
shows *P InAngle A B C*

```
  using InAngle-def assms(1) assms(2) assms(3) assms(4) by auto

lemma l11-29-b:
  assumes ∃ Q. (C InAngle A B Q ∧ A B Q CongA D E F)
  shows A B C LeA D E F
proof −
  obtain Q where P1: C InAngle A B Q ∧ A B Q CongA D E F
    using assms by blast
  obtain X where P2: Bet A X Q ∧ (X = B ∨ B Out X C)
    using InAngle-def P1 by auto
  {
    assume P2A: X = B
    obtain P where P3: A B C CongA D E P
      using angle-construction-3 assms conga-diff45 inangle-distincts by fastforce
    have P InAngle D E F
    proof −
      have O1: Bet D E F
       by (metis (no-types) P1 P2 Tarski-neutral-dimensionless.bet-conga--bet Tarski-neutral-dimensionless-axioms P2A)
      have O2: P ≠ E
        using P3 conga-diff56 by auto
      have O3: D ≠ E
        using P3 conga-diff45 by auto
      have F ≠ E
        using P1 conga-diff56 by blast
      thus ?thesis using in-angle-line
        by (simp add: O1 O2 O3)
    qed
    then have A B C LeA D E F
      using LeA-def P3 by blast
  }
  {
    assume G1: B Out X C
    obtain DD where G2: E Out D DD ∧ Cong E DD B A
      by (metis assms conga-diff1 conga-diff45 segment-construction-3)
    have G3: D ≠ E ∧ DD ≠ E
      using G2 out-diff1 out-diff2 by blast
    obtain FF where G3G: E Out F FF ∧ Cong E FF B Q
      by (metis P1 conga-diff56 inangle-distincts segment-construction-3)
    then have G3A: F ≠ E
      using out-diff1 by blast
    have G3B: FF ≠ E
      using G3G out-distinct by blast
    have G4: Bet A B C ∨ B Out A C ∨ ¬ Col A B C
      using not-bet-out by blast
    {
      assume G5: Bet A B C
      have G6: F InAngle D E F
        by (simp add: G3 G3A inangle3123)
      have A B C CongA D E F
          by (smt Bet-perm G3 G3A G5 Out-def P1 P2 bet-conga--bet between-exchange3 conga-line inangle-distincts
outer-transitivity-between2)
      then have  A B C LeA D E F
        using G6 LeA-def by blast
    }
    {
      assume G8: B Out A C
      have G9: D InAngle D E F
        by (simp add: G3 G3A inangle1123)
      have A B C CongA D E D
        by (simp add: G3 G8 l11-21-b out-trivial)
      then have  A B C LeA D E F using G9 LeA-def by blast
    }
    {
      assume R1: ¬ Col A B C
      have R2: Bet A B Q ∨ B Out A Q ∨ ¬ Col A B Q
        using not-bet-out by blast
```

{
assume *R3*: *Bet A B Q*
obtain *P* **where** *R4*: *A B C CongA D E P*
by (*metis G3 LeA-def* ‹*Bet A B C* ⟹ *A B C LeA D E F*› *angle-construction-3 not-bet-distincts*)
have *R5*: *P InAngle D E F*
proof −
have *R6*: *P* ≠ *E*
using *R4 conga-diff56* **by** *auto*
have *Bet D E F*
by (*metis* (*no-types*) *P1 R3 Tarski-neutral-dimensionless.bet-conga--bet Tarski-neutral-dimensionless-axioms*)
thus *?thesis*
by (*simp add*: *R6 G3 G3A in-angle-line*)
qed
then have *A B C LeA D E F* **using** *R4 R5 LeA-def* **by** *blast*
}
{
assume *S1*: *B Out A Q*
have *S2*: *B Out A C*
using *G1 P2 S1 l6-7 out-bet-out-1* **by** *blast*
have *S3*: *Col A B C*
by (*simp add*: *Col-perm S2 out-col*)
then have *A B C LeA D E F*
using *R1* **by** *blast*
}
{
assume *S3B*: ¬ *Col A B Q*
obtain *P* **where** *S4*: *A B C CongA D E P* ∧ *D E OS P F*
by (*meson P1 R1 Tarski-neutral-dimensionless.ncol-conga-ncol Tarski-neutral-dimensionless-axioms S3B angle-construction-1*)
obtain *PP* **where** *S4A*: *E Out P PP* ∧ *Cong E PP B X*
by (*metis G1 S4 os-distincts out-diff1 segment-construction-3*)
have *S5*: *P InAngle D E F*
proof −
have *PP InAngle DD E FF*
proof −
have *Z3*: *PP* ≠ *E*
using *S4A l6-3-1* **by** *blast*
have *Z4*: *Bet DD PP FF*
proof −
have *L1*: *C B Q CongA P E F*
proof −
have *K1*: *B A OS C Q*
using *Col-perm P1 R1 S3B in-angle-one-side invert-one-side* **by** *blast*
have *K2*: *E D OS P F*
by (*simp add*: *S4 invert-one-side*)
have *C B A CongA P E D*
by (*simp add*: *S4 conga-comm*)
thus *?thesis*
using *K1 K2 P1 l11-22b* **by** *auto*
qed
have *L2*: *Cong DD FF A Q*
proof −
have *DD E FF CongA A B Q*
proof −
have *L3*: *A B Q CongA D E F*
by (*simp add*: *P1*)
then have *L3A*: *D E F CongA A B Q*
using *conga-sym* **by** *blast*
have *L4*: *E Out DD D*
using *G2 Out-cases* **by** *auto*
have *L5*: *E Out FF F*
using *G3G Out-cases* **by** *blast*
have *L6*: *B Out A A*
using *S3B not-col-distincts out-trivial* **by** *auto*
have *B Out Q Q*
by (*metis S3B not-col-distincts out-trivial*)

```
        thus ?thesis using L3A L4 L5 L6 l11-10
          by blast
      qed
      have L2B: Cong DD E A B
        using Cong-perm G2 by blast
      have Cong E FF B Q
        by (simp add: G3G)
      thus ?thesis
        using L2B ‹DD E FF CongA A B Q› cong2-conga-cong by auto
    qed
    have L8: Cong A X DD PP
    proof –
      have L9: A B X CongA DD E PP
      proof –
        have L9B: B Out A A
          using S3B not-col-distincts out-trivial by blast
        have L9D: E Out D D
          using G3 out-trivial by auto
        have E Out PP P
          using Out-cases S4A by blast
        thus ?thesis using l11-10 S4 L9B G1 L9D
          using G2 Out-cases by blast
      qed
      have L10: Cong A B DD E
        using G2 not-cong-4321 by blast
      have Cong B X E PP
        using Cong-perm S4A by blast
      thus ?thesis
        using L10 L9 cong2-conga-cong by blast
    qed
    have A X Q Cong3 DD PP FF
    proof –
      have L12B: Cong A Q DD FF
        using L2 not-cong-3412 by blast
      have Cong X Q PP FF
      proof –
        have L13A: X B Q CongA PP E FF
        proof –
          have L13AC: B Out Q Q
            by (metis S3B col-trivial-2 out-trivial)
          have L13AD: E Out PP P
            by (simp add: S4A l6-6)
          have E Out FF F
            by (simp add: G3G l6-6)
          thus ?thesis
            using L1 G1 L13AC L13AD l11-10 by blast
        qed
        have L13B: Cong X B PP E
          using S4A not-cong-4321 by blast
        have Cong B Q E FF
          using G3G not-cong-3412 by blast
        thus ?thesis
          using L13A L13B cong2-conga-cong by auto
      qed
      thus ?thesis
        by (simp add: Cong3-def L12B L8)
    qed
    thus ?thesis using P2 l4-6 by blast
  qed
  have PP = E ∨ E Out PP PP
    using out-trivial by auto
  thus ?thesis
    using InAngle-def G3 G3B Z3 Z4 by auto
qed
thus ?thesis
  using G2 G3G S4A l11-25 by blast
```

```
        qed
        then have  A B C LeA D E F
          using S4 LeA-def by blast
      }
      then have  A B C LeA D E F
        using R2 ‹B Out A Q ⟹ A B C LeA D E F› ‹Bet A B Q ⟹ A B C LeA D E F› by blast
    }
    then have A B C LeA D E F
      using G4 ‹B Out A C ⟹ A B C LeA D E F› ‹Bet A B C ⟹ A B C LeA D E F› by blast
  }
  thus ?thesis
    using P2 ‹X = B ⟹ A B C LeA D E F› by blast
qed

lemma bet-in-angle-bet:
  assumes Bet A B P and
    P InAngle A B C
  shows Bet A B C
  by (metis (no-types) Col-def Col-perm assms(1) assms(2) col-in-angle-out not-bet-and-out)

lemma lea-line:
  assumes Bet A B P and
    A B P LeA A B C
  shows Bet A B C
  by (metis Tarski-neutral-dimensionless.bet-conga--bet Tarski-neutral-dimensionless.l11-29-a Tarski-neutral-dimensionless-axioms
assms(1) assms(2) bet-in-angle-bet)

lemma eq-conga-out:
  assumes A B A CongA D E F
  shows E Out D F
  by (metis CongA-def assms l11-21-a out-trivial)

lemma out-conga-out:
  assumes B Out A C and
    A B C CongA D E F
  shows E Out D F
  using assms(1) assms(2) l11-21-a by blast

lemma conga-ex-cong3:
  assumes A B C CongA A′ B′ C′
  shows ∃ AA CC. ((B Out A AA ∧ B Out C CC) ⟶ AA B CC Cong3 A′ B′ C′)
  using out-diff2 by blast

lemma conga-preserves-in-angle:
  assumes A B C CongA A′ B′ C′ and
    A B I CongA A′ B′ I′ and
    I InAngle A B C and A′ B′ OS I′ C′
  shows I′ InAngle A′ B′ C′
proof −
  have P1: A ≠ B
    using assms(1) conga-diff1 by auto
  have P2: B ≠ C
    using assms(1) conga-diff2 by blast
  have P3: A′ ≠ B′
    using assms(1) conga-diff45 by auto
  have P4: B′ ≠ C′
    using assms(1) conga-diff56 by blast
  have P5: I ≠ B
    using assms(2) conga-diff2 by auto
  have P6: I′ ≠ B′
    using assms(2) conga-diff56 by blast
  have P7: Bet A B C ∨ B Out A C ∨ ¬ Col A B C
    using l6-4-2 by blast
  {
    assume Bet A B C
    have Q1: Bet A′ B′ C′
```

using ‹*Bet A B C*› *assms*(*1*) *assms*(*4*) *bet-col col124--nos col-conga-col* **by** *blast*
then have *I′ InAngle A′ B′ C′*
using *assms*(*4*) *bet-col col124--nos* **by** *auto*
}
{
assume *B Out A C*
then have *I′ InAngle A′ B′ C′*
by (*metis P4 assms*(*2*) *assms*(*3*) *in-angle-out l11-21-a out321--inangle*)
}
{
assume *Z1*: ¬ *Col A B C*
have *Q2*: *Bet A B I* ∨ *B Out A I* ∨ ¬ *Col A B I*
by (*simp add*: *or-bet-out*)
{
assume *Bet A B I*
then have *I′ InAngle A′ B′ C′*
using ‹*Bet A B C* ⟹ *I′ InAngle A′ B′ C′*› *assms*(*3*) *bet-in-angle-bet* **by** *blast*
}
{
assume *B Out A I*
then have *I′ InAngle A′ B′ C′*
using *P4 assms*(*2*) *l11-21-a out321--inangle* **by** *auto*
}
{
assume ¬ *Col A B I*
obtain *AA′* **where** *Q3*: *B′ Out A′ AA′* ∧ *Cong B′ AA′ B A*
using *P1 P3 segment-construction-3* **by** *presburger*
obtain *CC′* **where** *Q4*: *B′ Out C′ CC′* ∧ *Cong B′ CC′ B C*
using *P2 P4 segment-construction-3* **by** *presburger*
obtain *J* **where** *Q5*: *Bet A J C* ∧ (*J* = *B* ∨ *B Out J I*)
using *InAngle-def assms*(*3*) **by** *auto*
have *Q6*: *B* ≠ *J*
using *Q5 Z1 bet-col* **by** *auto*
have *Q7*: ¬ *Col A B J*
using *Q5 Q6* ‹¬ *Col A B I*› *col-permutation-2 col-transitivity-1 out-col* **by** *blast*
have ¬ *Col A′ B′ I′*
by (*metis assms*(*4*) *col123--nos*)
then have ∃ *C′*. (*A B J CongA A′ B′ C′* ∧ *A′ B′ OS C′ I′*)
using *Q7 angle-construction-1* **by** *blast*
then obtain *J′* **where** *Q8*: *A B J CongA A′ B′ J′* ∧ *A′ B′ OS J′ I′* **by** *blast*
have *Q9*: *B′* ≠ *J′*
using *Q8 conga-diff56* **by** *blast*
obtain *JJ′* **where** *Q10*: *B′ Out J′ JJ′* ∧ *Cong B′ JJ′ B J*
using *segment-construction-3 Q6 Q9* **by** *blast*
have *Q11*: ¬ *Col A′ B′ J′*
using *Q8 col123--nos* **by** *blast*
have *Q12*: *A′* ≠ *JJ′*
by (*metis Col-perm Q10 Q11 out-col*)
have *Q13*: *B′* ≠ *JJ′*
using *Q10 out-distinct* **by** *blast*
have *Q14*: ¬ *Col A′ B′ JJ′*
using *Col-perm Q10 Q11 Q13 l6-16-1 out-col* **by** *blast*
have *Q15*: *A B C CongA AA′ B′ CC′*
proof −
have *T2*: *C* ≠ *B* **using** *P2* **by** *auto*
have *T3*: *AA′* ≠ *B′*
using *Out-def Q3* **by** *blast*
have *T4*: *CC′* ≠ *B′*
using *Q4 out-distinct* **by** *blast*
have *T5*: ∀ *A′ C′ D′ F′*. (*B Out A′ A* ∧ *B Out C′ C* ∧ *B′ Out D′ AA′* ∧
B′ Out F′ CC′ ∧*Cong B A′ B′ D′* ∧ *Cong B C′ B′ F′* ⟶ *Cong A′ C′ D′ F′*)
by (*smt Q3 Q4 Tarski-neutral-dimensionless.l11-4-1 Tarski-neutral-dimensionless-axioms assms*(*1*) *l6-6 l6-7*)
thus *?thesis* **using** *P1 T2 T3 T4 l11-4-2* **by** *blast*
qed
have *Q16*: *A′ B′ J′ CongA A′ B′ JJ′*
proof −

have *P9*: B' *Out* A' A'
by (*simp add*: *P3 out-trivial*)
have B' *Out* JJ' J'
using *Out-cases Q10* **by** *auto*
thus *?thesis*
using *l11-10*
by (*simp add*: *P9 out2--conga*)
qed
have *Q17*: B' *Out* I' JJ' ∨ A' B' *TS* I' JJ'
proof –
have *Coplanar* A' I' B' J'
by (*metis* (*full-types*) *Q8 ncoplanar-perm-3 os--coplanar*)
then have *Coplanar* A' I' B' JJ'
using *Q10 Q9 col-cop--cop out-col* **by** *blast*
then have *R1*: *Coplanar* A' B' I' JJ' **using** *coplanar-perm-2*
by *blast*
have A' B' I' *CongA* A' B' JJ'
proof –
have *R2*: A' B' I' *CongA A B I*
by (*simp add*: *assms*(*2*) *conga-sym*)
have *A B I CongA* A' B' JJ'
proof –
have *f1*: ∀ *p pa pb*. ¬ *p Out pa pb* ∧ ¬ *p Out pb pa* ∨ *p Out pa pb*
using *Out-cases* **by** *blast*
then have *f2*: B' *Out* JJ' J'
using *Q10* **by** *blast*
have *B Out J I*
by (*metis Q5 Q6*)
thus *?thesis*
using *f2 f1* **by** (*meson P3 Q8 Tarski-neutral-dimensionless.l11-10 Tarski-neutral-dimensionless-axioms* ‹¬ *Col A B I*› *col-one-side-out col-trivial-2 one-side-reflexivity out-trivial*)
qed
thus *?thesis*
using *R2 conga-trans* **by** *blast*
qed
thus *?thesis* **using** *R1 conga-cop--or-out-ts* **by** *blast*
qed
{
assume *Z2*: B' *Out* I' JJ'
have *Z3*: *J B C CongA* J' B' C'
proof –
have *R1*: *B A OS J C*
by (*metis Q5 Q7 Z1 bet-out invert-one-side not-col-distincts out-one-side*)
have *R2*: B' A' *OS* J' C'
by (*meson Q10 Z2 assms*(*4*) *invert-one-side l6-6 one-side-symmetry out-out-one-side*)
have *J B A CongA* J' B' A'
using *Q8 conga-comm* **by** *blast*
thus *?thesis* **using** *assms*(*1*) *R1 R2 l11-22b* **by** *blast*
qed
then have I' *InAngle* A' B' C'
proof –
have *A J C Cong3* AA' JJ' CC'
proof –
have *R8*: *Cong A J* AA' JJ'
proof –
have *R8A*: *A B J CongA* AA' B' JJ'
proof –
have *R8AB*: *B Out A A*
by (*simp add*: *P1 out-trivial*)
have *R8AC*: *B Out J I*
using *Q5 Q6* **by** *auto*
have *R8AD*: B' *Out* AA' A'
using *Out-cases Q3* **by** *auto*
have B' *Out* JJ' I'
using *Out-cases Z2* **by** *blast*
thus *?thesis*

```
            using assms(2) R8AB R8AC R8AD l11-10 by blast
          qed
          have R8B: Cong A B AA′ B′
            using Q3 not-cong-4321 by blast
          have R8C: Cong B J B′ JJ′
            using Q10 not-cong-3412 by blast
          thus ?thesis
            using R8A R8B cong2-conga-cong by blast
        qed
        have LR8A: Cong A C AA′ CC′
          using Q15 Q3 Q4 cong2-conga-cong cong-4321 cong-symmetry by blast
        have Cong J C JJ′ CC′
        proof −
          have K1:B′ Out JJ′ J′
            using Out-cases Q10 by auto
          have B′ Out CC′ C′
            using Out-cases Q4 by auto
          then have J′ B′ C′ CongA JJ′ B′ CC′ using K1
            by (simp add: out2--conga)
          then have LR9A: J B C CongA JJ′ B′ CC′
            using Z3 conga-trans by blast                      have LR9B: Cong J B JJ′ B′
            using Q10 not-cong-4321 by blast
          have Cong B C B′ CC′
            using Q4 not-cong-3412 by blast
          thus ?thesis
            using LR9A LR9B cong2-conga-cong by blast
        qed
        thus ?thesis using R8 LR8A
          by (simp add: Cong3-def)
      qed
      then have R10: Bet AA′ JJ′ CC′ using Q5 l4-6 by blast
      have JJ′ InAngle AA′ B′ CC′
      proof −
        have R11: AA′ ≠ B′
          using Out-def Q3 by auto
        have R12: CC′ ≠ B′
          using Out-def Q4 by blast
        have Bet AA′ JJ′ CC′ ∧ (JJ′ = B′ ∨ B′ Out JJ′ JJ′)
          using R10 out-trivial by auto
        thus ?thesis
          using InAngle-def Q13 R11 R12 by auto
      qed
      thus ?thesis
        using Z2 Q3 Q4 l11-25 by blast
    qed
  }
  {
    assume X1: A′ B′ TS I′ JJ′
    have A′ B′ OS I′ J′
      by (simp add: Q8 one-side-symmetry)
    then have X2: B′ A′ OS I′ JJ′
      using Q10 invert-one-side out-out-one-side by blast
    then have I′ InAngle A′ B′ C′
      using X1 invert-one-side l9-9 by blast
  }
  then have I′ InAngle A′ B′ C′
    using Q17 ‹B′ Out I′ JJ′ ⟹ I′ InAngle A′ B′ C′› by blast
 }
 then have I′ InAngle A′ B′ C′
   using Q2 ‹B Out A I ⟹ I′ InAngle A′ B′ C′› ‹Bet A B I ⟹ I′ InAngle A′ B′ C′› by blast
}
thus ?thesis
  using P7 ‹B Out A C ⟹ I′ InAngle A′ B′ C′› ‹Bet A B C ⟹ I′ InAngle A′ B′ C′› by blast
qed

lemma l11-30:
```

assumes *A B C LeA D E F* **and**
A B C CongA A′ B′ C′ **and**
D E F CongA D′ E′ F′
shows *A′ B′ C′ LeA D′ E′ F′*
proof −
obtain *Q* **where** *P1*: *C InAngle A B Q* ∧ *A B Q CongA D E F*
using *assms*(*1*) *l11-29-a* **by** *blast*
have *P1A*: *C InAngle A B Q* **using** *P1* **by** *simp*
have *P1B*: *A B Q CongA D E F* **using** *P1* **by** *simp*
have *P2*: *A* ≠ *B*
using *P1A inangle-distincts* **by** *auto*
have *P3*: *C* ≠ *B*
using *P1A inangle-distincts* **by** *blast*
have *P4*: *A′* ≠ *B′*
using *CongA-def assms*(*2*) **by** *blast*
have *P5*: *C′* ≠ *B′*
using *CongA-def assms*(*2*) **by** *auto*
have *P6*: *D* ≠ *E*
using *CongA-def P1B* **by** *blast*
have *P7*: *F* ≠ *E*
using *CongA-def P1B* **by** *blast*
have *P8*: *D′* ≠ *E′*
using *CongA-def assms*(*3*) **by** *blast*
have *P9*: *F′* ≠ *E′*
using *CongA-def assms*(*3*) **by** *blast*
have *P10*: *Bet A′ B′ C′* ∨ *B′ Out A′ C′* ∨ ¬ *Col A′ B′ C′*
using *or-bet-out* **by** *blast*
{
assume *Bet A′ B′ C′*
then have ∃ *Q′*. (*C′ InAngle A′ B′ Q′* ∧ *A′ B′ Q′ CongA D′ E′ F′*)
by (*metis P1 P4 P5 P8 P9 assms*(*2*) *assms*(*3*) *bet-conga--bet bet-in-angle-bet conga-line conga-sym inangle3123*)
}
{
assume *R1*: *B′ Out A′ C′*
obtain *Q′* **where** *R2*: *D′ E′ F′ CongA A′ B′ Q′*
using *P4 P8 P9 angle-construction-3* **by** *blast*
then have *C′ InAngle A′ B′ Q′*
using *col-in-angle P1 R1 conga-diff56 out321--inangle* **by** *auto*
then have ∃ *Q′*. (*C′ InAngle A′ B′ Q′* ∧ *A′ B′ Q′ CongA D′ E′ F′*)
using *R2 conga-sym* **by** *blast*
}
{
assume *R3*: ¬ *Col A′ B′ C′*
have *R3A*: *Bet D′ E′ F′* ∨ *E′ Out D′ F′* ∨ ¬ *Col D′ E′ F′*
using *or-bet-out* **by** *blast*
{
assume *Bet D′ E′ F′*
have ∃ *Q′*. (*C′ InAngle A′ B′ Q′* ∧ *A′ B′ Q′ CongA D′ E′ F′*)
by (*metis P4 P5 P8 P9* ‹*Bet D′ E′ F′*› *conga-line in-angle-line point-construction-different*)
}
{
assume *R4A*: *E′ Out D′ F′*
obtain *Q′* **where** *R4*: *D′ E′ F′ CongA A′ B′ Q′*
using *P4 P8 P9 angle-construction-3* **by** *blast*
then have *R5*: *B′ Out A′ Q′* **using** *out-conga-out R4A* **by** *blast*
have *R6*: *A B Q CongA D′ E′ F′*
using *P1 assms*(*3*) *conga-trans* **by** *blast*
then have *R7*: *B Out A Q* **using** *out-conga-out R4A R4*
using *conga-sym* **by** *blast*
have *R8*: *B Out A C*
using *P1A R7 in-angle-out* **by** *blast*
then have *R9*: *B′ Out A′ C′* **using** *out-conga-out assms*(*2*)
by *blast*
have ∃ *Q′*. (*C′ InAngle A′ B′ Q′* ∧ *A′ B′ Q′ CongA D′ E′ F′*)
by (*simp add*: *R9* ‹*B′ Out A′ C′* ⟹ ∃ *Q′*. *C′ InAngle A′ B′ Q′* ∧ *A′ B′ Q′ CongA D′ E′ F′*›)
}

{
assume ¬ *Col* D' E' F'
obtain *QQ* **where** *S1*: D' E' F' *CongA* A' B' *QQ* ∧ A' B' *OS QQ* C'
using *R3* ‹¬ *Col* D' E' F'› *angle-construction-1* **by** *blast*
have *S1A*: *A B Q CongA* A' B' *QQ* **using** *S1*
using *P1 assms*(*3*) *conga-trans* **by** *blast*
have A' B' *OS* C' *QQ* **using** *S1*
by (*simp add*: *S1 one-side-symmetry*)
then have *S2*: C' *InAngle* A' B' *QQ* **using** *conga-preserves-in-angle S1A*
using *P1A assms*(*2*) **by** *blast*
have *S3*: A' B' *QQ CongA* D' E' F'
by (*simp add*: *S1 conga-sym*)
then have ∃ Q'. (C' *InAngle* A' B' Q' ∧ A' B' Q' *CongA* D' E' F')
using *S2* **by** *auto*
}
then have ∃ Q'. (C' *InAngle* A' B' Q' ∧ A' B' Q' *CongA* D' E' F')
using *R3A* ‹E' *Out* D' F' ⟹ ∃ Q'. C' *InAngle* A' B' Q' ∧ A' B' Q' *CongA* D' E' F'› ‹*Bet* D' E' F' ⟹ ∃ Q'. C' *InAngle* A' B' Q' ∧ A' B' Q' *CongA* D' E' F'› **by** *blast*
}
thus *?thesis* **using** *l11-29-b*
using *P10* ‹B' *Out* A' C' ⟹ ∃ Q'. C' *InAngle* A' B' Q' ∧ A' B' Q' *CongA* D' E' F'› ‹*Bet* A' B' C' ⟹ ∃ Q'. C' *InAngle* A' B' Q' ∧ A' B' Q' *CongA* D' E' F'› **by** *blast*
qed

lemma *l11-31-1*:
assumes *B Out A C* **and**
$D \neq E$ **and**
$F \neq E$
shows *A B C LeA D E F*
by (*metis* (*full-types*) *LeA-def assms*(*1*) *assms*(*2*) *assms*(*3*) *l11-21-b out321--inangle segment-construction-3*)

lemma *l11-31-2*:
assumes $A \neq B$ **and**
$C \neq B$ **and**
$D \neq E$ **and**
$F \neq E$ **and**
Bet D E F
shows *A B C LeA D E F*
by (*metis LeA-def angle-construction-3 assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *conga-diff56 in-angle-line*)

lemma *lea-refl*:
assumes $A \neq B$ **and**
$C \neq B$
shows *A B C LeA A B C*
by (*meson assms*(*1*) *assms*(*2*) *conga-refl l11-29-b out341--inangle out-trivial*)

lemma *conga--lea*:
assumes *A B C CongA D E F*
shows *A B C LeA D E F*
by (*metis Tarski-neutral-dimensionless.conga-diff1 Tarski-neutral-dimensionless.conga-diff2 Tarski-neutral-dimensionless.l11-30 Tarski-neutral-dimensionless-axioms assms conga-refl lea-refl*)

lemma *conga--lea456123*:
assumes *A B C CongA D E F*
shows *D E F LeA A B C*
by (*simp add*: *Tarski-neutral-dimensionless.conga--lea Tarski-neutral-dimensionless-axioms assms conga-sym*)

lemma *lea-left-comm*:
assumes *A B C LeA D E F*
shows *C B A LeA D E F*
by (*metis assms conga-pseudo-refl conga-refl l11-30 lea-distincts*)

lemma *lea-right-comm*:
assumes *A B C LeA D E F*
shows *A B C LeA F E D*
by (*meson assms conga-right-comm l11-29-a l11-29-b*)

lemma *lea-comm*:
assumes*A B C LeA D E F*
shows *C B A LeA F E D*
using *assms lea-left-comm lea-right-comm* **by** *blast*

lemma *lta-left-comm*:
assumes *A B C LtA D E F*
shows *C B A LtA D E F*
by (*meson LtA-def Tarski-neutral-dimensionless.conga-left-comm Tarski-neutral-dimensionless.lea-left-comm Tarski-neutral-dimensio*
assms)

lemma *lta-right-comm*:
assumes *A B C LtA D E F*
shows *A B C LtA F E D*
by (*meson Tarski-neutral-dimensionless.LtA-def Tarski-neutral-dimensionless.conga-comm Tarski-neutral-dimensionless.lea-comm Tarski-neutral-dimensionless.lta-left-comm Tarski-neutral-dimensionless-axioms assms*)

lemma *lta-comm*:
assumes *A B C LtA D E F*
shows *C B A LtA F E D*
using *assms lta-left-comm lta-right-comm* **by** *blast*

lemma *lea-out4--lea*:
assumes *A B C LeA D E F* **and**
B Out A A′ **and**
B Out C C′ **and**
E Out D D′ **and**
E Out F F′
shows *A′ B C′ LeA D′ E F′*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *l11-30 l6-6 out2--conga* **by** *auto*

lemma *lea121345*:
assumes $A \neq B$ **and**
$C \neq D$ **and**
$D \neq E$
shows *A B A LeA C D E*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *l11-31-1 out-trivial* **by** *auto*

lemma *inangle--lea*:
assumes *P InAngle A B C*
shows *A B P LeA A B C*
by (*metis Tarski-neutral-dimensionless.l11-29-b Tarski-neutral-dimensionless-axioms assms conga-refl inangle-distincts*)

lemma *inangle--lea-1*:
assumes *P InAngle A B C*
shows *P B C LeA A B C*
by (*simp add*: *Tarski-neutral-dimensionless.inangle--lea Tarski-neutral-dimensionless.lea-comm Tarski-neutral-dimensionless-axioms assms l11-24*)

lemma *inangle--lta*:
assumes ¬ *Col P B C* **and**
P InAngle A B C
shows *A B P LtA A B C*
by (*metis LtA-def TS-def Tarski-neutral-dimensionless.conga-cop--or-out-ts Tarski-neutral-dimensionless.conga-os--out Tarski-neutral-dimensionless.inangle--lea Tarski-neutral-dimensionless.ncol-conga-ncol Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*2*) *col-one-side-out col-trivial-3 in-angle-one-side inangle--coplanar invert-two-sides l11-21-b ncoplanar-perm-12 not-col-permutation-3 one-side-reflexivity*)

lemma *in-angle-trans*:
assumes *C InAngle A B D* **and**
D InAngle A B E
shows *C InAngle A B E*
proof −
obtain *CC* **where** *P1*: $Bet\ A\ CC\ D \wedge (CC = B \vee B\ Out\ CC\ C)$

```
    using InAngle-def assms(1) by auto
  obtain DD where P2: Bet A DD E ∧ (DD = B ∨ B Out DD D)
    using InAngle-def assms(2) by auto
  then have P3: Bet A DD E by simp
  have P4: DD = B ∨ B Out DD D using P2 by simp
  {
    assume CC = B ∧ DD = B
    then have C InAngle A B E
      using InAngle-def P2 assms(1) assms(2) by auto
  }
  {
    assume CC = B ∧ B Out DD D
    then have C InAngle A B E
      by (metis InAngle-def P1 assms(1) assms(2) bet-in-angle-bet)
  }
  {
    assume B Out CC C ∧ DD = B
    then have C InAngle A B E
      by (metis Out-def P2 assms(2) in-angle-line inangle-distincts)
  }
  {
    assume P3: B Out CC C ∧ B Out DD D
    then have P3A: B Out CC C by simp
    have P3B: B Out DD D using P3 by simp
    have C InAngle A B DD
      using P3 assms(1) inangle-distincts l11-25 out-trivial by blast
    then obtain CC' where T1: Bet A CC' DD ∧ (CC' = B ∨ B Out CC' C)
      using InAngle-def by auto
    {
      assume CC' = B
      then have C InAngle A B E
        by (metis P2 P3 T1 assms(2) between-exchange4 in-angle-line inangle-distincts out-diff2)
    }
    {
      assume B Out CC' C
      then have C InAngle A B E
        by (metis InAngle-def P2 T1 assms(1) assms(2) between-exchange4)
    }

    then have C InAngle A B E
      using T1 ‹CC' = B ⟹ C InAngle A B E› by blast
  }
  thus ?thesis
     using P1 P2 ‹B Out CC C ∧ DD = B ⟹ C InAngle A B E› ‹CC = B ∧ B Out DD D ⟹ C InAngle A B E›
‹CC = B ∧ DD = B ⟹ C InAngle A B E› by blast
qed

lemma lea-trans:
  assumes A B C LeA A1 B1 C1 and
    A1 B1 C1 LeA A2 B2 C2
  shows A B C LeA A2 B2 C2
proof −
  obtain P1 where T1: P1 InAngle A1 B1 C1 ∧ A B C CongA A1 B1 P1
    using LeA-def assms(1) by auto
  obtain P2 where T2: P2 InAngle A2 B2 C2 ∧ A1 B1 C1 CongA A2 B2 P2
    using LeA-def assms(2) by blast
  have T3: A ≠ B
    using CongA-def T1 by auto
  have T4: C ≠ B
    using CongA-def T1 by blast
  have T5: A1 ≠ B1
    using T1 inangle-distincts by blast
  have T6: C1 ≠ B1
    using T1 inangle-distincts by blast
  have T7: A2 ≠ B2
    using T2 inangle-distincts by blast
```

```
have T8: C2 ≠ B2
  using T2 inangle-distincts by blast
have T9: Bet A B C  ∨ B Out A C ∨ ¬ Col A B C
  using not-out-bet by auto
{
  assume Bet A B C
  then have A B C LeA A2 B2 C2
    by (metis T1 T2 T3 T4 T7 T8 bet-conga--bet bet-in-angle-bet l11-31-2)
}
{
  assume B Out A C
  then have A B C LeA A2 B2 C2
    by (simp add: T7 T8 l11-31-1)
}
{
  assume H1: ¬ Col A B C
  have T10: Bet A2 B2 C2 ∨ B2 Out A2 C2 ∨ ¬ Col A2 B2 C2
    using not-out-bet by auto
  {
    assume Bet A2 B2 C2
    then have A B C LeA A2 B2 C2
      by (simp add: T3 T4 T7 T8 l11-31-2)
  }
  {
    assume T10A: B2 Out A2 C2
    have B Out A C
    proof –
      have B1 Out A1 P1
      proof –
        have B1 Out A1 C1 using T2 conga-sym T2 T10A in-angle-out out-conga-out by blast
        thus ?thesis using T1 in-angle-out by blast
      qed
      thus ?thesis using T1 conga-sym l11-21-a by blast
    qed
    then have A B C LeA A2 B2 C2
      using ‹B Out A C ⟹ A B C LeA A2 B2 C2› by blast
  }
  {
    assume T12: ¬ Col A2 B2 C2
    obtain P where T13: A B C CongA A2 B2 P ∧ A2 B2 OS P C2
      using T12 H1 angle-construction-1 by blast
    have T14: A2 B2 OS P2 C2
    proof –
      have ¬ Col B2 A2 P2
      proof –
        have B2 ≠ A2
          using T7 by auto
          {
            assume H2: P2 = A2
            have A2 B2 A2 CongA A1 B1 C1
              using T2 H2 conga-sym by blast
            then have B1 Out A1 C1
              using eq-conga-out by blast
            then have B1 Out A1 P1
              using T1 in-angle-out by blast
            then have B Out A C
              using T1 conga-sym out-conga-out by blast
            then have False
              using Col-cases H1 out-col by blast
          }
          then have P2 ≠ A2 by blast
          have Bet A2 B2 P2 ∨ B2 Out A2 P2 ∨ ¬ Col A2 B2 P2
            using not-out-bet by auto
          {
            assume H4: Bet A2 B2 P2
            then have Bet A2 B2 C2
```

using *T2 bet-in-angle-bet* **by** *blast*
then have *Col B2 A2 P2* ⟶ *False*
using *Col-def T12* **by** *blast*
then have ¬ *Col B2 A2 P2*
using *H4 bet-col not-col-permutation-4* **by** *blast*
}
{
assume *H5*: *B2 Out A2 P2*
then have *B1 Out A1 C1*
using *T2 conga-sym out-conga-out* **by** *blast*
then have *B1 Out A1 P1*
using *T1 in-angle-out* **by** *blast*
then have *B Out A C*
using *H1 T1 ncol-conga-ncol not-col-permutation-4 out-col* **by** *blast*
then have ¬ *Col B2 A2 P2*
using *Col-perm H1 out-col* **by** *blast*
}
{
assume ¬ *Col A2 B2 P2*
then have ¬ *Col B2 A2 P2*
using *Col-perm* **by** *blast*
}
thus *?thesis*
using ‹*B2 Out A2 P2* ⟹ ¬ *Col B2 A2 P2*› ‹*Bet A2 B2 P2* ⟹ ¬ *Col B2 A2 P2*› ‹*Bet A2 B2 P2* ∨ *B2 Out A2 P2* ∨ ¬ *Col A2 B2 P2*› **by** *blast*
qed
thus *?thesis*
by (*simp add*: *T2 T12 in-angle-one-side*)
qed
have *S1*: *A2 B2 OS P P2*
using *T13 T14 one-side-symmetry one-side-transitivity* **by** *blast*
have *A1 B1 P1 CongA A2 B2 P*
using *conga-trans conga-sym T1 T13* **by** *blast*
then have *P InAngle A2 B2 P2*
using *conga-preserves-in-angle T2 T1 S1* **by** *blast*
then have *P InAngle A2 B2 C2*
using *T2 in-angle-trans* **by** *blast*
then have *A B C LeA A2 B2 C2*
using *T13 LeA-def* **by** *blast*
}
then have *A B C LeA A2 B2 C2*
using *T10* ‹*B2 Out A2 C2* ⟹ *A B C LeA A2 B2 C2*› ‹*Bet A2 B2 C2* ⟹ *A B C LeA A2 B2 C2*› **by** *blast*
}
thus *?thesis*
using *T9* ‹*B Out A C* ⟹ *A B C LeA A2 B2 C2*› ‹*Bet A B C* ⟹ *A B C LeA A2 B2 C2*› **by** *blast*
qed

lemma *in-angle-asym*:
assumes *D InAngle A B C* **and**
C InAngle A B D
shows *A B C CongA A B D*
proof −
obtain *CC* **where** *P1*: *Bet A CC D* ∧ (*CC* = *B* ∨ *B Out CC C*)
using *InAngle-def assms*(*2*) **by** *auto*
obtain *DD* **where** *P2*: *Bet A DD C* ∧ (*DD* = *B* ∨ *B Out DD D*)
using *InAngle-def assms*(*1*) **by** *auto*
{
assume (*CC* = *B*) ∧ (*DD* = *B*)
then have *A B C CongA A B D*
by (*metis P1 P2 assms*(*2*) *conga-line inangle-distincts*)
}
{
assume (*CC* = *B*) ∧ (*B Out DD D*)
then have *A B C CongA A B D*
by (*metis P1 assms*(*1*) *bet-in-angle-bet conga-line inangle-distincts*)
}

{
assume (*B Out CC C*) ∧ (*DD* = *B*)
then have *A B C CongA A B D*
by (*metis P2 assms*(*2*) *bet-in-angle-bet conga-line inangle-distincts*)
}
{
assume *V1*: (*B Out CC C*) ∧ (*B Out DD D*)
obtain *X* **where** *P3*: *Bet CC X C* ∧ *Bet DD X D*
using *P1 P2 between-symmetry inner-pasch* **by** *blast*
then have *B Out X D*
using *V1 out-bet-out-2* **by** *blast*
then have *B Out C D*
using *P3 V1 out2-bet-out* **by** *blast*
then have *A B C CongA A B D*
using *assms*(*2*) *inangle-distincts l6-6 out2--conga out-trivial* **by** *blast*
}
thus *?thesis* **using** *P1 P2*
using ‹*B Out CC C* ∧ *DD* = *B* ⟹ *A B C CongA A B D*› ‹*CC* = *B* ∧ *B Out DD D* ⟹ *A B C CongA A B D*›
‹*CC* = *B* ∧ *DD* = *B* ⟹ *A B C CongA A B D*› **by** *blast*
qed

lemma *lea-asym*:
assumes *A B C LeA D E F* **and**
D E F LeA A B C
shows *A B C CongA D E F*
proof *cases*
assume *P1*: *Col A B C*
{
assume *P1A*: *Bet A B C*
have *P2*: *D* ≠ *E*
using *assms*(*1*) *lea-distincts* **by** *blast*
have *P3*: *F* ≠ *E*
using *assms*(*2*) *lea-distincts* **by** *auto*
have *P4*: *A* ≠ *B*
using *assms*(*1*) *lea-distincts* **by** *auto*
have *P5*: *C* ≠ *B*
using *assms*(*2*) *lea-distincts* **by** *blast*
obtain *P* **where** *P6*: *P InAngle D E F* ∧ *A B C CongA D E P*
using *LeA-def assms*(*1*) **by** *blast*
then have *A B C CongA D E P* **by** *simp*
then have *Bet D E P* **using** *P1 P1A bet-conga--bet*
by *blast*
then have *Bet D E F*
using *P6 bet-in-angle-bet* **by** *blast*
then have *A B C CongA D E F*
by (*metis Tarski-neutral-dimensionless.bet-conga--bet Tarski-neutral-dimensionless.conga-line Tarski-neutral-dimensionless.l11-29-c*
Tarski-neutral-dimensionless-axioms P2 P3 P4 P5 assms(*2*) *bet-in-angle-bet*)
}
{
assume *T1*: ¬ *Bet A B C*
then have *T2*: *B Out A C*
using *P1 not-out-bet* **by** *auto*
obtain *P* **where** *T3*: *P InAngle A B C* ∧ *D E F CongA A B P*
using *LeA-def assms*(*2*) **by** *blast*
then have *T3A*: *P InAngle A B C* **by** *simp*
have *T3B*: *D E F CongA A B P* **using** *T3* **by** *simp*
have *T4*: *E Out D F*
proof −
have *T4A*: *B Out A P*
using *T2 T3 in-angle-out* **by** *blast*
have *A B P CongA D E F*
by (*simp add*: *T3 conga-sym*)
thus *?thesis*
using *T4A l11-21-a* **by** *blast*
qed
then have *A B C CongA D E F*

```
      by (simp add: T2 l11-21-b)
    }
    thus ?thesis
      using ‹Bet A B C ⟹ A B C CongA D E F› by blast
  next
    assume T5: ¬ Col A B C
    obtain Q where T6: C InAngle A B Q ∧ A B Q CongA D E F
      using assms(1) l11-29-a by blast
    then have T6A: C InAngle A B Q by simp
    have T6B: A B Q CongA D E F by (simp add: T6)
    obtain P where T7: P InAngle A B C ∧ D E F CongA A B P
      using LeA-def assms(2) by blast
    then have T7A: P InAngle A B C by simp
    have T7B: D E F CongA A B P by (simp add: T7)
    have T13: A B Q CongA  A B P
      using T6 T7 conga-trans by blast
    have T14: Bet A B Q ∨ B Out A Q ∨ ¬ Col A B Q
      using not-out-bet by auto
    {
      assume R1: Bet A B Q
      then have A B C CongA D E F
        using T13 T5 T7 bet-col bet-conga--bet bet-in-angle-bet by blast
    }
    {
      assume R2: B Out A Q
      then have A B C CongA D E F
        using T6 in-angle-out l11-21-a l11-21-b by blast
    }
    {
      assume R3: ¬ Col A B Q
      have R3A: Bet A B P ∨ B Out A P ∨ ¬ Col A B P
        using not-out-bet by blast
      {
        assume R3AA: Bet A B P
        then have A B C CongA D E F
          using T5 T7 bet-col bet-in-angle-bet by blast
      }
      {
        assume R3AB: B Out A P
        then have A B C CongA D E F
          by (meson Col-cases R3 T13 ncol-conga-ncol out-col)
      }
      {
        assume R3AC: ¬ Col A B P
        have R3AD: B Out P Q ∨ A B TS P Q
        proof –
          have Coplanar A B P Q
            using T6A T7A coplanar-perm-8 in-angle-trans inangle--coplanar by blast
          thus ?thesis
            by (simp add: T13 conga-sym conga-cop--or-out-ts)
        qed
        {
          assume B Out P Q
          then have C InAngle A B P
            by (meson R3 T6A bet-col between-symmetry l11-24 l11-25-aux)
          then have A B C CongA A B P
            by (simp add: T7A in-angle-asym)
          then have A B C CongA D E F
         by (meson T7B Tarski-neutral-dimensionless.conga-sym Tarski-neutral-dimensionless.conga-trans Tarski-neutral-dimensionless-a
        }
        {
          assume W1: A B TS P Q
          have A B OS P Q
            using Col-perm R3 R3AC T6A T7A in-angle-one-side in-angle-trans by blast
          then have A B C CongA D E F
            using W1 l9-9 by blast
```

}
then have *A B C CongA D E F*
using *R3AD* ‹*B Out P Q* ⟹ *A B C CongA D E F*› by *blast*
}
then have *A B C CongA D E F*
using *R3A* ‹*B Out A P* ⟹ *A B C CongA D E F*› ‹*Bet A B P* ⟹ *A B C CongA D E F*› by *blast*
}
thus *?thesis*
using *T14* ‹*B Out A Q* ⟹ *A B C CongA D E F*› ‹*Bet A B Q* ⟹ *A B C CongA D E F*› by *blast*
qed

lemma *col-lta--bet*:
assumes *Col X Y Z* **and**
A B C LtA X Y Z
shows *Bet X Y Z*
proof −
have *A B C LeA X Y Z* ∧ ¬ *A B C CongA X Y Z*
using *LtA-def assms*(*2*) **by** *auto*
then have *Y Out X Z* ⟶ *False*
using *Tarski-neutral-dimensionless.lea-asym Tarski-neutral-dimensionless.lea-distincts Tarski-neutral-dimensionless-axioms l11-31-1*
by *fastforce*
thus *?thesis* **using** *not-out-bet assms*(*1*)
by *blast*
qed

lemma *col-lta--out*:
assumes *Col A B C* **and**
A B C LtA X Y Z
shows *B Out A C*
proof −
have *A B C LeA X Y Z* ∧ ¬ *A B C CongA X Y Z*
using *LtA-def assms*(*2*) **by** *auto*
thus *?thesis*
by (*metis assms*(*1*) *l11-31-2 lea-asym lea-distincts or-bet-out*)
qed

lemma *lta-distincts*:
assumes *A B C LtA D E F*
shows *A*≠*B* ∧ *C*≠*B* ∧ *D*≠*E* ∧ *F*≠*E* ∧ *D* ≠ *F*
by (*metis LtA-def assms bet-neq12--neq col-lta--bet lea-distincts not-col-distincts*)

lemma *gta-distincts*:
assumes *A B C GtA D E F*
shows *A*≠*B* ∧ *C*≠*B* ∧ *D*≠*E* ∧ *F*≠*E* ∧ *A* ≠ *C*
using *GtA-def assms lta-distincts* **by** *presburger*

lemma *acute-distincts*:
assumes *Acute A B C*
shows *A*≠*B* ∧ *C*≠*B*
using *Acute-def assms lta-distincts* **by** *blast*

lemma *obtuse-distincts*:
assumes *Obtuse A B C*
shows *A*≠*B* ∧ *C*≠*B* ∧ *A* ≠ *C*
using *Obtuse-def assms lta-distincts* **by** *blast*

lemma *two-sides-in-angle*:
assumes *B* ≠ *P'* **and**
B P TS A C **and**
Bet P B P'
shows *P InAngle A B C* ∨ *P' InAngle A B C*
proof −
obtain *T* **where** *P1*: *Col T B P* ∧ *Bet A T C*
using *TS-def assms*(*2*) **by** *auto*
have *P2*: *A* ≠ *B*

using *assms*(*2*) *ts-distincts* **by** *blast*
have *P3*: $C \neq B$
using *assms*(*2*) *ts-distincts* **by** *blast*
show *?thesis*
proof *cases*
assume $B = T$
thus *?thesis*
using *P1 P2 P3 assms*(*1*) *in-angle-line* **by** *auto*
next
assume $B \neq T$
thus *?thesis*
by (*metis InAngle-def P1 assms*(*1*) *assms*(*2*) *assms*(*3*) *between-symmetry l6-3-2 or-bet-out ts-distincts*)
qed
qed

lemma *in-angle-reverse*:
assumes $A' \neq B$ **and**
Bet A B A' **and**
C InAngle A B D
shows *D InAngle A' B C*
proof −
have *P1*: $A \neq B$
using *assms*(*3*) *inangle-distincts* **by** *auto*
have *P2*: $D \neq B$
using *assms*(*3*) *inangle-distincts* **by** *blast*
have *P3*: $C \neq B$
using *assms*(*3*) *inangle-distincts* **by** *auto*
show *?thesis*
proof *cases*
assume *Col B A C*
thus *?thesis*
by (*smt P1 P2 P3 assms*(*1*) *assms*(*2*) *assms*(*3*) *bet-in-angle-bet between-inner-transitivity between-symmetry in-angle-line l6-3-2 out321--inangle outer-transitivity-between third-point*)
next
assume *P4*: ¬ *Col B A C*
thus *?thesis*
proof *cases*
assume *Col B D C*
thus *?thesis*
by (*smt P2 P4 assms*(*1*) *assms*(*2*) *assms*(*3*) *bet-col1 col2--eq col-permutation-2 in-angle-one-side l9-19-R1 out341--inangle*)
next
assume *P5*: ¬ *Col B D C*
have *P6*: *C B TS A D*
using *P4 P5 assms*(*3*) *in-angle-two-sides* **by** *auto*
obtain *X* **where** *P7*: *Bet A X D* ∧ ($X = B$ ∨ *B Out X C*)
using *InAngle-def assms*(*3*) **by** *auto*
have *P8*: $X = B \Longrightarrow$ *D InAngle A' B C*
using *Out-def P1 P2 P3 P7 assms*(*1*) *assms*(*2*) *l5-2 out321--inangle* **by** *auto*
{
assume *P9*: *B Out X C*
have *P10*: $C \neq B$
by (*simp add*: *P3*)
have *P10A*: ¬ *Col B A C*
by (*simp add*: *P4*)
have *P10B*: ¬ *Col B D C*
by (*simp add*: *P5*)
have *P10C*: *C InAngle D B A*
by (*simp add*: *assms*(*3*) *l11-24*)
{
assume *Col D B A*
have *Col B A C*
proof −
have $B \neq X$
using *P9 out-distinct* **by** *blast*
have *Col B X A*

```
            by (meson Bet-perm P10C P5 P7 ‹Col D B A› bet-col1 col-permutation-3 in-angle-out or-bet-out out-col)
          have Col B X C
            by (simp add: P9 out-col)
          thus ?thesis
            using ‹B ≠ X› ‹Col B X A› col-transitivity-1 by blast
        qed
        then have False
          by (simp add: P4)
      }
      then have P10E: ¬ Col D B A by auto
      have P11: D B OS C A
        by (simp add: P10C P10E P5 in-angle-one-side)
      have P12: ¬ Col A D B
        using Col-cases P10E by auto
      have P13: ¬ Col A′ D B
        by (metis Col-def ‹Col D B A ⟹ False› assms(1) assms(2) col-transitivity-1)
      have P14: D B TS A A′
        using P12 P13 TS-def assms(2) col-trivial-3 by blast
      have P15: D B TS C A′
        using P11 P14 l9-8-2 one-side-symmetry by blast
      have P16: ¬ Col C D B
        by (simp add: P5 not-col-permutation-3)
      obtain Y where P17: Col Y D B ∧ Bet C Y A′
        using P15 TS-def by auto
      have P18: Bet A′ Y C
        using Bet-perm P17 by blast
      {
        assume S1: Y ≠ B
        have S2: Col D B Y
          using P17 not-col-permutation-2 by blast
        then have S3: Bet D B Y ∨ Bet B Y D ∨ Bet Y D B
          using Col-def S2 by auto
        {
          assume S4: Bet D B Y
          have S5: C B OS A′ Y
                by (metis P17 P18 P5 S1 bet-out-1 col-transitivity-2 l6-6 not-col-permutation-3 not-col-permutation-5
out-one-side)
          have S6: C B TS Y D
            by (metis Bet-perm P16 P17 S1 S4 bet--ts col3 col-trivial-3 invert-two-sides not-col-permutation-1)
          have C B TS A A′
            by (metis (full-types) P4 assms(1) assms(2) bet--ts invert-two-sides not-col-permutation-5)
          then have C B TS Y A
            using S5 l9-2 l9-8-2 by blast
          then have S9: C B OS A D
            using P6 S6 l9-8-1 l9-9 by blast
          then have B Out Y D
            using P6 S9 l9-9 by auto
        }
        {
          assume Bet B Y D
          then have B Out Y D
            by (simp add: S1 bet-out)
        }
        {
          assume Bet Y D B
          then have B Out Y D
            by (simp add: P2 bet-out-1 l6-6)
        }
        have B Out Y D
          using S3 ‹Bet B Y D ⟹ B Out Y D› ‹Bet D B Y ⟹ B Out Y D› ‹Bet Y D B ⟹ B Out Y D› by blast
      }
      then have P19: (Y = B ∨ B Out Y D) by auto
      have D InAngle A′ B C
        using InAngle-def P18 P19 P2 P3 assms(1) by auto
    }
    thus ?thesis using P7 P8 by blast
```

qed
qed
qed

lemma *in-angle-trans2*:
assumes *C InAngle A B D* **and**
D InAngle A B E
shows *D InAngle C B E*
proof −
obtain *pp* :: $'p \Rightarrow 'p \Rightarrow 'p$ **where**
f1: $\forall$ *p pa. Bet p pa* (*pp p pa*) $\wedge$ *pa* $\neq$ (*pp p pa*)
using *point-construction-different* **by** *moura*
then have *f2*: $\forall$ *p. C InAngle D B* (*pp p B*) $\vee$ $\neg$ *D InAngle p B A*
by (*metis assms*(*1*) *in-angle-reverse in-angle-trans l11-24*)
have *f3*: *D InAngle E B A*
using *assms*(*2*) *l11-24* **by** *blast*
then have $E \neq B$
by (*simp add*: *inangle-distincts*)
thus *?thesis*
using *f3 f2 f1* **by** (*meson Bet-perm in-angle-reverse l11-24*)
qed

lemma *l11-36-aux1*:
assumes $A \neq B$ **and**
$A' \neq B$ **and**
$D \neq E$ **and**
$D' \neq E$ **and**
Bet A B A' **and**
Bet D E D' **and**
A B C LeA D E F
shows *D' E F LeA A' B C*
proof −
obtain *P* **where** *P1*: *C InAngle A B P* $\wedge$
A B P CongA D E F
using *assms*(*7*) *l11-29-a* **by** *blast*
thus *?thesis*
by (*metis LeA-def Tarski-neutral-dimensionless.l11-13 Tarski-neutral-dimensionless-axioms assms*(*2*) *assms*(*4*) *assms*(*5*) *assms*(*6*) *conga-sym in-angle-reverse*)
qed

lemma *l11-36-aux2*:
assumes $A \neq B$ **and**
$A' \neq B$ **and**
$D \neq E$ **and**
$D' \neq E$ **and**
Bet A B A' **and**
Bet D E D' **and**
D' E F LeA A' B C
shows *A B C LeA D E F*
by (*metis Bet-cases assms*(*1*) *assms*(*3*) *assms*(*5*) *assms*(*6*) *assms*(*7*) *l11-36-aux1 lea-distincts*)

lemma *l11-36*:
assumes $A \neq B$ **and**
$A' \neq B$ **and**
$D \neq E$ **and**
$D' \neq E$ **and**
Bet A B A' **and**
Bet D E D'
shows *A B C LeA D E F* $\longleftrightarrow$ *D' E F LeA A' B C*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *assms*(*6*) *l11-36-aux1 l11-36-aux2* **by** *auto*

lemma *l11-41-aux*:
assumes $\neg$ *Col A B C* **and**
Bet B A D **and**
$A \neq D$
shows *A C B LtA C A D*

proof –
  obtain *M* **where** *P1*: *M Midpoint A C*
    using *midpoint-existence* **by** *auto*
  obtain *P* **where** *P2*: *M Midpoint B P*
    using *symmetric-point-construction* **by** *auto*
  have *P3*: *A C B Cong3 C A P*
    by (*smt Cong3-def P1 P2 assms*(*1*) *l7-13-R1 l7-2 midpoint-distinct-1 not-col-distincts*)
  have *P4*: $A \neq C$
    using *assms*(*1*) *col-trivial-3* **by** *blast*
  have *P5*: $B \neq C$
    using *assms*(*1*) *col-trivial-2* **by** *blast*
  have *P7*: $A \neq M$
    using *P1 P4 is-midpoint-id* **by** *blast*
  have *P8*: *A C B CongA C A P*
    by (*simp add*: *P3 P4 P5 cong3-conga*)
  have *P8A*: *Bet D A B*
    using *Bet-perm assms*(*2*) **by** *blast*
  have *P8B*: *Bet P M B*
    by (*simp add*: *P2 between-symmetry midpoint-bet*)
  then obtain *X* **where** *P9*: *Bet A X P* ∧ *Bet M X D* **using** *P8A inner-pasch* **by** *blast*
  have *P9A*: *Bet A X P* **by** (*simp add*: *P9*)
  have *P9B*: *Bet M X D* **by** (*simp add*: *P9*)
  have *P10A*: *P InAngle C A D*
  proof –
    have *K1*: *P InAngle M A D*
      by (*metis InAngle-def P3 P5 P7 P9 assms*(*3*) *bet-out cong3-diff2*)
    have *K2*: *A Out C M*
      using *Out-def P1 P4 P7 midpoint-bet* **by** *auto*
    have *K3*: *A Out D D*
      using *assms*(*3*) *out-trivial* **by** *auto*
    have *A Out P P*
      using *K1 inangle-distincts out-trivial* **by** *auto*
    thus *?thesis*
      using *K1 K2 K3 l11-25* **by** *blast*
  qed
  then have *P10*: *A C B LeA C A D*
    using *LeA-def P8* **by** *auto*
  {
    assume *K5*: *A C B CongA C A D*
    then have *K6*: *C A D CongA C A P*
      using *P8 conga-sym conga-trans* **by** *blast*
    have *K7*: *Coplanar C A D P*
      using *P10A inangle--coplanar ncoplanar-perm-18* **by** *blast*
    then have *K8*: *A Out D P* ∨ *C A TS D P*
      by (*simp add*: *K6 conga-cop--or-out-ts*)
    {
      assume *A Out D P*

      then have *Col M B A*
        by (*meson P8A P8B bet-col1 bet-out--bet between-symmetry not-col-permutation-4*)
      then have *K8F*: *Col A M B*
        using *not-col-permutation-1* **by** *blast*
      have *Col A M C*
        by (*simp add*: *P1 bet-col midpoint-bet*)
      then have *False*
        using *K8F P7 assms*(*1*) *col-transitivity-1* **by** *blast*
    }
    then have *K9*: ¬ *A Out D P* **by** *auto*
    {
      assume *V1*: *C A TS D P*
      then have *V3*: *A C TS B P*
        by (*metis P10A P8A assms*(*1*) *col-trivial-1 col-trivial-2 in-angle-reverse in-angle-two-sides invert-two-sides l11-24 l9-18 not-col-permutation-5*)
      have *A C TS B D*
        by (*simp add*: *assms*(*1*) *assms*(*2*) *assms*(*3*) *bet--ts not-col-permutation-5*)
      then have *A C OS D P*

```
      using V1 V3 invert-two-sides l9-8-1 l9-9 by blast
    then have False
      using V1 invert-one-side l9-9 by blast
  }
  then have ¬ C A TS D P by auto
  then have False using K8 K9 by auto
 }
 then have ¬ A C B CongA C A D by auto
 thus ?thesis
  by (simp add: LtA-def P10)
qed

lemma l11-41:
 assumes ¬ Col A B C and
  Bet B A D and
  A ≠ D
 shows A C B LtA C A D ∧ A B C LtA C A D
proof –
 have P1: A C B LtA C A D
  using assms(1) assms(2) assms(3) l11-41-aux by auto
 have A B C LtA C A D
 proof –
  obtain E where T1: Bet C A E ∧ Cong A E C A
   using segment-construction by blast
  have T1A: Bet C A E using T1 by simp
  have T1B: Cong A E C A using T1 by simp
  have T2: A B C LtA B A E
   using T1 assms(1) cong-reverse-identity l11-41-aux not-col-distincts not-col-permutation-5 by blast
  have T3: B A C CongA C A B
   by (metis assms(1) conga-pseudo-refl not-col-distincts)
  have T3A: D A C CongA E A B
   by (metis CongA-def T1 T3 assms(2) assms(3) cong-reverse-identity l11-13)
  then have T4: B A E CongA C A D
   using conga-comm conga-sym by blast
  have A B C CongA A B C
  using T2 Tarski-neutral-dimensionless.conga-refl Tarski-neutral-dimensionless.lta-distincts Tarski-neutral-dimensionless-axioms
by fastforce
  then have T5: A B C LeA C A D
  by (meson T2 T4 Tarski-neutral-dimensionless.LtA-def Tarski-neutral-dimensionless.l11-30 Tarski-neutral-dimensionless-axioms)
  have ¬ A B C CongA C A D
  by (meson T2 Tarski-neutral-dimensionless.LtA-def Tarski-neutral-dimensionless.conga-right-comm Tarski-neutral-dimensionless.co
Tarski-neutral-dimensionless-axioms T3A)
  thus ?thesis
   by (simp add: LtA-def T5)
 qed
 thus ?thesis by (simp add: P1)
qed

lemma not-conga:
 assumes A B C CongA A′ B′ C′ and
  ¬ A B C CongA D E F
 shows ¬ A′ B′ C′ CongA D E F
 by (meson assms(1) assms(2) conga-trans)

lemma not-conga-sym:
 assumes ¬ A B C CongA D E F
 shows ¬ D E F CongA A B C
 using assms conga-sym by blast

lemma not-and-lta:
 shows ¬ (A B C LtA D E F ∧ D E F LtA A B C)
proof –
 {
  assume P1: A B C LtA D E F ∧ D E F LtA A B C
  then have A B C CongA D E F
   using LtA-def lea-asym by blast
```

then have *False*
using *LtA-def P1* **by** *blast*
}
thus *?thesis* **by** *auto*
qed

lemma *conga-preserves-lta*:
assumes *A B C CongA A′ B′ C′* **and**
D E F CongA D′ E′ F′ **and**
A B C LtA D E F
shows *A′ B′ C′ LtA D′ E′ F′*
by (*meson Tarski-neutral-dimensionless.LtA-def Tarski-neutral-dimensionless.conga-trans Tarski-neutral-dimensionless.l11-30 Tarski-neutral-dimensionless.not-conga-sym Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*2*) *assms*(*3*))

lemma *lta-trans*:
assumes *A B C LtA A1 B1 C1* **and**
A1 B1 C1 LtA A2 B2 C2
shows *A B C LtA A2 B2 C2*
proof −
have *P1*: *A B C LeA A2 B2 C2*
by (*meson LtA-def assms*(*1*) *assms*(*2*) *lea-trans*)
{
assume *A B C CongA A2 B2 C2*
then have *False*
by (*meson Tarski-neutral-dimensionless.LtA-def Tarski-neutral-dimensionless.lea-asym Tarski-neutral-dimensionless.lea-trans Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*2*) *conga--lea456123*)
}
thus *?thesis*
using *LtA-def P1* **by** *blast*
qed

lemma *obtuse-sym*:
assumes *Obtuse A B C*
shows *Obtuse C B A*
by (*meson Obtuse-def Tarski-neutral-dimensionless.lta-right-comm Tarski-neutral-dimensionless-axioms assms*)

lemma *acute-sym*:
assumes *Acute A B C*
shows *Acute C B A*
by (*meson Acute-def Tarski-neutral-dimensionless.lta-left-comm Tarski-neutral-dimensionless-axioms assms*)

lemma *acute-col--out*:
assumes *Col A B C* **and**
Acute A B C
shows *B Out A C*
by (*meson Tarski-neutral-dimensionless.Acute-def Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*2*) *col-lta--out*)

lemma *col-obtuse--bet*:
assumes *Col A B C* **and**
Obtuse A B C
shows *Bet A B C*
using *Obtuse-def assms*(*1*) *assms*(*2*) *col-lta--bet* **by** *blast*

lemma *out--acute*:
assumes *B Out A C*
shows *Acute A B C*
proof −
have *P1*: $A \neq B$
using *assms out-diff1* **by** *auto*
then obtain *D* **where** *P3*: *B D Perp A B*
using *perp-exists* **by** *blast*
then have *P4*: $B \neq D$
using *perp-distinct* **by** *auto*
have *P5*: *Per A B D*
by (*simp add*: *P3 l8-2 perp-per-1*)
have *P6*: *A B C LeA A B D*

using *P1 P4 assms l11-31-1* **by** *auto*
{
assume *A B C CongA A B D*
then have *False*
by (*metis Col-cases P1 P4 P5 assms col-conga-col l8-9 out-col*)
}
then have *A B C LtA A B D*
using *LtA-def P6* **by** *auto*
thus *?thesis*
using *P5 Acute-def* **by** *auto*
qed

lemma *bet--obtuse*:
assumes *Bet A B C* **and**
$A \neq B$ **and** $B \neq C$
shows *Obtuse A B C*
proof −
obtain *D* **where** *P1*: *B D Perp A B*
using *assms*(*2*) *perp-exists* **by** *blast*
have *P5*: $B \neq D$
using *P1 perp-not-eq-1* **by** *auto*
have *P6*: *Per A B D*
using *P1 Perp-cases perp-per-1* **by** *blast*
have *P7*: *A B D LeA A B C*
using *assms*(*2*) *assms*(*3*) *P5 assms*(*1*) *l11-31-2* **by** *auto*
{
assume *A B D CongA A B C*
then have *False*
using *assms*(*2*) *P5 P6 assms*(*1*) *bet-col ncol-conga-ncol per-not-col* **by** *blast*
}
then have *A B D LtA A B C*
using *LtA-def P7* **by** *blast*
thus *?thesis*
using *Obtuse-def P6* **by** *blast*
qed

lemma *l11-43-aux*:
assumes $A \neq B$ **and**
$A \neq C$ **and**
Per B A C ∨ *Obtuse B A C*
shows *Acute A B C*
proof *cases*
assume *P1*: *Col A B C*
{
assume *Per B A C*
then have *Acute A B C*
using *Col-cases P1 assms*(*1*) *assms*(*2*) *per-col-eq* **by** *blast*
}
{
assume *Obtuse B A C*
then have *Bet B A C*
using *P1 col-obtuse--bet col-permutation-4* **by** *blast*
then have *Acute A B C*
by (*simp add*: *assms*(*1*) *bet-out out--acute*)
}
thus *?thesis*
using ‹*Per B A C* ⟹ *Acute A B C*› *assms*(*3*) **by** *blast*
next
assume *P2*: ¬ *Col A B C*
then have *P3*: $B \neq C$
using *col-trivial-2* **by** *auto*
obtain *B'* **where** *P4*: *Bet B A B'* ∧ *Cong A B' B A*
using *segment-construction* **by** *blast*
have *P5*: ¬ *Col B' A C*
by (*metis Col-def P2 P4 col-transitivity-2 cong-reverse-identity*)
then have *P6*: $B' \neq A \wedge B' \neq C$

```
    using not-col-distincts by blast
  then have P7: A C B LtA C A B' ∧ A B C LtA C A B'
    using P2 P4 l11-41 by auto
  then have P7A: A C B LtA C A B' by simp
  have P7B: A B C LtA C A B' by (simp add: P7)
  {
    assume Per B A C
    have Acute A B C
      by (metis Acute-def P4 P7B ‹Per B A C› assms(1) bet-col col-per2--per col-trivial-3 l8-3 lta-right-comm)
  }
  {
    assume T1: Obtuse B A C
    then obtain a b c where T2: Per a b c ∧ a b c LtA B A C
      using Obtuse-def by blast
    then have T2A: Per a b c by simp
    have T2B: a b c LtA B A C by (simp add: T2)
    then have T3: a b c LeA B A C ∧ ¬ a b c CongA B A C
      by (simp add: LtA-def)
    then have T3A: a b c LeA B A C by simp
    have T3B: ¬ a b c CongA B A C by (simp add: T3)
    obtain P where T4: P InAngle B A C ∧ a b c CongA B A P
      using LeA-def T3 by blast
    then have T5: Per B A P using T4 T2 l11-17 by blast
    then have T6: Per P A B
      using l8-2 by blast
    have Col A B B'
      by (simp add: P4 bet-col col-permutation-4)
    then have Per P A B'
      using T6 assms(1) per-col by blast
    then have S3: B A P CongA B' A P
      using l8-2 P6 T5 T4 CongA-def assms(1) l11-16 by auto
    have C A B' LtA P A B
    proof −
      have S4: B A P LeA B A C ⟷ B' A C LeA B' A P
        using P4 P6 assms(1) l11-36 by auto
      have S5: C A B' LeA P A B
      proof −
        have S6: B A P LeA B A C
          using T4 inangle--lea by auto
        have B' A P CongA P A B
          using S3 conga-left-comm not-conga-sym by blast
        thus ?thesis
          using P6 S4 S6 assms(2) conga-pseudo-refl l11-30 by auto
      qed
      {
        assume T10: C A B' CongA P A B
        have Per B' A C
        proof −
          have B A P CongA B' A C
            using T10 conga-comm conga-sym by blast
          thus ?thesis
            using T5 l11-17 by blast
        qed
        then have Per C A B
          using Col-cases P6 ‹Col A B B'› l8-2 l8-3 by blast
        have a b c CongA B A C
        proof −
          have a ≠ b
            using T3A lea-distincts by auto
          have c ≠ b
            using T2B lta-distincts by blast
          have Per B A C
            using Per-cases ‹Per C A B› by blast
          thus ?thesis
            using T2 ‹a ≠ b› ‹c ≠ b› assms(1) assms(2) l11-16 by auto
        qed
```

```
        then have False
          using T3B by blast
      }
      then have ¬ C A B' CongA P A B by blast
      thus ?thesis
        by (simp add: LtA-def S5)
    qed
    then have A B C LtA B A P
      by (meson P7 lta-right-comm lta-trans)
    then have Acute A B C  using T5
      using Acute-def by blast
  }
  thus ?thesis
    using ‹Per B A C ⟹ Acute A B C› assms(3) by blast
qed

lemma l11-43:
  assumes A ≠ B and
    A ≠ C and
    Per B A C ∨ Obtuse B A C
  shows Acute A B C ∧ Acute A C B
  using Per-perm assms(1) assms(2) assms(3) l11-43-aux obtuse-sym by blast

lemma acute-lea-acute:
  assumes Acute D E F and
    A B C LeA D E F
  shows Acute A B C
proof -
  obtain A' B' C' where P1: Per A' B' C' ∧ D E F LtA A' B' C'
    using Acute-def assms(1) by auto
  have P2: A B C LeA A' B' C'
    using LtA-def P1 assms(2) lea-trans by blast
  have ¬ A B C CongA A' B' C'
    by (meson LtA-def P1 assms(2) conga--lea456123 lea-asym lea-trans)
  then have A B C LtA A' B' C'
    by (simp add: LtA-def P2)
  thus ?thesis
    using Acute-def P1 by auto
qed

lemma lea-obtuse-obtuse:
  assumes Obtuse D E F and
    D E F LeA A B C
  shows Obtuse A B C
proof -
  obtain A' B' C' where P1: Per A' B' C' ∧ A' B' C' LtA D E F
    using Obtuse-def assms(1) by auto
  then have P2: A' B' C' LeA A B C
    using LtA-def assms(2) lea-trans by blast
  have ¬ A' B' C' CongA A B C
    by (meson LtA-def P1 assms(2) conga--lea456123 lea-asym lea-trans)
  then have A' B' C' LtA A B C
    by (simp add: LtA-def P2)
  thus ?thesis
    using Obtuse-def P1 by auto
qed

lemma l11-44-1-a:
  assumes A ≠ B and
    A ≠ C and
    Cong B A B C
  shows B A C CongA B C A
  by (metis (no-types, opaque-lifting) Cong3-def assms(1) assms(2) assms(3) cong3-conga cong-inner-transitivity cong-pseudo-reflexivity

lemma l11-44-2-a:
  assumes ¬ Col A B C and
```

```
    B A Lt B C
  shows B C A LtA B A C
proof –
  have T1: A ≠ B
    using assms(1) col-trivial-1 by auto
  have T3: A ≠ C
    using assms(1) col-trivial-3 by auto
  have B A Le B C
    by (simp add: assms(2) lt--le)
  then obtain C′ where P1: Bet B C′ C ∧ Cong B A B C′
    using assms(2) Le-def by blast
  have T5: C ≠ C′
    using P1 assms(2) cong--nlt by blast
  have T5A: C′ ≠ A
    using Col-def Col-perm P1 assms(1) by blast
  then have T6: C′ InAngle B A C
    using InAngle-def P1 T1 T3 out-trivial by auto
  have T7: C′ A C LtA A C′ B ∧ C′ C A LtA A C′ B
  proof –
    have W1: ¬ Col C′ C A
      by (metis Col-def P1 T5 assms(1) col-transitivity-2)
    have W2: Bet C C′ B
      using Bet-perm P1 by blast
    have C′ ≠ B
      using P1 T1 cong-identity by blast
    thus ?thesis
      using l11-41 W1 W2 by simp
  qed
  have T90: B A C′ LtA B A C
  proof –
    have T90A: B A C′ LeA B A C
      by (simp add: T6 inangle--lea)
    have B A C′ CongA B A C′
      using T1 T5A conga-refl by auto
    {
      assume B A C′ CongA B A C
      then have R1: A Out C′ C
        by (metis P1 T7 assms(1) bet-out conga-os--out lta-distincts not-col-permutation-4 out-one-side)
      have B A OS C′ C
        by (metis Col-perm P1 T1 assms(1) bet-out cong-diff-2 out-one-side)
      then have False
        using Col-perm P1 T5 R1 bet-col col2--eq one-side-not-col123 out-col by blast
    }
    then have ¬ B A C′ CongA B A C by blast
    thus ?thesis
      by (simp add: LtA-def T90A)
  qed
  have B A C′ CongA B C′ A
    using P1 T1 T5A l11-44-1-a by auto
  then have K2: A C′ B CongA B A C′
    using conga-left-comm not-conga-sym by blast
  have B C A LtA B A C′
  proof –
    have K1: B C A CongA B C A
      using assms(1) conga-refl not-col-distincts by blast
    have B C A LtA A C′ B
    proof –
      have C′ C A CongA B C A
      proof –
        have K2: C Out B C′
          using P1 T5 bet-out-1 l6-6 by auto
        have C Out A A
          by (simp add: T3 out-trivial)
        thus ?thesis
          by (simp add: K2 out2--conga)
      qed
```

have *A C′ B CongA A C′ B*
using *CongA-def K2 conga-refl* **by** *auto*
thus *?thesis*
using *T7* ‹*C′ C A CongA B C A*› *conga-preserves-lta* **by** *auto*
qed
thus *?thesis*
using *K1 K2 conga-preserves-lta* **by** *auto*
qed
thus *?thesis*
using *T90 lta-trans* **by** *blast*
qed

lemma *not-lta-and-conga*:
¬ (*A B C LtA D E F* ∧ *A B C CongA D E F*)
by (*simp add*: *LtA-def*)

lemma *conga-sym-equiv*:
A B C CongA A′ B′ C′ ⟷ *A′ B′ C′ CongA A B C*
using *not-conga-sym* **by** *blast*

lemma *conga-dec*:
A B C CongA D E F ∨ ¬ *A B C CongA D E F*
by *auto*

lemma *lta-not-conga*:
assumes *A B C LtA D E F*
shows ¬ *A B C CongA D E F*
using *assms not-lta-and-conga* **by** *auto*

lemma *lta--lea*:
assumes *A B C LtA D E F*
shows *A B C LeA D E F*
using *LtA-def assms* **by** *auto*

lemma *nlta*:
¬ *A B C LtA A B C*
using *not-and-lta* **by** *blast*

lemma *lea--nlta*:
assumes *A B C LeA D E F*
shows ¬ *D E F LtA A B C*
by (*meson Tarski-neutral-dimensionless.lea-asym Tarski-neutral-dimensionless.not-lta-and-conga Tarski-neutral-dimensionless-axioms*
assms lta--lea)

lemma *lta--nlea*:
assumes *A B C LtA D E F*
shows ¬ *D E F LeA A B C*
using *assms lea--nlta* **by** *blast*

lemma *l11-44-1-b*:
assumes ¬ *Col A B C* **and**
B A C CongA B C A
shows *Cong B A B C*
proof −
have *B A Lt B C* ∨ *B A Gt B C* ∨ *Cong B A B C*
by (*simp add*: *or-lt-cong-gt*)
thus *?thesis*
by (*meson Gt-def assms*(*1*) *assms*(*2*) *conga-sym l11-44-2-a not-col-permutation-3 not-lta-and-conga*)
qed

lemma *l11-44-2-b*:
assumes *B A C LtA B C A*
shows *B C Lt B A*
proof *cases*
assume *Col A B C*
thus *?thesis*

using *Col-perm assms bet--lt1213 col-lta--bet lta-distincts* **by** *blast*
next
assume *P1*: ¬ *Col A B C*
then have *P2*: *A* ≠ *B*
using *col-trivial-1* **by** *blast*
have *P3*: *A* ≠ *C*
using *P1 col-trivial-3* **by** *auto*
have *B A Lt B C* ∨ *B A Gt B C* ∨ *Cong B A B C*
by (*simp add*: *or-lt-cong-gt*)
{
assume *B A Lt B C*
then have *B C Lt B A*
using *P1 assms l11-44-2-a not-and-lta* **by** *blast*
}
{
assume *B A Gt B C*
then have *B C Lt B A*
using *Gt-def P1 assms l11-44-2-a not-and-lta* **by** *blast*
}
{
assume *Cong B A B C*
then have *B A C CongA B C A*
by (*simp add*: *P2 P3 l11-44-1-a*)
then have *B C Lt B A*
using *assms not-lta-and-conga* **by** *blast*
}
thus *?thesis*
by (*meson P1 Tarski-neutral-dimensionless.not-and-lta Tarski-neutral-dimensionless-axioms* ‹*B A Gt B C* ⟹ *B C Lt B A*› ‹*B A Lt B C* ∨ *B A Gt B C* ∨ *Cong B A B C*› *assms l11-44-2-a*)
qed

lemma *l11-44-1*:
assumes ¬ *Col A B C*
shows *B A C CongA B C A* ⟷ *Cong B A B C*
using *assms l11-44-1-a l11-44-1-b not-col-distincts* **by** *blast*

lemma *l11-44-2*:
assumes ¬ *Col A B C*
shows *B A C LtA B C A* ⟷ *B C Lt B A*
using *assms l11-44-2-a l11-44-2-b not-col-permutation-3* **by** *blast*

lemma *l11-44-2bis*:
assumes ¬ *Col A B C*
shows *B A C LeA B C A* ⟷ *B C Le B A*
proof −
{
assume *P1*: *B A C LeA B C A*
{
assume *B A Lt B C*
then have *B C A LtA B A C*
by (*simp add*: *assms l11-44-2-a*)
then have *False*
using *P1 lta--nlea* **by** *auto*
}
then have ¬ *B A Lt B C* **by** *blast*
have *B C Le B A*
using ‹¬ *B A Lt B C*› *nle--lt* **by** *blast*
}
{
assume *P2*: *B C Le B A*
have *B A C LeA B C A*
proof *cases*
assume *Cong B C B A*
then have *B A C CongA B C A*
by (*metis assms conga-sym l11-44-1-a not-col-distincts*)
thus *?thesis*

```
      by (simp add: conga--lea)
    next
      assume ¬ Cong B C B A
      then have B A C LtA B C A
        by (simp add: l11-44-2 assms Lt-def P2)
      thus ?thesis
        by (simp add: lta--lea)
    qed
  }
  thus ?thesis
    using ‹B A C LeA B C A ⟹ B C Le B A› by blast
qed

lemma l11-46:
  assumes A ≠ B and
    B ≠ C and
    Per A B C ∨ Obtuse A B C
  shows B A Lt A C ∧ B C Lt A C
proof cases
  assume Col A B C
  thus ?thesis
    by (meson assms(1) assms(2) assms(3) bet--lt1213 bet--lt2313 col-obtuse--bet lt-left-comm per-not-col)
next
  assume P1: ¬ Col A B C
  have P2: A ≠ C
    using P1 col-trivial-3 by auto
  have P3: Acute B A C ∧ Acute B C A
    using assms(1) assms(2) assms(3) l11-43 by auto
  then obtain A′ B′ C′ where P4: Per A′ B′ C′ ∧ B C A LtA A′ B′ C′
    using Acute-def P3 by auto
  {
    assume P5: Per A B C
    have P5A: A C B CongA A C B
      by (simp add: P2 assms(2) conga-refl)
    have S1: A ≠ B
      by (simp add: assms(1))
    have S2: B ≠ C
      by (simp add: assms(2))
    have S3: A′ ≠ B′
      using P4 lta-distincts by blast
    have S4: B′ ≠ C′
      using P4 lta-distincts by blast
    then have A′ B′ C′ CongA A B C using l11-16
      using S1 S2 S3 S4 P4 P5 by blast
    then have A C B LtA A B C
      using P5A P4 conga-preserves-lta lta-left-comm by blast
  }
  {
    assume Obtuse A B C
    obtain A′′ B′′ C′′ where P6: Per A′′ B′′ C′′ ∧ A′′ B′′ C′′ LtA A B C
      using Obtuse-def ‹Obtuse A B C› by auto
    have B C A LtA A′ B′ C′
      by (simp add: P4)
    then have P7: A C B LtA A′ B′ C′
      by (simp add: lta-left-comm)
    have A′ B′ C′ LtA A B C
    proof −
      have U1: A′′ B′′ C′′ CongA A′ B′ C′
      proof −
        have V2: A′′ ≠ B′′
          using P6 lta-distincts by blast
        have V3: C′′ ≠ B′′
          using P6 lta-distincts by blast
        have V5: A′ ≠ B′
          using P7 lta-distincts by blast
        have C′ ≠ B′
```

using *P4 lta-distincts* **by** *blast*
thus *?thesis* **using** *P6 V2 V3 P4 V5*
by (*simp add*: *l11-16*)
qed
have *U2*: *A B C CongA A B C*
using *assms*(*1*) *assms*(*2*) *conga-refl* **by** *auto*
have *U3*: *A′′ B′′ C′′ LtA A B C*
by (*simp add*: *P6*)
thus *?thesis*
using *U1 U2 conga-preserves-lta* **by** *auto*
qed
then have *A C B LtA A B C*
using *P7 lta-trans* **by** *blast*
}
then have *A C B LtA A B C*
using ‹*Per A B C* ⟹ *A C B LtA A B C*› *assms*(*3*) **by** *blast*
then have *A B Lt A C*
by (*simp add*: *l11-44-2-b*)
then have *B A Lt A C*
using *Lt-cases* **by** *blast*
have *C A B LtA C B A*
proof −
obtain *A′ B′ C′* **where** *U4*: *Per A′ B′ C′* ∧ *B A C LtA A′ B′ C′*
using *Acute-def P3* **by** *blast*
{
assume *Per A B C*
then have *W3*: *A′ B′ C′ CongA C B A*
using *U4 assms*(*2*) *l11-16 l8-2 lta-distincts* **by** *blast*
have *W2*: *C A B CongA C A B*
using *P2 assms*(*1*) *conga-refl* **by** *auto*
have *C A B LtA A′ B′ C′*
by (*simp add*: *U4 lta-left-comm*)
then have *C A B LtA C B A*
using *W2 W3 conga-preserves-lta* **by** *blast*
}
{
assume *Obtuse A B C*
then obtain *A′′ B′′ C′′* **where** *W4*: *Per A′′ B′′ C′′* ∧ *A′′ B′′ C′′ LtA A B C*
using *Obtuse-def* **by** *auto*
have *W5*: *C A B LtA A′ B′ C′*
by (*simp add*: *U4 lta-left-comm*)
have *A′ B′ C′ LtA C B A*
proof −
have *W6*: *A′′ B′′ C′′ CongA A′ B′ C′* **using** *l11-16 W4 U4*
using *lta-distincts* **by** *blast*
have *C B A CongA C B A*
using *assms*(*1*) *assms*(*2*) *conga-refl* **by** *auto*
thus *?thesis*
using *W4 W6 conga-left-comm conga-preserves-lta* **by** *blast*
qed
then have *C A B LtA C B A*
using *W5 lta-trans* **by** *blast*
}
thus *?thesis*
using ‹*Per A B C* ⟹ *C A B LtA C B A*› *assms*(*3*) **by** *blast*
qed
then have *C B Lt C A*
by (*simp add*: *l11-44-2-b*)
then have *C B Lt A C*
using *Lt-cases* **by** *auto*
then have *B C Lt A C*
using *Lt-cases* **by** *blast*
thus *?thesis*
by (*simp add*: ‹*B A Lt A C*›)
qed

lemma *l11-47*:
  assumes *Per A C B* **and**
    H PerpAt C H A B
  shows *Bet A H B* $\wedge$ *A* $\neq$ *H* $\wedge$ *B* $\neq$ *H*
proof −
  have *P1*: *Per C H A*
    using *assms*(*2*) *perp-in-per-1* **by** *auto*
  have *P2*: *C H Perp A B*
    using *assms*(*2*) *perp-in-perp* **by** *auto*
  thus *?thesis*
  proof *cases*
    assume *Col A C B*
    thus *?thesis*
      by (*metis P1 assms*(*1*) *assms*(*2*) *per-distinct-1 per-not-col perp-in-distinct perp-in-id*)
  next
    assume *P3*: ¬ *Col A C B*
    have *P4*: *A* $\neq$ *H*
      by (*metis P2 Per-perm Tarski-neutral-dimensionless.l8-7 Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*2*) *col-trivial-1 perp-in-per-2 perp-not-col2*)
    have *P5*: *Per C H B*
      using *assms*(*2*) *perp-in-per-2* **by** *auto*
    have *P6*: *B* $\neq$ *H*
      using *P1 P2 assms*(*1*) *l8-2 l8-7 perp-not-eq-1* **by** *blast*
    have *P7*: *H A Lt A C* $\wedge$ *H C Lt A C*
      by (*metis P1 P2 P4 l11-46 l8-2 perp-distinct*)
    have *P8*: *C A Lt A B* $\wedge$ *C B Lt A B*
      using *P3 assms*(*1*) *l11-46 not-col-distincts* **by** *blast*
    have *P9*: *H B Lt B C* $\wedge$ *H C Lt B C*
      by (*metis P2 P5 P6 Per-cases l11-46 perp-not-eq-1*)
    have *P10*: *Bet A H B*
    proof −
      have *T1*: *Col A H B*
        using *assms*(*2*) *col-permutation-5 perp-in-col* **by** *blast*
      have *T2*: *A H Le A B* **using** *P7 P8*
        by (*meson lt-comm lt-transitivity nlt--le not-and-lt*)
      have *H B Le A B*
        by (*meson Lt-cases P8 P9 le-transitivity local.le-cases lt--nle*)
      thus *?thesis*
        using *T1 T2 l5-12-b* **by** *blast*
    qed
    thus *?thesis*
      by (*simp add*: *P4 P6*)
  qed
qed

lemma *l11-49*:
  assumes *A B C CongA A′ B′ C′* **and**
    Cong B A B′ A′ **and**
    Cong B C B′ C′
  shows *Cong A C A′ C′* $\wedge$ (*A* $\neq$ *C* $\longrightarrow$ (*B A C CongA B′ A′ C′* $\wedge$ *B C A CongA B′ C′ A′*))
proof −
  have *T1*: *Cong A C A′ C′*
    using *assms*(*1*) *assms*(*2*) *assms*(*3*) *cong2-conga-cong not-cong-2143* **by** *blast*
  {
    assume *P1*: *A* $\neq$ *C*
    have *P2*: *A* $\neq$ *B*
      using *CongA-def assms*(*1*) **by** *blast*
    have *P3*: *C* $\neq$ *B*
      using *CongA-def assms*(*1*) **by** *blast*
    have *B A C Cong3 B′ A′ C′*
      by (*simp add*: *Cong3-def T1 assms*(*2*) *assms*(*3*))
    then have *T2*: *B A C CongA B′ A′ C′*
      using *P1 P2 cong3-conga* **by** *auto*
    have *B C A Cong3 B′ C′ A′*
      using *Cong3-def T1 assms*(*2*) *assms*(*3*) *cong-3-swap-2* **by** *blast*
    then have *B C A CongA B′ C′ A′*

```
      using P1 P3 cong3-conga by auto
    then have B A C CongA B' A' C' ∧ B C A CongA B' C' A' using T2 by blast
  }
  thus ?thesis
    by (simp add: T1)
qed

lemma l11-50-1:
  assumes ¬ Col A B C and
    B A C CongA B' A' C' and
    A B C CongA A' B' C' and
    Cong A B A' B'
  shows Cong A C A' C' ∧ Cong B C B' C' ∧ A C B CongA A' C' B'
proof −
  obtain C'' where P1: B' Out C'' C' ∧ Cong B' C'' B C
    by (metis Col-perm assms(1) assms(3) col-trivial-3 conga-diff56 l6-11-existence)
  have P2: B' ≠ C''
    using P1 out-diff1 by auto
  have P3: ¬ Col A' B' C'
    using assms(1) assms(3) ncol-conga-ncol by blast
  have P4: ¬ Col A' B' C''
    by (meson P1 P2 P3 col-transitivity-1 not-col-permutation-2 out-col)
  have P5: Cong A C A' C''
  proof −
    have Q1: B Out A A
      using assms(1) not-col-distincts out-trivial by auto
    have Q2: B Out C C
      using assms(1) col-trivial-2 out-trivial by force
    have Q3: B' Out A' A'
      using P3 not-col-distincts out-trivial by auto
    have Q5: Cong B A B' A'
      using assms(4) not-cong-2143 by blast
    have Cong B C B' C''
      using P1 not-cong-3412 by blast
    thus ?thesis
      using l11-4-1 P1 Q1 Q2 Q3 Q5 assms(3) by blast
  qed
  have P6: B A C Cong3 B' A' C''
    using Cong3-def Cong-perm P1 P5 assms(4) by blast
  have P7: B A C CongA B' A' C''
    by (metis P6 assms(1) cong3-conga not-col-distincts)
  have P8: B' A' C' CongA B' A' C''
    by (meson P7 assms(2) conga-sym conga-trans)
  have B' A' OS C' C''
    using Col-perm Out-cases P1 P3 out-one-side by blast
  then have A' Out C' C''
    using P8 conga-os--out by auto
  then have Col A' C' C''
    using out-col by auto
  then have P9: C' = C''
    using Col-perm P1 out-col P3 col-transitivity-1 by blast
  have T1: Cong A C A' C'
    by (simp add: P5 P9)
  have T2: Cong B C B' C'
    using Cong-perm P1 P9 by blast
  then have A C B CongA A' C' B'
    using T1 assms(1) assms(2) assms(4) col-trivial-2 l11-49 by blast
  thus ?thesis using T1 T2 by blast
qed

lemma l11-50-2:
  assumes ¬ Col A B C and
    B C A CongA B' C' A' and
    A B C CongA A' B' C' and
    Cong A B A' B'
  shows Cong A C A' C' ∧ Cong B C B' C' ∧ C A B CongA C' A' B'
```

```
proof -
  have P1: A ≠ B
    using assms(1) col-trivial-1 by auto
  have P2: B ≠ C
    using assms(1) col-trivial-2 by auto
  have P3: A′ ≠ B′
    using P1 assms(4) cong-diff by blast
  have P4: B′ ≠ C′
    using assms(2) conga-diff45 by auto
  then obtain C′′ where P5: B′ Out C′′ C′ ∧ Cong B′ C′′ B C
    using P2 l6-11-existence by presburger
  have P5BIS: B′ ≠ C′′
    using P5 out-diff1 by auto
  have P5A: Col B′ C′′ C′
    using P5 out-col by auto
  have P6: ¬ Col A′ B′ C′
    using assms(1) assms(3) ncol-conga-ncol by blast
  {
    assume Col A′ B′ C′′
    then have Col B′ C′′ A′
      using not-col-permutation-2 by blast
    then have Col B′ C′ A′ using col-transitivity-1 P5BIS P5A by blast
    then have Col A′ B′ C′
      using Col-perm by blast
    then have False
      using P6 by auto
  }
  then have P7: ¬ Col A′ B′ C′′ by blast
  have P8: Cong A C A′ C′′
  proof -
    have B Out A A
      by (simp add: P1 out-trivial)
    have K1: B Out C C
      using P2 out-trivial by auto
    have K2: B′ Out A′ A′
      using P3 out-trivial by auto
    have Cong B A B′ A′
      by (simp add: Cong-perm assms(4))
    have Cong B C B′ C′′
      using Cong-perm P5 by blast
    thus ?thesis
      using P5 ‹Cong B A B′ A′› P1 out-trivial K1 K2 assms(3) l11-4-1 by blast
  qed
  have P9: B C A Cong3 B′ C′′ A′
    using Cong3-def Cong-perm P5 P8 assms(4) by blast
  then have P10: B C A CongA B′ C′′ A′
    using assms(1) cong3-conga not-col-distincts by auto
  have P11: B′ C′ A′ CongA B′ C′′ A′
    using P9 assms(2) cong3-conga2 conga-sym by blast
  show ?thesis
  proof cases
    assume L1: C′ = C′′
    then have L2: Cong A C A′ C′
      by (simp add: P8)
    have L3: Cong B C B′ C′
      using Cong-perm L1 P5 by blast
    have C A B Cong3 C′ A′ B′
      by (simp add: L1 P9 cong-3-swap cong-3-swap-2)
    then have C A B CongA C′ A′ B′
      by (metis CongA-def P1 assms(2) cong3-conga)
    thus ?thesis using L2 L3 by auto
  next
    assume R1: C′ ≠ C′′
    have R1A: ¬ Col C′′ C′ A′
      by (metis P5A P7 R1 col-permutation-2 col-trivial-2 colx)
    have R1B: Bet B′ C′′ C′ ∨ Bet B′ C′ C′′
```

```
      using Out-def P5 by auto
    {
      assume S1: Bet B' C'' C'
      then have S2: C'' A' C' LtA A' C'' B' ∧ C'' C' A' LtA A' C'' B'
        using P5BIS R1A between-symmetry l11-41 by blast
      have B' C' A' CongA C'' C' A'
          by (metis P11 R1 Tarski-neutral-dimensionless.conga-comm Tarski-neutral-dimensionless-axioms S1 bet-out-1
conga-diff45 not-conga-sym out2--conga out-trivial)
      then have B' C' A' LtA A' C'' B'
      by (meson P11 Tarski-neutral-dimensionless.conga-right-comm Tarski-neutral-dimensionless.not-conga Tarski-neutral-dimensionle
Tarski-neutral-dimensionless-axioms S2 not-lta-and-conga)
      then have Cong A C A' C' ∧ Cong B C B' C'
      by (meson P11 Tarski-neutral-dimensionless.conga-right-comm Tarski-neutral-dimensionless-axioms not-lta-and-conga)
    }
    {
      assume Z1: Bet B' C' C''
      have Z2: ¬ Col C' C'' A'
        by (simp add: R1A not-col-permutation-4)
      have Z3: C'' Out C' B'
        by (simp add: R1 Z1 bet-out-1)
      have Z4: C'' Out A' A'
        using P7 not-col-distincts out-trivial by blast
      then have Z4A: B' C'' A' CongA C' C'' A'
        by (simp add: Z3 out2--conga)
      have Z4B: B' C'' A' LtA A' C' B'
      proof -
        have Z5: C' C'' A' CongA B' C'' A'
          using Z4A not-conga-sym by blast
        have Z6: A' C' B' CongA A' C' B'
          using P11 P4 conga-diff2 conga-refl by blast
        have C' C'' A' LtA A' C' B'
          using P4 Z1 Z2 between-symmetry l11-41 by blast
        thus ?thesis
          using Z5 Z6 conga-preserves-lta by auto
      qed
      have B' C'' A' CongA B' C' A'
        using P11 not-conga-sym by blast
      then have Cong A C A' C' ∧ Cong B C B' C'
      by (meson Tarski-neutral-dimensionless.conga-right-comm Tarski-neutral-dimensionless-axioms Z4B not-lta-and-conga)
    }
    then have R2: Cong A C A' C' ∧ Cong B C B' C'
      using R1B ‹Bet B' C'' C' ⟹ Cong A C A' C' ∧ Cong B C B' C'› by blast
    then have C A B CongA C' A' B'
      using P1 assms(2) l11-49 not-cong-2143 by blast
    thus ?thesis using R2 by auto
  qed
qed

lemma l11-51:
  assumes A ≠ B and
    A ≠ C and
    B ≠ C and
    Cong A B A' B' and
    Cong A C A' C' and
    Cong B C B' C'
  shows
    B A C CongA B' A' C' ∧ A B C CongA A' B' C' ∧ B C A CongA B' C' A'
proof -
  have B A C Cong3 B' A' C' ∧ A B C Cong3 A' B' C' ∧ B C A Cong3 B' C' A'
    using Cong3-def Cong-perm assms(4) assms(5) assms(6) by blast
  thus ?thesis
    using assms(1) assms(2) assms(3) cong3-conga by auto
qed

lemma conga-distinct:
  assumes A B C CongA D E F
```

shows $A \neq B \wedge C \neq B \wedge D \neq E \wedge F \neq E$
using *CongA-def assms* **by** *auto*

lemma *l11-52*:
assumes *A B C CongA A′ B′ C′* **and**
Cong A C A′ C′ **and**
Cong B C B′ C′ **and**
B C Le A C
shows *Cong B A B′ A′* ∧ *B A C CongA B′ A′ C′* ∧ *B C A CongA B′ C′ A′*
proof −
have *P1*: $A \neq B$
using *CongA-def assms*(*1*) **by** *blast*
have *P2*: $C \neq B$
using *CongA-def assms*(*1*) **by** *blast*
have *P3*: $A' \neq B'$
using *CongA-def assms*(*1*) **by** *blast*
have *P4*: $C' \neq B'$
using *assms*(*1*) *conga-diff56* **by** *auto*
have *P5*: *Cong B A B′ A′*
proof *cases*
assume *P6*: *Col A B C*
then have *P7*: *Bet A B C* ∨ *Bet B C A* ∨ *Bet C A B*
using *Col-def* **by** *blast*
{
assume *P8*: *Bet A B C*
then have *Bet A′ B′ C′*
using *assms*(*1*) *bet-conga--bet* **by** *blast*
then have *Cong B A B′ A′*
using *P8 assms*(*2*) *assms*(*3*) *l4-3 not-cong-2143* **by** *blast*
}
{
assume *P9*: *Bet B C A*
then have *P10*: *B′ Out A′ C′*
using *Out-cases P2 assms*(*1*) *bet-out l11-21-a* **by** *blast*
then have *P11*: *Bet B′ A′ C′* ∨ *Bet B′ C′ A′*
by (*simp add*: *Out-def*)
{
assume *Bet B′ A′ C′*
then have *Cong B A B′ A′*
using *P3 assms*(*2*) *assms*(*3*) *assms*(*4*) *bet-le-eq l5-6* **by** *blast*
}
{
assume *Bet B′ C′ A′*
then have *Cong B A B′ A′*
using *Cong-perm P9 assms*(*2*) *assms*(*3*) *l2-11-b* **by** *blast*
}
then have *Cong B A B′ A′*
using *P11* ‹*Bet B′ A′ C′* ⟹ *Cong B A B′ A′*› **by** *blast*
}
{
assume *Bet C A B*
then have *Cong B A B′ A′*
using *P1 assms*(*4*) *bet-le-eq between-symmetry* **by** *blast*
}
thus *?thesis*
using *P7* ‹*Bet A B C* ⟹ *Cong B A B′ A′*› ‹*Bet B C A* ⟹ *Cong B A B′ A′*› **by** *blast*
next
assume *Z1*: ¬ *Col A B C*
obtain *A′′* **where** *Z2*: *B′ Out A′′ A′* ∧ *Cong B′ A′′ B A*
using *P1 P3 l6-11-existence* **by** *force*
then have *Z3*: *A′ B′ C′ CongA A′′ B′ C′*
by (*simp add*: *P4 out2--conga out-trivial*)
have *Z4*: *A B C CongA A′′ B′ C′*
using *Z3 assms*(*1*) *not-conga* **by** *blast*
have *Z5*: *Cong A′′ C′ A C*
using *Z2 Z4 assms*(*3*) *cong2-conga-cong cong-4321 cong-symmetry* **by** *blast*

have *Z6*: A'' B' C' *Cong3 A B C*
using *Cong3-def Cong-perm Z2 Z5 assms*(*3*) **by** *blast*
have *Z7*: *Cong* A'' C' A' C'
using *Z5 assms*(*2*) *cong-transitivity* **by** *blast*
have *Z8*: ¬ *Col* A' B' C'
by (*metis Z1 assms*(*1*) *ncol-conga-ncol*)
then have *Z9*: ¬ *Col* A'' B' C'
by (*metis Z2 col-transitivity-1 not-col-permutation-4 out-col out-diff1*)
{
assume *Z9A*: $A'' \neq A'$
have *Z10*: *Bet* B' A'' A' ∨ *Bet* B' A' A''
using *Out-def Z2* **by** *auto*
{
assume *Z11*: *Bet* B' A'' A'
have *Z12*: A'' C' B' *LtA* C' A'' A' ∧ A'' B' C' *LtA* C' A'' A'
by (*simp add*: *Z11 Z9 Z9A l11-41*)
have *Z13*: *Cong* A' C' A'' C'
using *Cong-perm Z7* **by** *blast*
have *Z14*: ¬ *Col* A'' C' A'
by (*metis Col-def Z11 Z9 Z9A col-transitivity-1*)
have *Z15*: C' A'' A' *CongA* C' A' A'' ⟷ *Cong* C' A'' C' A'
by (*simp add*: *Z14 l11-44-1*)
have *Z16*: *Cong* C' A' C' A''
using *Cong-perm Z7* **by** *blast*
then have *Z17*: *Cong* C' A'' C' A'
using *Cong-perm* **by** *blast*
then have *Z18*: C' A'' A' *CongA* C' A' A''
by (*simp add*: *Z15*)
have *Z19*: ¬ *Col* B' C' A''
using *Col-perm Z9* **by** *blast*
have *Z20*: B' A' C' *CongA* A'' A' C'
by (*metis Tarski-neutral-dimensionless.col-conga-col Tarski-neutral-dimensionless-axioms Z11 Z3 Z9 Z9A bet-out-1 col-trivial-3 out2--conga out-trivial*)
have *Z21*: ¬ *Col* B' C' A'
using *Col-perm Z8* **by** *blast*
then have *Z22*: C' B' A' *LtA* C' A' B' ⟷ C' A' *Lt* C' B'
by (*simp add*: *l11-44-2*)
have A'' B' C' *CongA* C' B' A'
using *Z3 conga-right-comm not-conga-sym* **by** *blast*
then have *U1*: C' B' A' *LtA* C' A' B'
proof −
have *f1*: ∀ *p pa pb pc pd pe pf pg ph pi pj pk pl pm*. ¬ *Tarski-neutral-dimensionless p pa* ∨ ¬ *Tarski-neutral-dimensionless.CongA p pa* (*pb*::′*p*) *pc pd pe pf pg* ∨ ¬ *Tarski-neutral-dimensionless.CongA p pa ph pi pj pk pl pm* ∨ ¬ *Tarski-neutral-dimensionless.LtA p pa pb pc pd ph pi pj* ∨ *Tarski-neutral-dimensionless.LtA p pa pe pf pg pk pl pm*
by (*simp add*: *Tarski-neutral-dimensionless.conga-preserves-lta*)
have *f2*: C' A'' A' *CongA* C' A' A''
by (*metis Z15 Z17*)
have *f3*: ∀ *p pa pb pc pd pe pf pg*. ¬ *Tarski-neutral-dimensionless p pa* ∨ ¬ *Tarski-neutral-dimensionless.CongA p pa* (*pb*::′*p*) *pc pd pe pf pg* ∨ *Tarski-neutral-dimensionless.CongA p pa pe pf pg pb pc pd*
by (*metis* (*no-types*) *Tarski-neutral-dimensionless.conga-sym*)
then have ¬ C' B' A' *LtA* C' A'' A' ∨ A'' B' C' *LtA* C' A' A''
using *f2 f1* **by** (*meson Tarski-neutral-dimensionless-axioms* ‹A'' B' C' *CongA* C' B' A'›)
then have C' B' A' *LtA* C' A' B' ∨ A'' B' C' *LtA* A'' A' C' ∨ $A'' = B'$
using *f2 f1* **by** (*metis* (*no-types*) *Tarski-neutral-dimensionless.conga-refl Tarski-neutral-dimensionless-axioms Z12* ‹A'' B' C' *CongA* C' B' A'› *lta-right-comm*)
thus *?thesis*
using *f3 f2 f1* **by** (*metis* (*no-types*) *Tarski-neutral-dimensionless-axioms Z12 Z20* ‹A'' B' C' *CongA* C' B' A'› *lta-right-comm*)
qed
then have *Z23*: C' A' *Lt* C' B'
using *Z22* **by** *auto*
have *Z24*: C' A'' *Lt* C' B'
using *Z16 Z23 cong2-lt--lt cong-reflexivity* **by** *blast*
have *Z25*: *C A Le C B*
proof −
have *Z26*: *Cong* C' A'' *C A*

```
      using Z5 not-cong-2143 by blast
    have Cong C′ B′ C B
      using assms(3) not-cong-4321 by blast
    thus ?thesis
      using l5-6 Z24 Z26 lt--le by blast
  qed
  then have Z27: Cong C A C B
    by (simp add: assms(4) le-anti-symmetry le-comm)
  have Cong C′ A′′ C′ B′
    by (metis Cong-perm Z13 Z27 assms(2) assms(3) cong-transitivity)
  then have False
    using Z24 cong--nlt by blast
  then have Cong B A B′ A′ by simp
}
{
  assume W1: Bet B′ A′ A′′
  have W2: A′ ≠ A′′
    using Z9A by auto
  have W3: A′ C′ B′ LtA C′ A′ A′′ ∧ A′ B′ C′ LtA C′ A′ A′′
    using W1 Z8 Z9A l11-41 by blast
  have W4: Cong A′ C′ A′′ C′
    using Z7 not-cong-3412 by blast
  have ¬ Col A′′ C′ A′
    by (metis Col-def W1 Z8 Z9A col-transitivity-1)
  then have W6: C′ A′′ A′ CongA C′ A′ A′′ ⟷ Cong C′ A′′ C′ A′
    using l11-44-1 by auto
  have W7: Cong C′ A′ C′ A′′
    using Z7 not-cong-4321 by blast
  then have W8: Cong C′ A′′ C′ A′
    using W4 not-cong-4321 by blast
  have W9: ¬ Col B′ C′ A′′
    by (simp add: Z9 not-col-permutation-1)
  have W10: B′ A′′ C′ CongA A′ A′′ C′
      by (metis Tarski-neutral-dimensionless.Out-def Tarski-neutral-dimensionless-axioms W1 Z9 Z9A bet-out-1
between-trivial not-col-distincts out2--conga)
  have W12: C′ B′ A′′ LtA C′ A′′ B′ ⟷ C′ A′′ Lt C′ B′
    by (simp add: W9 l11-44-2)
  have W12A: C′ B′ A′′ LtA C′ A′′ B′
  proof –
    have V1: A′ B′ C′ CongA C′ B′ A′′
      by (simp add: Z3 conga-right-comm)
    have A′ A′′ C′ CongA B′ A′′ C′
      by (metis Tarski-neutral-dimensionless.Out-def Tarski-neutral-dimensionless-axioms W1 ‹¬ Col A′′ C′ A′›
between-equality-2 not-col-distincts or-bet-out out2--conga out-col)
    then have C′ A′ A′′ CongA C′ A′′ B′
      by (meson W6 W8 conga-left-comm not-conga not-conga-sym)
    thus ?thesis
      using W3 V1 conga-preserves-lta by auto
  qed
  then have C′ A′′ Lt C′ B′ using W12 by auto
  then have W14: C′ A′ Lt C′ B′
    using W8 cong2-lt--lt cong-reflexivity by blast
  have W15: C A Le C B
  proof –
    have Q1: C′ A′′ Le C′ B′
      using W12 W12A lt--le by blast
    have Q2: Cong C′ A′′ C A
      using Z5 not-cong-2143 by blast
    have Cong C′ B′ C B
      using assms(3) not-cong-4321 by blast
    thus ?thesis using Q1 Q2 l5-6 by blast
  qed
  have C B Le C A
    by (simp add: assms(4) le-comm)
  then have Cong C A C B
    by (simp add: W15 le-anti-symmetry)
```

then have *Cong* C' A' C' B'
by (*metis Cong-perm assms*(*2*) *assms*(*3*) *cong-inner-transitivity*)
then have *False*
using *W14 cong--nlt* **by** *blast*
then have *Cong B A* B' A' **by** *simp*
}
then have *Cong B A* B' A'
using *Z10* ‹*Bet* B' A'' A' $\Longrightarrow$ *Cong B A* B' A'› **by** *blast*
}
{
assume $A'' = A'$
then have *Cong B A* B' A'
using *Z2 not-cong-3412* **by** *blast*
}
thus *?thesis*
using ‹$A'' \neq A' \Longrightarrow$ *Cong B A* B' A'› **by** *blast*
qed
have *P6*: *A B C Cong3* A' B' C'
using *Cong3-def Cong-perm P5 assms*(*2*) *assms*(*3*) **by** *blast*
thus *?thesis*
using *P2 P5 assms*(*1*) *assms*(*3*) *assms*(*4*) *l11-49 le-zero* **by** *blast*
qed

lemma *l11-53*:
assumes *Per D C B* **and**
$C \neq D$ **and**
$A \neq B$ **and**
$B \neq C$ **and**
Bet A B C
shows *C A D LtA C B D* $\wedge$ *B D Lt A D*
proof −
have *P1*: $C \neq A$
using *assms*(*3*) *assms*(*5*) *between-identity* **by** *blast*
have *P2*: ¬ *Col B A D*
by (*smt assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *bet-col bet-col1 col3 col-permutation-4 l8-9*)
have *P3*: $A \neq D$
using *P2 col-trivial-2* **by** *blast*
have *P4*: *C A D LtA C B D*
proof −
have *P4A*: *B D A LtA D B C* $\wedge$ *B A D LtA D B C*
by (*simp add*: *P2 assms*(*4*) *assms*(*5*) *l11-41*)
have *P4AA*:*A Out B C*
using *assms*(*3*) *assms*(*5*) *bet-out* **by** *auto*
have *A Out D D*
using *P3 out-trivial* **by** *auto*
then have *P4B*: *C A D CongA B A D* **using** *P4AA*
by (*simp add*: *out2--conga*)
then have *P4C*: *B A D CongA C A D*
by (*simp add*: *P4B conga-sym*)
have *D B C CongA C B D*
using *assms*(*1*) *assms*(*4*) *conga-pseudo-refl per-distinct-1* **by** *auto*
thus *?thesis*
using *P4A P4C conga-preserves-lta* **by** *blast*
qed
obtain B' **where** *P5*: *C Midpoint B* B' $\wedge$ *Cong D B D* B'
using *Per-def assms*(*1*) **by** *auto*
have *K2*: $A \neq B'$
using *Bet-cases P5 assms*(*4*) *assms*(*5*) *between-equality-2 midpoint-bet* **by** *blast*
{
assume *Col B D* B'
then have *Col B A D*
by (*metis Col-cases P5 assms*(*1*) *assms*(*2*) *assms*(*4*) *col2--eq midpoint-col midpoint-distinct-2 per-not-col*)
then have *False*
by (*simp add*: *P2*)
}
then have *P6*: ¬ *Col B D* B' **by** *blast*

then have *D B B′ CongA D B′ B* ⟷ *Cong D B D B′*
by (*simp add*: *l11-44-1*)
then have *D B B′ CongA D B′ B* **using** *P5* **by** *simp*
{
assume *K1*: *Col A D B′*
have *Col B′ A B*
using *Col-def P5 assms*(*4*) *assms*(*5*) *midpoint-bet outer-transitivity-between* **by** *blast*
then have *Col B′ B D*
using *K1 K2 Col-perm col-transitivity-2* **by** *blast*
then have *Col B D B′*
using *Col-perm* **by** *blast*
then have *False*
by (*simp add*: *P6*)
}
then have *K3B*: ¬ *Col A D B′* **by** *blast*
then have *K4*: *D A B′ LtA D B′ A* ⟷ *D B′ Lt D A*
by (*simp add*: *l11-44-2*)
have *K4A*: *C A D LtA C B′ D*
by (*metis Midpoint-def P1 P3 P4 P5 P5 P6 assms*(*2*) *assms*(*4*) *col-trivial-1 cong-reflexivity conga-preserves-lta conga-refl l11-51 not-cong-2134*)
have *D B′ Lt D A*
proof −
have *D A B′ LtA D B′ A*
proof −
have *K5A*: *A Out D D*
using *P3 out-trivial* **by** *auto*
have *K5AA*: *A Out B′ C*
by (*smt K2 Out-def P1 P5 assms*(*4*) *assms*(*5*) *midpoint-bet outer-transitivity-between2*)
then have *K5*: *D A C CongA D A B′*
by (*simp add*: *K5A out2--conga*)
have *K6A*: *B′ Out D D*
using *K3B not-col-distincts out-trivial* **by** *blast*
have *B′ Out A C*
by (*smt P5 K5AA assms*(*4*) *assms*(*5*) *between-equality-2 l6-4-2 midpoint-bet midpoint-distinct-2 out-col outer-transitivity-between2*
then have *K6*: *D B′ C CongA D B′ A*
by (*simp add*: *K6A out2--conga*)
have *D A C LtA D B′ C*
by (*simp add*: *K4A lta-comm*)
thus *?thesis*
using *K5 K6 conga-preserves-lta* **by** *auto*
qed
thus *?thesis*
by (*simp add*: *K4*)
qed
thus *?thesis*
using *P4 P5 cong2-lt--lt cong-pseudo-reflexivity not-cong-4312* **by** *blast*
qed

lemma *cong2-conga-obtuse--cong-conga2*:
assumes *Obtuse A B C* **and**
A B C CongA A′ B′ C′ **and**
Cong A C A′ C′ **and**
Cong B C B′ C′
shows *Cong B A B′ A′* ∧ *B A C CongA B′ A′ C′* ∧
B C A CongA B′ C′ A′
proof −
have *B C Le A C*
proof *cases*
assume *Col A B C*
thus *?thesis*
by (*simp add*: *assms*(*1*) *col-obtuse--bet l5-12-a*)
next
assume ¬ *Col A B C*
thus *?thesis*
using *l11-46 assms*(*1*) *lt--le not-col-distincts* **by** *auto*
qed

thus *?thesis*
using *l11-52 assms(2) assms(3) assms(4)* **by** *blast*
qed

lemma *cong2-per2--cong-conga2*:
assumes $A \neq B$ **and**
$B \neq C$ **and**
Per A B C **and**
Per A' B' C' **and**
Cong A C A' C' **and**
Cong B C B' C'
shows *Cong B A B' A'* $\wedge$ *B A C CongA B' A' C'* $\wedge$
B C A CongA B' C' A'
proof –
have *P1*: *B C Le A C* $\wedge$ $\neg$ *Cong B C A C*
using *assms(1) assms(2) assms(3) cong--nlt l11-46 lt--le* **by** *blast*
then have *A B C CongA A' B' C'*
using *assms(2) assms(3) assms(4) assms(5) assms(6) cong-diff cong-inner-transitivity cong-symmetry l11-16* **by** *blast*
thus *?thesis*
using *P1 assms(5) assms(6) l11-52* **by** *blast*
qed

lemma *cong2-per2--cong*:
assumes *Per A B C* **and**
Per A' B' C' **and**
Cong A C A' C' **and**
Cong B C B' C'
shows *Cong B A B' A'*
proof *cases*
assume $B = C$
thus *?thesis*
using *assms(3) assms(4) cong-reverse-identity not-cong-2143* **by** *blast*
next
assume $B \neq C$
show *?thesis*
proof *cases*
assume $A = B$
thus *?thesis*
proof –
have *Cong A C B' C'*
using ‹$A = B$› *assms(4)* **by** *blast*
then have $B' = A'$
by (*meson Cong3-def Per-perm assms(2) assms(3) cong-inner-transitivity cong-pseudo-reflexivity l8-10 l8-7*)
thus *?thesis*
using ‹$A = B$› *cong-trivial-identity* **by** *blast*
qed
next
assume $A \neq B$
show *?thesis*
proof *cases*
assume $A' = B'$
thus *?thesis*
by (*metis Cong3-def Per-perm* ‹$A \neq B$› *assms(1) assms(3) assms(4) cong-inner-transitivity cong-pseudo-reflexivity l8-10 l8-7*)
next
assume $A' \neq B'$
thus *?thesis*
using *cong2-per2--cong-conga2* ‹$A \neq B$› ‹$B \neq C$› *assms(1) assms(2) assms(3) assms(4)* **by** *blast*
qed
qed
qed

lemma *cong2-per2--cong-3*:
assumes *Per A B C*
Per A' B' C' **and**

Cong A C A′ C′ **and**
Cong B C B′ C′
shows *A B C Cong3 A′ B′ C′*
by (*metis Tarski-neutral-dimensionless.Cong3-def Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *cong2-per2--cong cong-3-swap*)

lemma *cong-lt-per2--lt*:
assumes *Per A B C* **and**
Per A′ B′ C′ **and**
Cong A B A′ B′ **and**
B C Lt B′ C′
shows *A C Lt A′ C′*
proof *cases*
assume *A = B*
thus *?thesis*
using *assms*(*3*) *assms*(*4*) *cong-reverse-identity* **by** *blast*
next
assume *A ≠ B*
show *?thesis*
proof *cases*
assume *B = C*
thus *?thesis*
by (*smt assms*(*2*) *assms*(*3*) *assms*(*4*) *cong2-lt--lt cong-4312 cong-diff cong-reflexivity l11-46 lt-diff*)
next
assume *P0*: *B ≠ C*
have *B C Lt B′ C′*
by (*simp add*: *assms*(*4*))
then have *R1*: *B C Le B′ C′ ∧ ¬ Cong B C B′ C′*
by (*simp add*: *Lt-def*)
then obtain *C0* **where** *P1*: *Bet B′ C0 C′ ∧ Cong B C B′ C0*
using *Le-def* **by** *auto*
then have *P2*: *Per A′ B′ C0*
by (*metis Col-def Per-cases assms*(*2*) *bet-out-1 col-col-per-per col-trivial-1 l8-5 out-diff2*)
have *C0 A′ Lt C′ A′* **using** *l11-53*
by (*metis P1 P2 R1 P0 bet--lt2313 between-symmetry cong-diff*)
then have *P3*: *A′ C0 Lt A′ C′*
using *Lt-cases* **by** *blast*
have *P4*: *Cong A′ C0 A C*
using *P1 P2 assms*(*1*) *assms*(*3*) *l10-12 not-cong-3412* **by** *blast*
have *Cong A′ C′ A′ C′*
by (*simp add*: *cong-reflexivity*)
thus *?thesis*
using *cong2-lt--lt P3 P4* **by** *blast*
qed
qed

lemma *cong-le-per2--le*:
assumes *Per A B C* **and**
Per A′ B′ C′ **and**
Cong A B A′ B′ **and**
B C Le B′ C′
shows *A C Le A′ C′*
proof *cases*
assume *Cong B C B′ C′*
thus *?thesis*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *cong--le l10-12* **by** *blast*
next
assume *¬ Cong B C B′ C′*
then have *B C Lt B′ C′*
using *Lt-def assms*(*4*) **by** *blast*
thus *?thesis*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *cong-lt-per2--lt lt--le* **by** *auto*
qed

lemma *lt2-per2--lt*:
assumes *Per A B C* **and**

Per A′ B′ C′ **and**
A B Lt A′ B′ **and**
B C Lt B′ C′
shows *A C Lt A′ C′*
proof −
have *P2*: *B A Lt B′ A′*
by (*simp add*: *assms*(*3*) *lt-comm*)
have *P3*: *B C Le B′ C′* ∧ ¬ *Cong B C B′ C′*
using *assms*(*4*) *cong--nlt lt--le* **by** *auto*
then obtain *C0* **where** *P4*: *Bet B′ C0 C′* ∧ *Cong B C B′ C0*
using *Le-def* **by** *auto*
have *P4A*: *B′* ≠ *C′*
using *assms*(*4*) *lt-diff* **by** *auto*
have *Col B′ C′ C0*
using *P4 bet-col not-col-permutation-5* **by** *blast*
then have *P5*: *Per A′ B′ C0*
using *assms*(*2*) *P4A per-col* **by** *blast*
have *P6*: *A C Lt A′ C0*
by (*meson P2 P4 P5 assms*(*1*) *cong-lt-per2--lt l8-2 lt-comm not-cong-2143*)
have *B′ C0 Lt B′ C′*
by (*metis P4 assms*(*4*) *bet--lt1213 cong--nlt*)
then have *A′ C0 Lt A′ C′*
using *P5 assms*(*2*) *cong-lt-per2--lt cong-reflexivity* **by** *blast*
thus *?thesis*
using *P6 lt-transitivity* **by** *blast*
qed

lemma *le-lt-per2--lt*:
assumes *Per A B C* **and**
Per A′ B′ C′ **and**
A B Le A′ B′ **and**
B C Lt B′ C′
shows *A C Lt A′ C′*
using *Lt-def assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *cong-lt-per2--lt lt2-per2--lt* **by** *blast*

lemma *le2-per2--le*:
assumes *Per A B C* **and**
Per A′ B′ C′ **and**
A B Le A′ B′ **and**
B C Le B′ C′
shows *A C Le A′ C′*
proof *cases*
assume *Cong B C B′ C′*
thus *?thesis*
by (*meson Per-cases Tarski-neutral-dimensionless.cong-le-per2--le Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*2*) *assms*(*3*) *le-comm not-cong-2143*)
next
assume ¬ *Cong B C B′ C′*
then have *B C Lt B′ C′*
by (*simp add*: *Lt-def assms*(*4*))
thus *?thesis*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *le-lt-per2--lt lt--le* **by** *blast*
qed

lemma *cong-lt-per2--lt-1*:
assumes *Per A B C* **and**
Per A′ B′ C′ **and**
A B Lt A′ B′ **and**
Cong A C A′ C′
shows *B′ C′ Lt B C*
by (*meson Gt-def assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *cong2-per2--cong cong-4321 cong--nlt cong-symmetry lt2-per2--lt or-lt-cong-gt*)

lemma *symmetry-preserves-conga*:
assumes *A* ≠ *B* **and** *C* ≠ *B* **and**
M Midpoint A A′ **and**

M Midpoint B B′ **and**
M Midpoint C C′
shows *A B C CongA A′ B′ C′*
by (*metis Mid-perm assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *conga-trivial-1 l11-51 l7-13 symmetric-point-uniqueness*)

lemma *l11-57*:
assumes *A A′ OS B B′* **and**
Per B A A′ **and**
Per B′ A′ A **and**
A A′ OS C C′ **and**
Per C A A′ **and**
Per C′ A′ A
shows *B A C CongA B′ A′ C′*
proof −
obtain *M* **where** *P1*: *M Midpoint A A′*
using *midpoint-existence* **by** *auto*
obtain *B′′* **where** *P2*: *M Midpoint B B′′*
using *symmetric-point-construction* **by** *auto*
obtain *C′′* **where** *P3*: *M Midpoint C C′′*
using *symmetric-point-construction* **by** *auto*
have *P4*: ¬ *Col A A′ B*
using *assms*(*1*) *col123--nos* **by** *auto*
have *P5*: ¬ *Col A A′ C*
using *assms*(*4*) *col123--nos* **by** *auto*
have *P6*: *B A C CongA B′′ A′ C′′*
by (*metis P1 P2 P3 assms*(*1*) *assms*(*4*) *os-distincts symmetry-preserves-conga*)
have *B′′ A′ C′′ CongA B′ A′ C′*
proof −
have *B* ≠ *M*
using *P1 P4 midpoint-col not-col-permutation-2* **by** *blast*
then have *P7*: ¬ *Col B′′ A A′*
using *Mid-cases P1 P2 P4 mid-preserves-col not-col-permutation-3* **by** *blast*
have *K3*: *Bet B′′ A′ B′*
proof −
have *Per B′′ A′ A*
using *P1 P2 assms*(*2*) *per-mid-per* **by** *blast*
have *Col B B′′ M* ∧ *Col A A′ M*
using *P1 P2 midpoint-col not-col-permutation-2* **by** *blast*
then have *Coplanar B A A′ B′′*
using *Coplanar-def* **by** *auto*
then have *Coplanar A B′ B′′ A′*
by (*meson assms*(*1*) *between-trivial2 coplanar-trans-1 ncoplanar-perm-4 ncoplanar-perm-8 one-side-chara os--coplanar*)
then have *P8*: *Col B′ B′′ A′*
using *cop-per2--col P1 P2 P7 assms*(*2*) *assms*(*3*) *not-col-distincts per-mid-per* **by** *blast*
have *A A′ TS B B′′*
using *P1 P2 P4 mid-two-sides* **by** *auto*
then have *A′ A TS B′′ B′*
using *assms*(*1*) *invert-two-sides l9-2 l9-8-2* **by** *blast*
thus *?thesis*
using *Col-cases P8 col-two-sides-bet* **by** *blast*
qed
have ¬ *Col C′′ A A′*
by (*smt Col-def P1 P3 P5 l7-15 l7-2 not-col-permutation-5*)
have *Bet C′′ A′ C′*
proof −
have *Z2*: *Col C′ C′′ A′*
proof −
have *Col C C′′ M* ∧ *Col A A′ M*
using *P1 P3 col-permutation-1 midpoint-col* **by** *blast*
then have *Coplanar C A A′ C′′*
using *Coplanar-def* **by** *blast*
then have *Z1*: *Coplanar A C′ C′′ A′*
by (*meson assms*(*4*) *between-trivial2 coplanar-trans-1 ncoplanar-perm-4 ncoplanar-perm-8 one-side-chara os--coplanar*)
have *Per C′′ A′ A*
using *P1 P3 assms*(*5*) *per-mid-per* **by** *blast*

thus *?thesis*
using *Z1 P5 assms*(*6*) *col-trivial-1 cop-per2--col* **by** *blast*
qed
have *A A′ TS C C″*
using *P1 P3 P5 mid-two-sides* **by** *auto*
then have *A′ A TS C″ C′*
using *assms*(*4*) *invert-two-sides l9-2 l9-8-2* **by** *blast*
thus *?thesis*
using *Col-cases Z2 col-two-sides-bet* **by** *blast*
qed
thus *?thesis*
by (*metis P6 K3 assms*(*1*) *assms*(*4*) *conga-diff45 conga-diff56 l11-14 os-distincts*)
qed
thus *?thesis*
using *P6 conga-trans* **by** *blast*
qed

lemma *cop3-orth-at--orth-at*:
assumes ¬ *Col D E F* **and**
Coplanar A B C D **and**
Coplanar A B C E **and**
Coplanar A B C F **and**
X OrthAt A B C U V
shows *X OrthAt D E F U V*
proof −
have *P1*: ¬ *Col A B C* ∧ *Coplanar A B C X*
using *OrthAt-def assms*(*5*) **by** *blast*
then have *P2*: *Coplanar D E F X*
using *assms*(*2*) *assms*(*3*) *assms*(*4*) *coplanar-pseudo-trans* **by** *blast*
{
fix *M*
assume *Coplanar A B C M*
then have *Coplanar D E F M*
using *P1 assms*(*2*) *assms*(*3*) *assms*(*4*) *coplanar-pseudo-trans* **by** *blast*
}
have *T1*: $U \neq V$
using *OrthAt-def assms*(*5*) **by** *blast*
have *T2*: *Col U V X*
using *OrthAt-def assms*(*5*) **by** *auto*
{
fix *P Q*
assume *P7*: *Coplanar D E F P* ∧ *Col U V Q*
then have *Coplanar A B C P*
by (*meson* ‹⋀*M*. *Coplanar A B C M* ⟹ *Coplanar D E F M*› *assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *l9-30*)
then have *Per P X Q* **using** *P7 OrthAt-def assms*(*5*) **by** *blast*
}
thus *?thesis* **using** *assms*(*1*)
by (*simp add*: *OrthAt-def P2 T1 T2*)
qed

lemma *col2-orth-at--orth-at*:
assumes $U \neq V$ **and**
Col P Q U **and**
Col P Q V **and**
X OrthAt A B C P Q
shows *X OrthAt A B C U V*
proof −
have *Col P Q X*
using *OrthAt-def assms*(*4*) **by** *auto*
then have *Col U V X*
by (*metis OrthAt-def assms*(*2*) *assms*(*3*) *assms*(*4*) *col3*)
thus *?thesis*
using *OrthAt-def assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *colx* **by** *presburger*
qed

lemma *col-orth-at--orth-at*:

assumes *U* ≠ *W* **and**
Col U V W **and**
X OrthAt A B C U V
shows *X OrthAt A B C U W*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *col2-orth-at--orth-at col-trivial-3* **by** *blast*

lemma *orth-at-symmetry*:
assumes *X OrthAt A B C U V*
shows *X OrthAt A B C V U*
by (*metis assms col2-orth-at--orth-at col-trivial-2 col-trivial-3*)

lemma *orth-at-distincts*:
assumes *X OrthAt A B C U V*
shows *A* ≠ *B* ∧ *B* ≠ *C* ∧ *A* ≠ *C* ∧ *U* ≠ *V*
using *OrthAt-def assms not-col-distincts* **by** *fastforce*

lemma *orth-at-chara*:
X OrthAt A B C X P ⟷
(¬ *Col A B C* ∧ *X* ≠ *P* ∧ *Coplanar A B C X* ∧ (∀ *D*.(*Coplanar A B C D* ⟶ *Per D X P*)))
proof −
{
assume *X OrthAt A B C X P*
then have ¬ *Col A B C* ∧ *X* ≠ *P* ∧ *Coplanar A B C X* ∧ (∀ *D*.(*Coplanar A B C D* ⟶ *Per D X P*))
using *OrthAt-def col-trivial-2* **by** *auto*
}
{
assume *T1*: ¬ *Col A B C* ∧ *X* ≠ *P* ∧ *Coplanar A B C X* ∧ (∀ *D*.(*Coplanar A B C D* ⟶ *Per D X P*))
{
fix *P0 Q*
assume *Coplanar A B C P0* ∧ *Col X P Q*
then have *Per P0 X Q* **using** *T1 OrthAt-def per-col* **by** *auto*
}
then have *X OrthAt A B C X P*
by (*simp add*: *T1* ‹⋀*Q P0*. *Coplanar A B C P0* ∧ *Col X P Q* ⟹ *Per P0 X Q*› *Tarski-neutral-dimensionless.OrthAt-def Tarski-neutral-dimensionless-axioms col-trivial-3*)
}
thus *?thesis*
using ‹*X OrthAt A B C X P* ⟹ ¬ *Col A B C* ∧ *X* ≠ *P* ∧ *Coplanar A B C X* ∧ (∀*D*. *Coplanar A B C D* ⟶ *Per D X P*)› **by** *blast*
qed

lemma *cop3-orth--orth*:
assumes ¬ *Col D E F* **and**
Coplanar A B C D **and**
Coplanar A B C E **and**
Coplanar A B C F **and**
A B C Orth U V
shows *D E F Orth U V*
using *Orth-def assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *cop3-orth-at--orth-at* **by** *blast*

lemma *col2-orth--orth*:
assumes *U* ≠ *V* **and**
Col P Q U **and**
Col P Q V **and**
A B C Orth P Q
shows *A B C Orth U V*
by (*meson Orth-def Tarski-neutral-dimensionless.col2-orth-at--orth-at Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*))

lemma *col-orth--orth*:
assumes *U* ≠ *W* **and**
Col U V W **and**
A B C Orth U V
shows *A B C Orth U W*
by (*meson assms*(*1*) *assms*(*2*) *assms*(*3*) *col2-orth--orth col-trivial-3*)

lemma *orth-symmetry*:
  assumes *A B C Orth U V*
  shows *A B C Orth V U*
  by (*meson Orth-def assms orth-at-symmetry*)

lemma *orth-distincts*:
  assumes *A B C Orth U V*
  shows $A \neq B \wedge B \neq C \wedge A \neq C \wedge U \neq V$
  using *Orth-def assms orth-at-distincts* **by** *blast*

lemma *col-cop-orth--orth-at*:
  assumes *A B C Orth U V* **and**
    Coplanar A B C X **and**
    Col U V X
  shows *X OrthAt A B C U V*
proof –
  obtain *Y* **where** *P1*:
    ¬ *Col A B C* ∧ *U* ≠ *V* ∧ *Coplanar A B C Y* ∧ *Col U V Y* ∧
(∀ *P Q*. (*Coplanar A B C P* ∧ *Col U V Q*) ⟶ *Per P Y Q*)
    by (*metis OrthAt-def Tarski-neutral-dimensionless.Orth-def Tarski-neutral-dimensionless-axioms assms*(*1*))
  then have *P2*: *X* = *Y*
    using *assms*(*2*) *assms*(*3*) *per-distinct-1* **by** *blast*
  {
    fix *P Q*
    assume *Coplanar A B C P* ∧ *Col U V Q*
    then have *Per P X Q* **using** *P1 P2* **by** *auto*
  }
  thus *?thesis*
    using *OrthAt-def Orth-def assms*(*1*) *assms*(*2*) *assms*(*3*) **by** *auto*
qed

lemma *l11-60-aux*:
  assumes ¬ *Col A B C* **and**
    Cong A P A Q **and**
    Cong B P B Q **and**
    Cong C P C Q **and**
    Coplanar A B C D
  shows *Cong D P D Q*
proof –
  obtain *M* **where** *P1*: *Bet P M Q* ∧ *Cong P M M Q*
    by (*meson Midpoint-def Tarski-neutral-dimensionless.midpoint-existence Tarski-neutral-dimensionless-axioms*)
  obtain *X* **where** *P2*: (*Col A B X* ∧ *Col C D X*) ∨
            (*Col A C X* ∧ *Col B D X*) ∨
            (*Col A D X* ∧ *Col B C X*)
    using *assms*(*5*) *Coplanar-def* **by** *auto*
  {
    assume *Col A B X* ∧ *Col C D X*
    then have *Cong D P D Q*
      by (*metis* (*no-types*, *lifting*) *assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *l4-17 not-col-distincts not-col-permutation-5*)
  }
  {
    assume *Col A C X* ∧ *Col B D X*
    then have *Cong D P D Q*
      by (*metis* (*no-types*, *lifting*) *assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *l4-17 not-col-distincts not-col-permutation-5*)
  }
  {
    assume *Col A D X* ∧ *Col B C X*
    then have *Cong D P D Q*
      by (*smt assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *l4-17 not-col-distincts not-col-permutation-1*)
  }
  thus *?thesis*
    using *P2* ‹*Col A B X* ∧ *Col C D X* ⟹ *Cong D P D Q*› ‹*Col A C X* ∧ *Col B D X* ⟹ *Cong D P D Q*› **by** *blast*
qed

lemma *l11-60*:
  assumes ¬ *Col A B C* **and**

Per A D P **and**
Per B D P **and**
Per C D P **and**
Coplanar A B C E
shows Per E D P
by (meson Per-def assms(1) assms(2) assms(3) assms(4) assms(5) l11-60-aux per-double-cong)

lemma l11-60-bis:
assumes ¬ Col A B C **and**
D ≠ P **and**
Coplanar A B C D **and**
Per A D P **and**
Per B D P **and**
Per C D P
shows D OrthAt A B C D P
using assms(1) assms(2) assms(3) assms(4) assms(5) assms(6) l11-60 orth-at-chara **by** auto

lemma l11-61:
assumes A ≠ A' **and**
A ≠ B **and**
A ≠ C **and**
Coplanar A A' B B' **and**
Per B A A' **and**
Per B' A' A **and**
Coplanar A A' C C' **and**
Per C A A' **and**
Per B A C
shows Per B' A' C'
proof −
have P1: ¬ Col C A A'
using assms(1) assms(3) assms(8) per-col-eq **by** blast
obtain C'' **where** P2: A A' Perp C'' A' ∧ A A' OS C C'' **using** l10-15
using Col-perm P1 col-trivial-2 **by** blast
have P6: B' ≠ A
using assms(1) assms(6) per-distinct **by** blast
have P8: ¬ Col A' A C''
using P2 not-col-permutation-4 one-side-not-col124 **by** blast
have P9: Per A' A' B'
by (simp add: l8-2 l8-5)
have P10: Per A A' B'
by (simp add: assms(6) l8-2)
{
fix B'
assume A A' OS B B' ∧ Per B' A' A
then have B A C CongA B' A' C'' **using** l11-17
by (meson P2 Perp-cases Tarski-neutral-dimensionless.l11-57 Tarski-neutral-dimensionless-axioms assms(5) assms(8) perp-per-1)
then have Per B' A' C''
using assms(9) l11-17 **by** blast
}
then have Q1: ∀ B'. (A A' OS B B' ∧ Per B' A' A) ⟶ Per B' A' C'' **by** simp
{
fix B'
assume P12: Coplanar A A' B B' ∧ Per B' A' A ∧ B' ≠ A
have Per B' A' C''
proof cases
assume B' = A'
thus ?thesis
by (simp add: Per-perm l8-5)
next
assume P13: B' ≠ A'
have P14: ¬ Col B' A' A
using P12 P13 assms(1) l8-9 **by** auto
have P15: ¬ Col B A A'
using assms(1) assms(2) assms(5) per-not-col **by** auto
then have Z1: A A' TS B B' ∨ A A' OS B B'

using *P12 P14 cop--one-or-two-sides not-col-permutation-5* **by** *blast*
{
assume *A A′ OS B B′*
then have *Per B′ A′ C′′*
by (*simp add*: *P12* ‹⋀*B′a*. *A A′ OS B B′a* ∧ *Per B′a A′ A* ⟹ *Per B′a A′ C′′*›)
}
{
assume *Q2*: *A A′ TS B B′*
obtain *B′′* **where** *Z2*: *Bet B′ A′ B′′* ∧ *Cong A′ B′′ A′ B′*
using *segment-construction* **by** *blast*
have *B′* ≠ *B′′*
using *P13 Z2 bet-neq12--neq* **by** *blast*
then have *Z4*: *A′* ≠ *B′′*
using *Z2 cong-diff-4* **by** *blast*
then have *A A′ TS B′′ B′*
by (*meson TS-def Z2 Q2 bet--ts invert-two-sides l9-2 not-col-permutation-1*)
then have *Z5*: *A A′ OS B B′′*
using *Q2 l9-8-1* **by** *auto*
have *Per B′′ A′ A*
using *P12 P13 Z2 bet-col col-per2--per l8-2 l8-5* **by** *blast*
then have *Per C′′ A′ B′′*
using *l8-2 Q1 Z5* **by** *blast*
then have *Per B′ A′ C′′*
by (*metis Col-def Per-perm Tarski-neutral-dimensionless.l8-3 Tarski-neutral-dimensionless-axioms Z2 Z4*)
}
thus *?thesis* **using** *Z1*
using ‹*A A′ OS B B′* ⟹ *Per B′ A′ C′′*› **by** *blast*
qed
}
then have ∀ *B′*. (*Coplanar A A′ B B′* ∧ *Per B′ A′ A* ∧ *B′* ≠ *A*) ⟶ *Per B′ A′ C′′*
by *simp*
then have *Per B′ A′ C′′*
using *P6 assms*(*4*) *assms*(*6*) **by** *blast*
then have *P11*: *Per C′′ A′ B′*
using *Per-cases* **by** *auto*
have *Coplanar A′ A C′′ C′*
by (*meson P1 P2 assms*(*7*) *coplanar-trans-1 ncoplanar-perm-6 ncoplanar-perm-8 os--coplanar*)
thus *?thesis*
using *P8 P9 P10 P11 l8-2 l11-60* **by** *blast*
qed

lemma *l11-61-bis*:
assumes *D OrthAt A B C D P* **and**
D E Perp E Q **and**
Coplanar A B C E **and**
Coplanar D E P Q
shows *E OrthAt A B C E Q*
proof −
have *P4*: *D* ≠ *E*
using *assms*(*2*) *perp-not-eq-1* **by** *auto*
have *P5*: *E* ≠ *Q*
using *assms*(*2*) *perp-not-eq-2* **by** *auto*
have ∃ *D′*. (*D E Perp D′ D* ∧ *Coplanar A B C D′*)
proof −
obtain *F* **where** *T1*: *Coplanar A B C F* ∧ ¬ *Col D E F*
using *P4 ex-ncol-cop* **by** *blast*
obtain *D′* **where** *T2*: *D E Perp D′ D* ∧ *Coplanar D E F D′*
using *P4 ex-perp-cop* **by** *blast*
have *Coplanar A B C D′*
proof −
have *T3A*: ¬ *Col A B C*
using *OrthAt-def assms*(*1*) **by** *auto*
have *T3B*: *Coplanar A B C D*
using *OrthAt-def assms*(*1*) **by** *blast*
then have *T4*: *Coplanar D E F A*
by (*meson T1 T3A assms*(*3*) *coplanar-pseudo-trans ncop-distincts*)

```
      have T5: Coplanar D E F B
        using T1 T3A T3B assms(3) coplanar-pseudo-trans ncop-distincts by blast
      have Coplanar D E F C
        using T1 T3A T3B assms(3) coplanar-pseudo-trans ncop-distincts by blast
      thus ?thesis
        using T1 T2 T4 T5 coplanar-pseudo-trans by blast
    qed
    thus ?thesis
      using T2 by auto
  qed
  then obtain D' where R1: D E Perp D' D ∧ Coplanar A B C D' by auto
  then have R2: D ≠ D'
    using perp-not-eq-2 by blast
  {
    fix M
    assume R3: Coplanar A B C M
    have Col D P P
      by (simp add: col-trivial-2)
    then have Per E D P
      using assms(1) assms(3) orth-at-chara by auto
    then have R4: Per P D E using l8-2 by auto
    have R5: Per Q E D
      using Perp-cases assms(2) perp-per-2 by blast
    have R6: Coplanar D E D' M
    proof -
      have S1: ¬ Col A B C
        using OrthAt-def assms(1) by auto
      have Coplanar A B C D
        using OrthAt-def assms(1) by auto
      thus ?thesis
        using S1 assms(3) R1 R3 coplanar-pseudo-trans by blast
    qed
    have R7: Per D' D E
      using Perp-cases R1 perp-per-1 by blast
    have Per D' D P
      using R1 assms(1) orth-at-chara by blast
    then have Per P D D'
      using Per-cases by blast
    then have Per Q E M
      using l11-61 R4 R5 R6 R7 OrthAt-def P4 R2 assms(1) assms(4) by blast
    then have Per M E Q using l8-2 by auto
  }
  {
    fix P0 Q0
    assume Coplanar A B C P0 ∧ Col E Q Q0
    then have Per P0 E Q0
      using P5 ‹⋀M. Coplanar A B C M ⟹ Per M E Q› per-col by blast
  }
  thus ?thesis
    using OrthAt-def P5 assms(1) assms(3) col-trivial-3 by auto
qed

lemma l11-62-unicity:
  assumes Coplanar A B C D and
    Coplanar A B C D' and
    ∀ E. Coplanar A B C E ⟶ Per E D P and
    ∀ E. Coplanar A B C E ⟶ Per E D' P
  shows D = D'
  by (metis assms(1) assms(2) assms(3) assms(4) l8-8 not-col-distincts per-not-colp)

lemma l11-62-unicity-bis:
  assumes X OrthAt A B C X U and
    Y OrthAt A B C Y U
  shows X = Y
proof -
  have P1: Coplanar A B C X
```

```
    using assms(1) orth-at-chara by blast
  have P2: Coplanar A B C Y
    using assms(2) orth-at-chara by blast
  {
    fix E
    assume Coplanar A B C E
    then have Per E X U
      using OrthAt-def assms(1) col-trivial-2 by auto
  }
  {
    fix E
    assume Coplanar A B C E
    then have Per E Y U
      using assms(2) orth-at-chara by auto
  }
  thus ?thesis
    by (meson P1 P2 ‹⋀E. Coplanar A B C E ⟹ Per E X U› l8-2 l8-7)
qed

lemma orth-at2--eq:
  assumes X OrthAt A B C U V and
    Y OrthAt A B C U V
  shows X = Y
proof −
  have P1: Coplanar A B C X
    using assms(1)
    by (simp add: OrthAt-def)
  have P2: Coplanar A B C Y
    using OrthAt-def assms(2) by auto
  {
    fix E
    assume Coplanar A B C E
    then have Per E X U
      using OrthAt-def assms(1) col-trivial-3 by auto
  }
  {
    fix E
    assume Coplanar A B C E
    then have Per E Y U
      using OrthAt-def assms(2) col-trivial-3 by auto
  }
  thus ?thesis
    by (meson P1 P2 Per-perm ‹⋀E. Coplanar A B C E ⟹ Per E X U› l8-7)
qed

lemma col-cop-orth-at--eq:
  assumes X OrthAt A B C U V and
    Coplanar A B C Y and
    Col U V Y
  shows X = Y
proof −
  have Y OrthAt A B C U V
    using Orth-def assms(1) assms(2) assms(3) col-cop-orth--orth-at by blast
  thus ?thesis
    using assms(1) orth-at2--eq by auto
qed

lemma orth-at--ncop1:
  assumes U ≠ X and
    X OrthAt A B C U V
  shows ¬ Coplanar A B C U
  using assms(1) assms(2) col-cop-orth-at--eq not-col-distincts by blast

lemma orth-at--ncop2:
  assumes V ≠ X and
    X OrthAt A B C U V
```

shows ¬ *Coplanar A B C V*
using *assms(1) assms(2) col-cop-orth-at--eq not-col-distincts* **by** *blast*

lemma *orth-at--ncop*:
assumes *X OrthAt A B C X P*
shows ¬ *Coplanar A B C P*
by (*metis assms orth-at--ncop2 orth-at-distincts*)

lemma *l11-62-existence*:
∃ *D*. (*Coplanar A B C D* ∧ (∀ *E*. (*Coplanar A B C E* ⟶ *Per E D P*)))
proof *cases*
assume *Coplanar A B C P*
thus *?thesis*
using *l8-5* **by** *auto*
next
assume *P1*: ¬ *Coplanar A B C P*
then have *P2*: ¬ *Col A B C*
using *ncop--ncol* **by** *auto*
have ¬ *Col A B P*
using *P1 ncop--ncols* **by** *auto*
then obtain *D0* **where** *P4*: *Col A B D0* ∧ *A B Perp P D0* **using** *l8-18-existence* **by** *blast*
have *P5*: *Coplanar A B C D0*
using *P4 ncop--ncols* **by** *auto*
have *A* ≠ *B*
using *P2 not-col-distincts* **by** *auto*
then obtain *D1* **where** *P10*: *A B Perp D1 D0* ∧ *Coplanar A B C D1*
using *ex-perp-cop* **by** *blast*
have *P11*: ¬ *Col A B D1*
using *P10 P4 perp-not-col2* **by** *blast*
{
fix *D*
assume *Col D0 D1 D*
then have *Coplanar A B C D*
by (*metis P10 P5 col-cop2--cop perp-not-eq-2*)
}
obtain *A0* **where** *P11*: *A* ≠ *A0* ∧ *B* ≠ *A0* ∧ *D0* ≠ *A0* ∧ *Col A B A0*
using *P4 diff-col-ex3* **by** *blast*
have *P12*: *Coplanar A B C A0*
using *P11 ncop--ncols* **by** *blast*
have *P13*: *Per P D0 A0*
using *l8-16-1 P11 P4* **by** *blast*
show *?thesis*
proof *cases*
assume *Z1*: *Per P D0 D1*
{
fix *E*
assume *R1*: *Coplanar A B C E*
have *R2*: ¬ *Col A0 D1 D0*
by (*metis P10 P11 P4 col-permutation-5 colx perp-not-col2*)
have *R3*: *Per A0 D0 P*
by (*simp add*: *P13 l8-2*)
have *R4*: *Per D1 D0 P*
by (*simp add*: *Z1 l8-2*)
have *R5*: *Per D0 D0 P*
by (*simp add*: *l8-2 l8-5*)
have *Coplanar A0 D1 D0 E*
using *R1 P2 P12 P10 P5 coplanar-pseudo-trans* **by** *blast*
then have *Per E D0 P*
using *l11-60 R2 R3 R4 R5* **by** *blast*
}
thus *?thesis* **using** *P5* **by** *auto*
next
assume *S1*: ¬ *Per P D0 D1*
{
assume *S2*: *Col D0 D1 P*
have ∀ *D*. *Col D0 D1 D* ⟶ *Coplanar A B C D*

by (*simp add*: ‹⋀*Da*. *Col D0 D1 Da* ⟹ *Coplanar A B C Da*›)
then have *False*
using *P1 S2* **by** *blast*
}
then have *S2A*: ¬ *Col D0 D1 P* **by** *blast*
then obtain *D* **where** *S3*: *Col D0 D1 D* ∧ *D0 D1 Perp P D*
using *l8-18-existence* **by** *blast*
have *S4*: *Coplanar A B C D*
by (*simp add*: *S3* ‹⋀*Da*. *Col D0 D1 Da* ⟹ *Coplanar A B C Da*›)
{
fix *E*
assume *S5*: *Coplanar A B C E*
have *S6*: *D* ≠ *D0*
using *S1 S3 l8-2 perp-per-1* **by** *blast*
have *S7*: *Per D0 D P*
by (*metis Perp-cases S3 S6 perp-col perp-per-1*)
have *S8*: *Per D D0 A0*
proof −
have *V4*: *D0* ≠ *D1*
using *P10 perp-not-eq-2* **by** *blast*
have *V6*: *Per A0 D0 D1*
using *P10 P11 P4 l8-16-1 l8-2* **by** *blast*
thus *?thesis*
using *S3 V4 V6 l8-2 per-col* **by** *blast*
qed
have *S9*: *Per A0 D P*
proof −
obtain *A0′* **where** *W1*: *D Midpoint A0 A0′*
using *symmetric-point-construction* **by** *auto*
obtain *D0′* **where** *W2*: *D Midpoint D0 D0′*
using *symmetric-point-construction* **by** *auto*
have *Cong P A0 P A0′*
proof −
have *V3*: *Cong P D0 P D0′*
using *S7 W2 l8-2 per-double-cong* **by** *blast*
have *V4*: *Cong D0 A0 D0′ A0′*
using *W1 W2 cong-4321 l7-13* **by** *blast*
have *Per P D0′ A0′*
proof −
obtain *P′* **where** *V5*: *D Midpoint P P′*
using *symmetric-point-construction* **by** *blast*
have *Per P′ D0 A0*
proof −
have ¬ *Col P D D0*
by (*metis S2A S3 S6 col2--eq col-permutation-1*)
thus *?thesis*
by (*metis* (*full-types*) *P13 S3 S8 V5 S2A col-per2--per midpoint-col*)
qed
thus *?thesis*
using *midpoint-preserves-per V5 Mid-cases W1 W2* **by** *blast*
qed
thus *?thesis* **using** *l10-12 P13 V3 V4* **by** *blast*
qed
thus *?thesis*
using *Per-def Per-perm W1* **by** *blast*
qed
have *S13*: *Per D D P*
using *Per-perm l8-5* **by** *blast*
have *S14*: ¬ *Col D0 A0 D* **using** *P11 S7 S9 per-not-col Col-perm S6 S8* **by** *blast*
have *Coplanar A B C D* **using** *S4* **by** *auto*
then have *Coplanar D0 A0 D E*
using *P12 P2 P5 S5 coplanar-pseudo-trans* **by** *blast*
then have *Per E D P*
using *S13 S14 S7 S9 l11-60* **by** *blast*
}
thus *?thesis* **using** *S4* **by** *blast*

```
  qed
qed

lemma l11-62-existence-bis:
  assumes ¬ Coplanar A B C P
  shows ∃ X. X OrthAt A B C X P
proof −
  obtain X where P1: Coplanar A B C X ∧ (∀ E. Coplanar A B C E ⟶ Per E X P)
    using l11-62-existence by blast
  then have P2: X ≠ P
    using assms by auto
  have P3: ¬ Col A B C
    using assms ncop--ncol by auto
  thus ?thesis
    using P1 P2 P3 orth-at-chara by auto
qed

lemma l11-63-aux:
  assumes Coplanar A B C D and
    D ≠ E and
    E OrthAt A B C E P
  shows ∃ Q. (D E OS P Q ∧ A B C Orth D Q)
proof −
  have P1: ¬ Col A B C
    using OrthAt-def assms(3) by blast
  have P2: E ≠ P
    using OrthAt-def assms(3) by blast
  have P3: Coplanar A B C E
    using OrthAt-def assms(3) by blast
  have P4: ∀ P0 Q. (Coplanar A B C P0 ∧ Col E P Q) ⟶ Per P0 E Q
    using OrthAt-def assms(3) by blast
  have P5: ¬ Coplanar A B C P
    using assms(3) orth-at--ncop by auto
  have P6: Col D E D
    by (simp add: col-trivial-3)
  have ¬ Col D E P
    using P3 P5 assms(1) assms(2) col-cop2--cop by blast
  then obtain Q where P6: D E Perp Q D ∧ D E OS P Q
    using P6 l10-15 by blast
  have A B C Orth D Q
  proof −
    obtain F where P7: Coplanar A B C F ∧ ¬ Col D E F
      using assms(2) ex-ncol-cop by blast
    obtain D' where P8: D E Perp D' D ∧ Coplanar D E F D'
      using assms(2) ex-perp-cop by presburger
    have P9: ¬ Col D' D E
      using P8 col-permutation-1 perp-not-col by blast
    have P10: Coplanar D E F A
      by (meson P1 P3 P7 assms(1) coplanar-pseudo-trans ncop-distincts)
    have P11: Coplanar D E F B
      by (meson P1 P3 P7 assms(1) coplanar-pseudo-trans ncop-distincts)
    have P12: Coplanar D E F C
      by (meson P1 P3 P7 assms(1) coplanar-pseudo-trans ncop-distincts)
    then have D OrthAt A B C D Q
    proof −
      have Per D' D Q
      proof −
        obtain E' where Y1: D E Perp E' E ∧ Coplanar D E F E'
          using assms(2) ex-perp-cop by blast
        have Y2: E ≠ E'
          using Y1 perp-distinct by auto
        have Y5: Coplanar E D E' D'
          by (meson P7 P8 Y1 coplanar-perm-12 coplanar-perm-7 coplanar-trans-1 not-col-permutation-2)
        have Y6: Per E' E D
          by (simp add: Perp-perm Tarski-neutral-dimensionless.perp-per-2 Tarski-neutral-dimensionless-axioms Y1)
        have Y7: Per D' D E
```

using *P8 col-trivial-2 col-trivial-3 l8-16-1* **by** *blast*
have *Y8*: *Coplanar E D P Q*
using *P6 ncoplanar-perm-6 os--coplanar* **by** *blast*
have *Y9*: *Per P E D* **using** *P4*
using *assms(1) assms(3) l8-2 orth-at-chara* **by** *blast*
have *Y10*: *Coplanar A B C E'*
using *P10 P11 P12 P7 Y1 coplanar-pseudo-trans* **by** *blast*
then have *Y11*: *Per E' E P*
using *P4 Y10 col-trivial-2* **by** *auto*
have $E \neq D$ **using** *assms(2)* **by** *blast*
thus *?thesis*
using *l11-61 Y2 assms(2) P2 Y5 Y6 Y7 Y8 Y9 Y10 Y11* **by** *blast*
qed
then have *X1*: *D OrthAt D' D E D Q* **using** *l11-60-bis*
by (*metis OS-def P6 P9 Per-perm TS-def Tarski-neutral-dimensionless.l8-5 Tarski-neutral-dimensionless-axioms col-trivial-3 invert-one-side ncop-distincts perp-per-1*)
have *X3*: *Coplanar D' D E A*
using *P10 P7 P8 coplanar-perm-14 coplanar-trans-1 not-col-permutation-3* **by** *blast*
have *X4*: *Coplanar D' D E B*
using *P11 P7 P8 coplanar-perm-14 coplanar-trans-1 not-col-permutation-3* **by** *blast*
have *Coplanar D' D E C*
using *P12 P7 P8 coplanar-perm-14 coplanar-trans-1 not-col-permutation-3* **by** *blast*
thus *?thesis*
using *X1 P1 X3 X4 cop3-orth-at--orth-at* **by** *blast*
qed
thus *?thesis*
using *Orth-def* **by** *blast*
qed
thus *?thesis* **using** *P6* **by** *blast*
qed

lemma *l11-63-existence*:
assumes *Coplanar A B C D* **and**
¬ *Coplanar A B C P*
shows ∃ *Q*. *A B C Orth D Q*
using *Orth-def assms(1) assms(2) l11-62-existence-bis l11-63-aux* **by** *fastforce*

lemma *l8-21-3*:
assumes *Coplanar A B C D* **and**
¬ *Coplanar A B C X*
shows
∃ *P T*. (*A B C Orth D P* ∧ *Coplanar A B C T* ∧ *Bet X T P*)
proof −
obtain *E* **where** *P1*: *E OrthAt A B C E X*
using *assms(2) l11-62-existence-bis* **by** *blast*
thus *?thesis*
proof *cases*
assume *P2*: *D = E*
obtain *Y* **where** *P3*: *Bet X D Y* ∧ *Cong D Y D X*
using *segment-construction* **by** *blast*
have *P4*: $D \neq X$
using *assms(1) assms(2)* **by** *auto*
have *P5*: *A B C Orth D X*
using *Orth-def P1 P2* **by** *auto*
have *P6*: $D \neq Y$
using *P3 P4 cong-reverse-identity* **by** *blast*
have *Col D X Y*
using *Col-def Col-perm P3* **by** *blast*
then have *A B C Orth D Y*
using *P5 P6 col-orth--orth* **by** *auto*
thus *?thesis*
using *P3 assms(1)* **by** *blast*
next
assume *K1*: $D \neq E$
then obtain *P'* **where** *P7*: *D E OS X P'* ∧ *A B C Orth D P'*
using *P1 assms(1) l11-63-aux* **by** *blast*

have *P8*: ¬ *Col A B C*
  using *assms*(*2*) *ncop--ncol* **by** *auto*
have *P9*: $E \neq X$
  using *P7 os-distincts* **by** *auto*
have *P10*: ∀ *P Q*. (*Coplanar A B C P* ∧ *Col E X Q*) ⟶ *Per P E Q*
  using *OrthAt-def P1* **by** *auto*
have *P11*: *D OrthAt A B C D P′*
  by (*simp add*: *P7 assms*(*1*) *col-cop-orth--orth-at col-trivial-3*)
have *P12*: $D \neq P'$
  using *P7 os-distincts* **by** *auto*
have *P13*: ¬ *Coplanar A B C P′*
  using *P11 orth-at--ncop* **by** *auto*
have *P14*: ∀ *P Q*. (*Coplanar A B C P* ∧ *Col D P′ Q*) ⟶ *Per P D Q*
  using *OrthAt-def P11* **by** *auto*
obtain *P* **where** *P15*: *Bet P′ D P* ∧ *Cong D P D P′*
  using *segment-construction* **by** *blast*
have *P16*: *D E TS X P*
proof −
  have *P16A*: *D E OS P′ X* **using** *P7 one-side-symmetry* **by** *blast*
  then have *D E TS P′ P*
  by (*metis P12 P15 Tarski-neutral-dimensionless.bet--ts Tarski-neutral-dimensionless-axioms cong-diff-3 one-side-not-col123*)
  thus *?thesis* **using** *l9-8-2 P16A* **by** *blast*
qed
obtain *T* **where** *P17*: *Col T D E* ∧ *Bet X T P*
  using *P16 TS-def* **by** *blast*
have *P18*: $D \neq P$
  using *P16 ts-distincts* **by** *blast*
have *Col D P′ P*
  using *Col-def Col-perm P15* **by** *blast*
then have *A B C Orth D P*
  using *P18 P7 col-orth--orth* **by** *blast*
thus *?thesis* **using** *col-cop2--cop*
  by (*meson P1 P17 Tarski-neutral-dimensionless.orth-at-chara Tarski-neutral-dimensionless-axioms K1 assms*(*1*) *col-permutation-1*)
qed
qed

lemma *mid2-orth-at2--cong*:
assumes *X OrthAt A B C X P* **and**
  Y OrthAt A B C Y Q **and**
  X Midpoint P P′ **and**
  Y Midpoint Q Q′
shows *Cong P Q P′ Q′*
proof −
have *Q1*: ¬ *Col A B C*
  using *assms*(*1*) *col--coplanar orth-at--ncop* **by** *blast*
have *Q2*: $X \neq P$
  using *assms*(*1*) *orth-at-distincts* **by** *auto*
have *Q3*: *Coplanar A B C X*
  using *OrthAt-def assms*(*1*) **by** *auto*
have *Q4*: ∀ *P0 Q*. (*Coplanar A B C P0* ∧ *Col X P Q*) ⟶ *Per P0 X Q*
  using *OrthAt-def assms*(*1*) **by** *blast*
have *Q5*: $Y \neq P$
  by (*metis assms*(*1*) *assms*(*2*) *orth-at--ncop2 orth-at-chara*)
have *Q6*: *Coplanar A B C Y*
  using *OrthAt-def assms*(*2*) **by** *blast*
have *Q7*: ∀ *P Q0*. (*Coplanar A B C P* ∧ *Col Y Q Q0*) ⟶ *Per P Y Q0*
  using *OrthAt-def assms*(*2*) **by** *blast*
obtain *Z* **where** *P1*: *Z Midpoint X Y*
  using *midpoint-existence* **by** *auto*
obtain *R* **where** *P2*: *Z Midpoint P R*
  using *symmetric-point-construction* **by** *auto*
obtain *R′* **where** *P3*: *Z Midpoint P′ R′*
  using *symmetric-point-construction* **by** *auto*
have *T1*: *Coplanar A B C Z*
  using *P1 Q3 Q6 bet-cop2--cop midpoint-bet* **by** *blast*

then have *Per Z X P*
using *Q4 assms*(*1*) *orth-at-chara* **by** *auto*
then have *T2*: *Cong Z P Z P'*
using *assms*(*3*) *per-double-cong* **by** *blast*
have *T3*: *Cong R Z R' Z*
by (*metis Cong-perm Midpoint-def P2 P3 T2 cong-transitivity*)
have *T4*: *Cong R Q R' Q'*
by (*meson P1 P2 P3 assms*(*3*) *assms*(*4*) *l7-13 not-cong-4321 symmetry-preserves-midpoint*)
have *Per Z Y Q*
using *Q7 T1 assms*(*2*) *orth-at-chara* **by** *auto*
then have *T5*: *Cong Z Q Z Q'*
using *assms*(*4*) *per-double-cong* **by** *auto*
have $R \neq Z$
by (*metis P2 P3 Q2 T3 assms*(*3*) *cong-diff-3 l7-17-bis midpoint-not-midpoint*)
thus *?thesis*
using *P2 P3 T2 T3 T4 T5 five-segment l7-2 midpoint-bet* **by** *blast*
qed

lemma *orth-at2-tsp--ts*:
assumes $P \neq Q$ **and**
P OrthAt A B C P X **and**
Q OrthAt A B C Q Y **and**
A B C TSP X Y
shows *P Q TS X Y*
proof −
obtain *T* **where** *P0*: *Coplanar A B C T* ∧ *Bet X T Y*
using *TSP-def assms*(*4*) **by** *auto*
have *P1*: ¬ *Col A B C*
using *assms*(*4*) *ncop--ncol tsp--ncop1* **by** *blast*
have *P2*: $P \neq X$
using *assms*(*2*) *orth-at-distincts* **by** *auto*
have *P3*: *Coplanar A B C P*
using *OrthAt-def assms*(*2*) **by** *blast*
have *P4*: ∀ *D*. *Coplanar A B C D* ⟶ *Per D P X*
using *assms*(*2*) *orth-at-chara* **by** *blast*
have *P5*: $Q \neq Y$
using *assms*(*3*) *orth-at-distincts* **by** *auto*
have *P6*: *Coplanar A B C Q*
using *OrthAt-def assms*(*3*) **by** *blast*
have *P7*: ∀ *D*. *Coplanar A B C D* ⟶ *Per D Q Y*
using *assms*(*3*) *orth-at-chara* **by** *blast*
have *P8*: ¬ *Col X P Q*
using *P3 P6 assms*(*1*) *assms*(*4*) *col-cop2--cop not-col-permutation-2 tsp--ncop1* **by** *blast*
have *P9*: ¬ *Col Y P Q*
using *P3 P6 assms*(*1*) *assms*(*4*) *col-cop2--cop not-col-permutation-2 tsp--ncop2* **by** *blast*
have *Col T P Q*
proof −
obtain *X'* **where** *Q1*: *P Midpoint X X'*
using *symmetric-point-construction* **by** *auto*
obtain *Y'* **where** *Q2*: *Q Midpoint Y Y'*
using *symmetric-point-construction* **by** *auto*
have *Per T P X*
using *P0 P4* **by** *auto*
then have *Q3*: *Cong T X T X'*
using *Q1 per-double-cong* **by** *auto*
have *Per T Q Y*
using *P0 P7* **by** *auto*
then have *Q4*: *Cong T Y T Y'* **using** *Q2 per-double-cong* **by** *auto*
have *Cong X Y X' Y'*
using *P1 Q1 Q2 assms*(*2*) *assms*(*3*) *mid2-orth-at2--cong* **by** *blast*
then have *X T Y Cong3 X' T Y'*
using *Cong3-def Q3 Q4 not-cong-2143* **by** *blast*
then have *Bet X' T Y'*
using *l4-6 P0* **by** *blast*
thus *?thesis*
using *Q1 Q2 Q3 Q4 Col-def P0 between-symmetry l7-22* **by** *blast*

qed
thus *?thesis*
using *P0 P8 P9 TS-def* **by** *blast*
qed

lemma *orth-dec*:
shows *A B C Orth U V* ∨ ¬ *A B C Orth U V* **by** *auto*

lemma *orth-at-dec*:
shows *X OrthAt A B C U V* ∨ ¬ *X OrthAt A B C U V* **by** *auto*

lemma *tsp-dec*:
shows *A B C TSP X Y* ∨ ¬ *A B C TSP X Y* **by** *auto*

lemma *osp-dec*:
shows *A B C OSP X Y* ∨ ¬ *A B C OSP X Y* **by** *auto*

lemma *ts2--inangle*:
assumes *A C TS B P* **and**
B P TS A C
shows *P InAngle A B C*
by (*metis InAngle-def assms*(*1*) *assms*(*2*) *bet-out ts2--ex-bet2 ts-distincts*)

lemma *os-ts--inangle*:
assumes *B P TS A C* **and**
B A OS C P
shows *P InAngle A B C*
proof −
have *P1*: ¬ *Col A B P*
using *TS-def assms*(*1*) **by** *auto*
have *P2*: ¬ *Col B A C*
using *assms*(*2*) *col123--nos* **by** *blast*
obtain *P′* **where** *P3*: *B Midpoint P P′*
using *symmetric-point-construction* **by** *blast*
then have *P4*: ¬ *Col B A P′*
by (*metis assms*(*2*) *col-one-side col-permutation-5 midpoint-col midpoint-distinct-2 one-side-not-col124*)
have *P5*: (*B* ≠ *P′* ∧ *B P TS A C* ∧ *Bet P B P′*) ⟶ (*P InAngle A B C* ∨ *P′ InAngle A B C*)
using *two-sides-in-angle* **by** *auto*
then have *P6*: *P InAngle A B C* ∨ *P′ InAngle A B C*
using *P3 P4 assms*(*1*) *midpoint-bet not-col-distincts* **by** *blast*
{
assume *P′ InAngle A B C*
then have *P7*: *A B OS P′ C*
using *Col-cases P2 P4 in-angle-one-side* **by** *blast*
then have *P8*: ¬ *A B TS P′ C*
using *l9-9* **by** *auto*
have *B A TS P P′*
using *P1 P3 P4 bet--ts midpoint-bet not-col-distincts not-col-permutation-4* **by** *auto*
then have *B A TS C P′*
using *P7 assms*(*2*) *invert-one-side l9-2 l9-8-2 l9-9* **by** *blast*
then have *B A TS P′ C*
using *l9-2* **by** *blast*
then have *A B TS P′ C*
by (*simp add*: *invert-two-sides*)
then have *P InAngle A B C*
using *P8* **by** *auto*
}
thus *?thesis*
using *P6* **by** *blast*
qed

lemma *os2--inangle*:
assumes *B A OS C P* **and**
B C OS A P
shows *P InAngle A B C*
using *assms*(*1*) *assms*(*2*) *col124--nos l9-9-bis os-ts--inangle two-sides-cases* **by** *blast*

lemma *acute-conga--acute*:
  assumes *Acute A B C* **and**
    A B C CongA D E F
  shows *Acute D E F*
proof −
  have *D E F LeA A B C*
    by (*simp add*: *assms*(*2*) *conga--lea456123*)
  thus *?thesis*
    using *acute-lea-acute assms*(*1*) **by** *blast*
qed

lemma *acute-out2--acute*:
  assumes *B Out A′ A* **and**
    B Out C′ C **and**
    Acute A B C
  shows *Acute A′ B C′*
  by (*meson Tarski-neutral-dimensionless.out2--conga Tarski-neutral-dimensionless-axioms acute-conga--acute assms*(*1*) *assms*(*2*) *assms*(*3*))

lemma *conga-obtuse--obtuse*:
  assumes *Obtuse A B C* **and**
    A B C CongA D E F
  shows *Obtuse D E F*
  using *assms*(*1*) *assms*(*2*) *conga--lea lea-obtuse-obtuse* **by** *blast*

lemma *obtuse-out2--obtuse*:
  assumes *B Out A′ A* **and**
    B Out C′ C **and**
    Obtuse A B C
  shows *Obtuse A′ B C′*
  by (*meson Tarski-neutral-dimensionless.out2--conga Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*2*) *assms*(*3*) *conga-obtuse--obtuse*)

lemma *bet-lea--bet*:
  assumes *Bet A B C* **and**
    A B C LeA D E F
  shows *Bet D E F*
proof −
  have *A B C CongA D E F*
    by (*metis assms*(*1*) *assms*(*2*) *l11-31-2 lea-asym lea-distincts*)
  thus *?thesis*
    using *assms*(*1*) *bet-conga--bet* **by** *blast*
qed

lemma *out-lea--out*:
  assumes *E Out D F* **and**
    A B C LeA D E F
  shows *B Out A C*
proof −
  have *D E F CongA A B C*
    using *Tarski-neutral-dimensionless.l11-31-1 Tarski-neutral-dimensionless.lea-asym Tarski-neutral-dimensionless.lea-distincts Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*2*) **by** *fastforce*
  thus *?thesis*
    using *assms*(*1*) *out-conga-out* **by** *blast*
qed

lemma *bet2-lta--lta*:
  assumes *A B C LtA D E F* **and**
    Bet A B A′ **and**
    A′ ≠ *B* **and**
    Bet D E D′ **and**
    D′ ≠ *E*
  shows *D′ E F LtA A′ B C*
proof −
  have *P1*: *D′ E F LeA A′ B C*

by (*metis Bet-cases assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *l11-36-aux2 lea-distincts lta--lea*)
have ¬ *D′ E F CongA A′ B C*
by (*metis assms*(*1*) *assms*(*2*) *assms*(*4*) *between-symmetry conga-sym l11-13 lta-distincts not-lta-and-conga*)
thus *?thesis*
by (*simp add*: *LtA-def P1*)
qed

lemma *lea123456-lta--lta*:
assumes *A B C LeA D E F* **and**
D E F LtA G H I
shows *A B C LtA G H I*
proof −
have ¬ *G H I LeA F E D*
by (*metis* (*no-types*) *Tarski-neutral-dimensionless.lea--nlta Tarski-neutral-dimensionless.lta-left-comm Tarski-neutral-dimensionless-c*
assms(*2*))
then have ¬ *A B C CongA G H I*
by (*metis Tarski-neutral-dimensionless.lta-distincts Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*2*) *conga-pseudo-refl*
l11-30)
thus *?thesis*
by (*meson LtA-def Tarski-neutral-dimensionless.lea-trans Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*2*))
qed

lemma *lea456789-lta--lta*:
assumes *A B C LtA D E F* **and**
D E F LeA G H I
shows *A B C LtA G H I*
by (*meson LtA-def assms*(*1*) *assms*(*2*) *conga--lea456123 lea-trans lta--nlea*)

lemma *acute-per--lta*:
assumes *Acute A B C* **and**
$D \neq E$ **and**
$E \neq F$ **and**
Per D E F
shows *A B C LtA D E F*
proof −
obtain *G H I* **where** *P1*: *Per G H I* ∧ *A B C LtA G H I*
using *Acute-def assms*(*1*) **by** *auto*
then have *G H I CongA D E F*
using *assms*(*2*) *assms*(*3*) *assms*(*4*) *l11-16 lta-distincts* **by** *blast*
thus *?thesis*
by (*metis P1 conga-preserves-lta conga-refl lta-distincts*)
qed

lemma *obtuse-per--lta*:
assumes *Obtuse A B C* **and**
$D \neq E$ **and**
$E \neq F$ **and**
Per D E F
shows *D E F LtA A B C*
proof −
obtain *G H I* **where** *P1*: *Per G H I* ∧ *G H I LtA A B C*
using *Obtuse-def assms*(*1*) **by** *auto*
then have *G H I CongA D E F*
using *assms*(*2*) *assms*(*3*) *assms*(*4*) *l11-16 lta-distincts* **by** *blast*
thus *?thesis*
by (*metis P1 Tarski-neutral-dimensionless.l11-51 Tarski-neutral-dimensionless-axioms assms*(*1*) *cong-reflexivity*
conga-preserves-lta obtuse-distincts)
qed

lemma *acute-obtuse--lta*:
assumes *Acute A B C* **and**
Obtuse D E F
shows *A B C LtA D E F*
by (*metis Acute-def assms*(*1*) *assms*(*2*) *lta-distincts lta-trans obtuse-per--lta*)

lemma *lea-in-angle*:

assumes *A B P LeA A B C* **and**
A B OS C P
shows *P InAngle A B C*
proof –
obtain *T* **where** *P3*: *T InAngle A B C* ∧ *A B P CongA A B T*
using *LeA-def assms*(*1*) **by** *blast*
thus *?thesis*
by (*metis assms*(*2*) *conga-preserves-in-angle conga-refl not-conga-sym one-side-symmetry os-distincts*)
qed

lemma *acute-bet--obtuse*:
assumes *Bet A B A′* **and**
A′ ≠ *B* **and**
Acute A B C
shows *Obtuse A′ B C*
proof *cases*
assume *P1*: *Col A B C*
show *?thesis*
proof *cases*
assume *Bet A B C*
thus *?thesis*
using *P1 acute-col--out assms*(*3*) *not-bet-and-out* **by** *blast*
next
assume ¬ *Bet A B C*
thus *?thesis*
by (*smt P1 assms*(*1*) *assms*(*2*) *bet--obtuse between-inner-transitivity between-symmetry outer-transitivity-between third-point*)
qed
next
assume *P2*: ¬ *Col A B C*
then obtain *D* **where** *P3*: *A B Perp D B* ∧ *A B OS C D*
using *col-trivial-2 l10-15* **by** *blast*
{
assume *P4*: *Col C B D*
then have *Per A B C*
proof –
have *P5*: *B* ≠ *D*
using *P3 perp-not-eq-2* **by** *auto*
have *Per A B D*
using *P3 Perp-perm perp-per-2* **by** *blast*
thus *?thesis*
using *P4 P5 not-col-permutation-2 per-col* **by** *blast*
qed
then have *A B C LtA A B C*
by (*metis Acute-def acute-per--lta assms*(*3*) *lta-distincts*)
then have *False*
by (*simp add*: *nlta*)
}
then have *P6*: ¬ *Col C B D* **by** *auto*
have *P7*: *B A′ OS C D*
by (*metis P3 assms*(*1*) *assms*(*2*) *bet-col col2-os--os l5-3*)
have *T1*: *Per A B D*
by (*simp add*: *P3 perp-left-comm perp-per-1*)
have *Q1*: *B C TS A′ A*
using *P2 assms*(*1*) *assms*(*2*) *bet--ts l9-2 not-col-permutation-1* **by** *auto*
have *A B C LtA A B D*
using *P2 P6 T1 acute-per--lta assms*(*3*) *not-col-distincts* **by** *auto*
then have *A B C LeA A B D*
by (*simp add*: *lta--lea*)
then have *C InAngle A B D*
by (*simp add*: *P3 lea-in-angle one-side-symmetry*)
then have *C InAngle D B A*
using *l11-24* **by** *blast*
then have *C B TS D A*
by (*simp add*: *P2 P6 in-angle-two-sides not-col-permutation-1 not-col-permutation-4*)
then have *B C TS D A*

```
    using invert-two-sides by blast
  then have B C OS A′ D
    using Q1 l9-8-1 by auto
  then have T1A: D InAngle A′ B C
    by (simp add: P7 os2--inangle)
  then have A B D CongA A′ B D
   by (metis Per-cases T1 Tarski-neutral-dimensionless.conga-comm Tarski-neutral-dimensionless.l11-18-1 Tarski-neutral-dimensionles
acute-distincts assms(1) assms(3) inangle-distincts)
  then have T2: A B D LeA A′ B C
    using LeA-def T1A by auto
  {
    assume A B D CongA A′ B C
    then have False
      by (metis OS-def P7 T1 TS-def Tarski-neutral-dimensionless.out2--conga Tarski-neutral-dimensionless-axioms ‹A
B C LtA A B D› ‹A B D CongA A′ B D› ‹⋀thesis. (⋀D. A B Perp D B ∧ A B OS C D ⟹ thesis) ⟹ thesis›
col-trivial-2 invert-one-side l11-17 l11-19 not-lta-and-conga out-trivial)
  }
  then have ¬ A B D CongA A′ B C by auto
  then have A B D LtA A′ B C
    using T2 LtA-def by auto
  thus ?thesis
    using T1 Obtuse-def by blast
qed

lemma bet-obtuse--acute:
  assumes Bet A B A′ and
    A′ ≠ B and
    Obtuse A B C
  shows Acute A′ B C
proof cases
  assume P1: Col A B C
  thus ?thesis
  proof cases
    assume Bet A B C
    then have B Out A′ C
      by (smt Out-def assms(1) assms(2) assms(3) l5-2 obtuse-distincts)
    thus ?thesis
      by (simp add: out--acute)
  next
    assume ¬ Bet A B C
    thus ?thesis
      using P1 assms(3) col-obtuse--bet by blast
  qed
next
  assume P2: ¬ Col A B C
  then obtain D where P3: A B Perp D B ∧ A B OS C D
    using col-trivial-2 l10-15 by blast
  {
    assume P3A: Col C B D
    have P3B: B ≠ D
      using P3 perp-not-eq-2 by blast
    have P3C: Per A B D
      using P3 Perp-perm perp-per-2 by blast
    then have Per A B C
      using P3A P3B not-col-permutation-2 per-col by blast
    then have A B C LtA A B C
      using P2 assms(3) not-col-distincts obtuse-per--lta by auto
    then have False
      by (simp add: nlta)
  }
  then have P4: ¬ Col C B D by auto
  have Col B A A′
    using Col-def Col-perm assms(1) by blast
  then have P5: B A′ OS C D
    using P3 assms(2) invert-one-side col2-os--os col-trivial-3 by blast
  have P7: Per A′ B D
```

by (*meson Col-def P3 Tarski-neutral-dimensionless.Per-perm Tarski-neutral-dimensionless-axioms assms*(*1*) *col-trivial-2 l8-16-1*)
  have *A′ B C LtA A′ B D*
  proof −
    have *P8*: *A′ B C LeA A′ B D*
    proof −
      have *P9*: *C InAngle A′ B D*
      proof −
        have *P10*: *B A′ OS D C*
          by (*simp add*: *P5 one-side-symmetry*)
        have *B D OS A′ C*
        proof −
          have *P6*: ¬ *Col A B D*
            using *P3 col124--nos* **by** *auto*
          then have *P11*: *B D TS A′ A*
            using *Col-perm P5 assms*(*1*) *bet--ts l9-2 os-distincts* **by** *blast*
          have *A B D LtA A B C*
          proof −
            have *P11A*: *A* ≠ *B*
              using *P2 col-trivial-1* **by** *auto*
            have *P11B*: *B* ≠ *D*
              using *P4 col-trivial-2* **by** *blast*
            have *Per A B D*
              using *P3 Perp-cases perp-per-2* **by** *blast*
            thus *?thesis*
              by (*simp add*: *P11A P11B Tarski-neutral-dimensionless.obtuse-per--lta Tarski-neutral-dimensionless-axioms assms*(*3*))
          qed
          then have *A B D LeA A B C*
            by (*simp add*: *lta--lea*)
          then have *D InAngle A B C*
            by (*simp add*: *P3 lea-in-angle*)
          then have *D InAngle C B A*
            using *l11-24* **by** *blast*
          then have *D B TS C A*
            by (*simp add*: *P4 P6 in-angle-two-sides not-col-permutation-4*)
          then have *B D TS C A*
            by (*simp add*: *invert-two-sides*)
          thus *?thesis*
            using *OS-def P11* **by** *blast*
        qed
        thus *?thesis*
          by (*simp add*: *P10 os2--inangle*)
      qed
      have *A′ B C CongA A′ B C*
        using *assms*(*2*) *assms*(*3*) *conga-refl obtuse-distincts* **by** *blast*
      thus *?thesis*
        by (*simp add*: *P9 inangle--lea*)
    qed
    {
      assume *A′ B C CongA A′ B D*
      then have *B Out C D*
        using *P5 conga-os--out invert-one-side* **by** *blast*
      then have *False*
        using *P4 not-col-permutation-4 out-col* **by** *blast*
    }
    then have ¬ *A′ B C CongA A′ B D* **by** *auto*
    thus *?thesis*
      by (*simp add*: *LtA-def P8*)
  qed
  thus *?thesis*
    using *Acute-def P7* **by** *blast*
qed

lemma *inangle-dec*:
  P InAngle A B C ∨ ¬ *P InAngle A B C* **by** *blast*

```
lemma lea-dec:
  A B C LeA D E F ∨ ¬ A B C LeA D E F by blast

lemma lta-dec:
  A B C LtA D E F ∨ ¬ A B C LtA D E F by blast

lemma lea-total:
  assumes A ≠ B and
    B ≠ C and
    D ≠ E and
    E ≠ F
  shows A B C LeA D E F ∨ D E F LeA A B C
proof cases
  assume P1: Col A B C
  show ?thesis
  proof cases
    assume B Out A C
    thus ?thesis
      using assms(3) assms(4) l11-31-1 by auto
  next
    assume ¬ B Out A C
    thus ?thesis
      by (metis P1 assms(1) assms(2) assms(3) assms(4) l11-31-2 or-bet-out)
  qed
next
  assume P2: ¬ Col A B C
  show ?thesis
  proof cases
    assume P3: Col D E F
    show ?thesis
    proof cases
      assume E Out D F
      thus ?thesis
        using assms(1) assms(2) l11-31-1 by auto
    next
      assume ¬ E Out D F
      thus ?thesis
        by (metis P3 assms(1) assms(2) assms(3) assms(4) l11-31-2 l6-4-2)
    qed
  next
    assume P4: ¬ Col D E F
    show ?thesis
    proof cases
      assume A B C LeA D E F
      thus ?thesis
        by simp
    next
      assume P5: ¬ A B C LeA D E F
      obtain P where P6: D E F CongA A B P ∧ A B OS P C
        using P2 P4 angle-construction-1 by blast
      then have P7: B A OS C P
        using invert-one-side one-side-symmetry by blast
      have B C OS A P
      proof -
        {
          assume Col P B C
          then have P7B: B Out C P
            using Col-cases P7 col-one-side-out by blast
          have A B C CongA D E F
          proof -
            have P7C: A B P CongA D E F
              by (simp add: P6 conga-sym)
            have P7D: B Out A A
              by (simp add: assms(1) out-trivial)
            have P7E: E Out D D
```

```
            by (simp add: assms(3) out-trivial)
          have E Out F F
            using assms(4) out-trivial by auto
          thus ?thesis
            using P7B P7C P7D P7E l11-10 by blast
        qed
        then have A B C LeA D E F
          by (simp add: conga--lea)
        then have False
          by (simp add: P5)
      }
      then have P8: ¬ Col P B C by auto
      {
        assume T0: B C TS A P
        have A B C CongA D E F
        proof −
          have T1: A B C LeA A B P
          proof −
            have T1A: C InAngle A B P
              by (simp add: P7 T0 one-side-symmetry os-ts--inangle)
            have A B C CongA A B C
              using assms(1) assms(2) conga-refl by auto
            thus ?thesis
              by (simp add: T1A inangle--lea)
          qed
          have T2: A B C CongA A B C
            using assms(1) assms(2) conga-refl by auto
          have A B P CongA D E F
            by (simp add: P6 conga-sym)
          thus ?thesis
            using P5 T1 T2 l11-30 by blast
        qed
        then have A B C LeA D E F
          by (simp add: conga--lea)
        then have False
          by (simp add: P5)
      }
      then have ¬ B C TS A P by auto
      thus ?thesis
        using Col-perm P7 P8 one-side-symmetry os-ts1324--os two-sides-cases by blast
    qed
    then have P InAngle A B C
      using P7 os2--inangle by blast
    then have D E F LeA A B C
      using P6 LeA-def by blast
    thus ?thesis
      by simp
  qed
 qed
qed

lemma or-lta2-conga:
 assumes A ≠ B and
   C ≠ B and
   D ≠ E and
   F ≠ E
 shows A B C LtA D E F ∨ D E F LtA A B C ∨ A B C CongA D E F
proof −
 have P1: A B C LeA D E F ∨ D E F LeA A B C
   using assms(1) assms(2) assms(3) assms(4) lea-total by auto
 {
   assume A B C LeA D E F
   then have A B C LtA D E F ∨ D E F LtA A B C ∨ A B C CongA D E F
     using LtA-def by blast
 }
 {
```

assume *D E F LeA A B C*
then have *A B C LtA D E F* ∨ *D E F LtA A B C* ∨ *A B C CongA D E F*
using *LtA-def conga-sym* **by** *blast*
}
thus *?thesis*
using *P1* ‹*A B C LeA D E F* ⟹ *A B C LtA D E F* ∨ *D E F LtA A B C* ∨ *A B C CongA D E F*› **by** *blast*
qed

lemma *angle-partition*:
assumes *A* ≠ *B* **and**
B ≠ *C*
shows *Acute A B C* ∨ *Per A B C* ∨ *Obtuse A B C*
proof −
obtain *A′ B′ D′* **where** *P1*: ¬ (*Bet A′ B′ D′* ∨ *Bet B′ D′ A′* ∨ *Bet D′ A′ B′*)
using *lower-dim* **by** *auto*
then have ¬ *Col A′ B′ D′*
by (*simp add*: *Col-def*)
obtain *C′* **where** *P3*: *A′ B′ Perp C′ B′*
by (*metis P1 Perp-perm between-trivial2 perp-exists*)
then have *P4*: *A B C LtA A′ B′ C′* ∨ *A′ B′ C′ LtA A B C* ∨ *A B C CongA A′ B′ C′*
by (*metis P1 assms*(*1*) *assms*(*2*) *between-trivial2 or-lta2-conga perp-not-eq-2*)
{
assume *A B C LtA A′ B′ C′*
then have *Acute A B C* ∨ *Per A B C* ∨ *Obtuse A B C*
using *Acute-def P3 Perp-cases perp-per-2* **by** *blast*
}
{
assume *A′ B′ C′ LtA A B C*
then have *Acute A B C* ∨ *Per A B C* ∨ *Obtuse A B C*
using *Obtuse-def P3 Perp-cases perp-per-2* **by** *blast*
}
{
assume *A B C CongA A′ B′ C′*
then have *Acute A B C* ∨ *Per A B C* ∨ *Obtuse A B C*
by (*metis P3 Perp-cases Tarski-neutral-dimensionless.conga-sym Tarski-neutral-dimensionless.l11-17 Tarski-neutral-dimensionless-*
perp-per-2)
}
thus *?thesis*
using *P4* ‹*A B C LtA A′ B′ C′* ⟹ *Acute A B C* ∨ *Per A B C* ∨ *Obtuse A B C*› ‹*A′ B′ C′ LtA A B C* ⟹ *Acute A B C* ∨ *Per A B C* ∨ *Obtuse A B C*› **by** *auto*
qed

lemma *acute-chara-1*:
assumes *Bet A B A′* **and**
B ≠ *A′* **and**
Acute A B C
shows *A B C LtA A′ B C*
proof −
have *Obtuse A′ B C*
using *acute-bet--obtuse assms*(*1*) *assms*(*2*) *assms*(*3*) **by** *blast*
thus *?thesis*
by (*simp add*: *acute-obtuse--lta assms*(*3*))
qed

lemma *acute-chara-2*:
assumes *Bet A B A′* **and**
A B C LtA A′ B C
shows *Acute A B C*
by (*metis Tarski-neutral-dimensionless.Acute-def Tarski-neutral-dimensionless-axioms acute-chara-1 angle-partition assms*(*1*) *assms*(*2*) *bet-obtuse--acute between-symmetry lta-distincts not-and-lta*)

lemma *acute-chara*:
assumes *Bet A B A′* **and**
B ≠ *A′*
shows *Acute A B C* ⟷ *A B C LtA A′ B C*
using *acute-chara-1 acute-chara-2 assms*(*1*) *assms*(*2*) **by** *blast*

lemma *obtuse-chara*:
  assumes *Bet A B A′* **and**
    B ≠ A′
  shows *Obtuse A B C ⟷ A′ B C LtA A B C*
  by (*metis Tarski-neutral-dimensionless.Obtuse-def Tarski-neutral-dimensionless-axioms acute-bet--obtuse acute-chara assms*(*1*) *assms*(*2*) *bet-obtuse--acute between-symmetry lta-distincts*)

lemma *conga--acute*:
  assumes *A B C CongA A C B*
  shows *Acute A B C*
  by (*metis acute-sym angle-partition assms conga-distinct conga-obtuse--obtuse l11-17 l11-43*)

lemma *cong--acute*:
  assumes *A ≠ B* **and**
    B ≠ C **and**
    Cong A B A C
  shows *Acute A B C*
  using *angle-partition assms*(*1*) *assms*(*2*) *assms*(*3*) *cong--nlt l11-46 lt-left-comm* **by** *blast*

lemma *nlta--lea*:
  assumes ¬ *A B C LtA D E F* **and**
    A ≠ B **and**
    B ≠ C **and**
    D ≠ E **and**
    E ≠ F
  shows *D E F LeA A B C*
  by (*metis LtA-def assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *conga--lea456123 or-lta2-conga*)

lemma *nlea--lta*:
  assumes ¬ *A B C LeA D E F* **and**
    A ≠ B **and**
    B ≠ C **and**
    D ≠ E **and**
    E ≠ F
  shows *D E F LtA A B C*
  using *assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *nlta--lea* **by** *blast*

lemma *triangle-strict-inequality*:
  assumes *Bet A B D* **and**
    Cong B C B D **and**
    ¬ *Bet A B C*
  shows *A C Lt A D*
proof *cases*
  assume *P1*: *Col A B C*
  then have *P2*: *B Out A C*
    using *assms*(*3*) *not-out-bet* **by** *auto*
  {
    assume *Bet B A C*
    then have *P3*: *A C Le A D*
      by (*meson assms*(*1*) *assms*(*2*) *cong--le l5-12-a le-transitivity*)
    have ¬ *Cong A C A D*
      by (*metis Out-def P1 P2 assms*(*1*) *assms*(*2*) *assms*(*3*) *l4-18*)
    then have *A C Lt A D*
      by (*simp add*: *Lt-def P3*)
  }
  {
    assume *Bet A C B*
    then have *P5*: *Bet A C D*
      using *assms*(*1*) *between-exchange4* **by** *blast*
    then have *P6*: *A C Le A D*
      by (*simp add*: *bet--le1213*)
    have ¬ *Cong A C A D*
      using *P5 assms*(*1*) *assms*(*3*) *between-cong* **by** *blast*
    then have *A C Lt A D*
      by (*simp add*: *Lt-def P6*)

}
thus *?thesis*
using *P1* ‹*Bet B A C* $\Longrightarrow$ *A C Lt A D*› *assms*(*3*) *between-symmetry third-point* **by** *blast*
next
assume *T1*: ¬ *Col A B C*
have *T2*: $A \neq D$
using *T1 assms*(*1*) *between-identity col-trivial-1* **by** *auto*
have *T3*: ¬ *Col A C D*
by (*metis Col-def T1 T2 assms*(*1*) *col-transitivity-2*)
have *T4*: *Bet D B A*
using *Bet-perm assms*(*1*) **by** *blast*
have *T5*: *C D A CongA D C B*
proof −
have *T6*: *C D B CongA D C B*
by (*metis assms*(*1*) *assms*(*2*) *assms*(*3*) *between-trivial conga-comm l11-44-1-a not-conga-sym*)
have *T7*: *D Out C C*
using *T3 not-col-distincts out-trivial* **by** *blast*
have *T8*: *D Out A B*
by (*metis assms*(*1*) *assms*(*2*) *assms*(*3*) *bet-out-1 cong-diff l6-6*)
have *T9*: *C Out D D*
using *T7 out-trivial* **by** *force*
have *C Out B B*
using *T1 not-col-distincts out-trivial* **by** *auto*
thus *?thesis*
using *T6 T7 T8 T9 l11-10* **by** *blast*
qed
have *A D C LtA A C D*
proof −
have *B InAngle D C A*
by (*metis InAngle-def T1 T3 T4 not-col-distincts out-trivial*)
then have *C D A LeA D C A*
using *T5 LeA-def* **by** *auto*
then have *T10*: *A D C LeA A C D*
by (*simp add*: *lea-comm*)
have ¬ *A D C CongA A C D*
by (*metis Col-perm T3 assms*(*1*) *assms*(*2*) *assms*(*3*) *bet-col l11-44-1-b l4-18 not-bet-distincts not-cong-3412*)
thus *?thesis*
using *LtA-def T10* **by** *blast*
qed
thus *?thesis*
by (*simp add*: *l11-44-2-b*)
qed

lemma *triangle-inequality*:
assumes *Bet A B D* **and**
Cong B C B D
shows *A C Le A D*
proof *cases*
assume *Bet A B C*
thus *?thesis*
by (*metis assms*(*1*) *assms*(*2*) *between-cong-3 cong--le le-reflexivity*)
next
assume ¬ *Bet A B C*
then have *A C Lt A D*
using *assms*(*1*) *assms*(*2*) *triangle-strict-inequality* **by** *auto*
thus *?thesis*
by (*simp add*: *Lt-def*)
qed

lemma *triangle-strict-inequality-2*:
assumes *Bet A′ B′ C′* **and**
Cong A B A′ B′ **and**
Cong B C B′ C′ **and**
¬ *Bet A B C*
shows *A C Lt A′ C′*
proof −

obtain *D* **where** *P1*: *Bet A B D* ∧ *Cong B D B C*
using *segment-construction* **by** *blast*
then have *P2*: *A C Lt A D*
using *P1 assms*(*4*) *cong-symmetry triangle-strict-inequality* **by** *blast*
have *Cong A D A′ C′*
using *P1 assms*(*1*) *assms*(*2*) *assms*(*3*) *cong-transitivity l2-11-b* **by** *blast*
thus *?thesis*
using *P2 cong2-lt--lt cong-reflexivity* **by** *blast*
qed

lemma *triangle-inequality-2*:
assumes *Bet A′ B′ C′* **and**
Cong A B A′ B′ **and**
Cong B C B′ C′
shows *A C Le A′ C′*
proof −
obtain *D* **where** *P1*: *Bet A B D* ∧ *Cong B D B C*
using *segment-construction* **by** *blast*
have *P2*: *A C Le A D*
by (*meson P1 Tarski-neutral-dimensionless.triangle-inequality Tarski-neutral-dimensionless-axioms not-cong-3412*)
have *Cong A D A′ C′*
using *P1 assms*(*1*) *assms*(*2*) *assms*(*3*) *cong-transitivity l2-11-b* **by** *blast*
thus *?thesis*
using *P2 cong--le le-transitivity* **by** *blast*
qed

lemma *triangle-strict-reverse-inequality*:
assumes *A Out B D* **and**
Cong A C A D **and**
¬ *A Out B C*
shows *B D Lt B C*
proof *cases*
assume *Col A B C*
thus *?thesis*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *col-permutation-4 cong-symmetry not-bet-and-out or-bet-out triangle-strict-inequality* **by** *blast*
next
assume *P1*: ¬ *Col A B C*
show *?thesis*
proof *cases*
assume *B* = *D*
thus *?thesis*
using *P1 lt1123 not-col-distincts* **by** *auto*
next
assume *P2*: *B* ≠ *D*
then have *P3*: ¬ *Col B C D*
using *P1 assms*(*1*) *col-trivial-2 colx not-col-permutation-5 out-col* **by** *blast*
have *P4*: ¬ *Col A C D*
using *P1 assms*(*1*) *col2--eq col-permutation-4 out-col out-distinct* **by** *blast*
have *P5*: *C* ≠ *D*
using *assms*(*1*) *assms*(*3*) **by** *auto*
then have *P6*: *A C D CongA A D C*
by (*metis P1 assms*(*2*) *col-trivial-3 l11-44-1-a*)
{
assume *T1*: *Bet A B D*
then have *T2*: *Bet D B A*
using *Bet-perm* **by** *blast*
have *B C D LtA B D C*
proof −
have *T3*: *D C B CongA B C D*
by (*metis P3 conga-pseudo-refl not-col-distincts*)
have *T3A*: *D Out B A*
by (*simp add*: *P2 T1 bet-out-1*)
have *T3B*: *C Out D D*
using *P5 out-trivial* **by** *auto*
have *T3C*: *C Out A A*

using *P1 not-col-distincts out-trivial* **by** *blast*
have *D Out C C*
by (*simp add*: *P5 out-trivial*)
then have *T4*: *D C A CongA B D C* **using** *T3A T3B T3C*
by (*meson Tarski-neutral-dimensionless.conga-comm Tarski-neutral-dimensionless.conga-right-comm Tarski-neutral-dimensionles*
Tarski-neutral-dimensionless-axioms P6)
have *D C B LtA D C A*
proof −
have *T4A*: *D C B LeA D C A*
proof −
have *T4AA*: *B InAngle D C A*
using *InAngle-def P1 P5 T2 not-col-distincts out-trivial* **by** *auto*
have *D C B CongA D C B*
using *T3 conga-right-comm* **by** *blast*
thus *?thesis*
by (*simp add*: *T4AA inangle--lea*)
qed
{
assume *T5*: *D C B CongA D C A*
have *D C OS B A*
using *Col-perm P3 T3A out-one-side* **by** *blast*
then have *C Out B A*
using *T5 conga-os--out* **by** *blast*
then have *False*
using *Col-cases P1 out-col* **by** *blast*
}
then have ¬ *D C B CongA D C A* **by** *auto*
thus *?thesis*
using *LtA-def T4A* **by** *blast*
qed
thus *?thesis*
using *T3 T4 conga-preserves-lta* **by** *auto*
qed
}
{
assume *Q1*: *Bet B D A*
obtain *E* **where** *Q2*: *Bet B C E* ∧ *Cong B C C E*
using *Cong-perm segment-construction* **by** *blast*
have *A D C LtA E C D*
proof −
have *Q3*: *D C OS A E*
proof −
have *Q4*: *D C TS A B*
by (*metis Col-perm P2 Q1 P4 bet--ts between-symmetry*)
then have *D C TS E B*
by (*metis Col-def Q1 Q2 TS-def bet--ts cong-identity invert-two-sides l9-2*)
thus *?thesis*
using *OS-def Q4* **by** *blast*
qed
have *Q5*: *A C D LtA E C D*
proof −
have *D C A LeA D C E*
proof −
have *B Out D A*
using *P2 Q1 bet-out* **by** *auto*
then have *B C OS D A*
by (*simp add*: *P3 out-one-side*)
then have *C B OS D A*
using *invert-one-side* **by** *blast*
then have *C E OS D A*
by (*metis Col-def Q2 Q3 col124--nos col-one-side diff-col-ex not-col-permutation-5*)
then have *Q5A*: *A InAngle D C E*
by (*simp add*: ‹*C E OS D A*› *Q3 invert-one-side one-side-symmetry os2--inangle*)
have *D C A CongA D C A*
using *CongA-def P6 conga-refl* **by** *auto*
thus *?thesis*

by (*simp add*: *Q5A inangle--lea*)
qed
then have *Q6*: *A C D LeA E C D*
using *lea-comm* **by** *blast*
{
assume *A C D CongA E C D*
then have *D C A CongA D C E*
by (*simp add*: *conga-comm*)
then have *C Out A E*
using *Q3 conga-os--out* **by** *auto*
then have *False*
by (*meson Col-def Out-cases P1 Q2 not-col-permutation-3 one-side-not-col123 out-one-side*)
}
then have ¬ *A C D CongA E C D* **by** *auto*
thus *?thesis*
by (*simp add*: *LtA-def Q6*)
qed
have *E C D CongA E C D*
by (*metis P1 P5 Q2 cong-diff conga-refl not-col-distincts*)
thus *?thesis*
using *Q5 P6 conga-preserves-lta* **by** *auto*
qed
then have *B C D LtA B D C*
using *Bet-cases P1 P2 Q1 Q2 bet2-lta--lta not-col-distincts* **by** *blast*
}
then have *B C D LtA B D C*
by (*meson Out-def* ‹*Bet A B D* ⟹ *B C D LtA B D C*› *assms*(*1*) *between-symmetry*)
thus *?thesis*
by (*simp add*: *l11-44-2-b*)
qed
qed

lemma *triangle-reverse-inequality*:
assumes *A Out B D* **and**
Cong A C A D
shows *B D Le B C*
proof *cases*
assume *A Out B C*
thus *?thesis*
by (*metis assms*(*1*) *assms*(*2*) *bet--le1213 cong-pseudo-reflexivity l6-11-uniqueness l6-6 not-bet-distincts not-cong-4312*)
next
assume ¬ *A Out B C*
thus *?thesis*
using *triangle-strict-reverse-inequality assms*(*1*) *assms*(*2*) *lt--le* **by** *auto*
qed

lemma *os3--lta*:
assumes *A B OS C D* **and**
B C OS A D **and**
A C OS B D
shows *B A C LtA B D C*
proof −
have *P1*: *D InAngle A B C*
by (*simp add*: *assms*(*1*) *assms*(*2*) *invert-one-side os2--inangle*)
then obtain *E* **where** *P2*: *Bet A E C* ∧ (*E* = *B* ∨ *B Out E D*)
using *InAngle-def* **by** *auto*
have *P3*: ¬ *Col A B C*
using *assms*(*1*) *one-side-not-col123* **by** *auto*
have *P4*: ¬ *Col A C D*
using *assms*(*3*) *col124--nos* **by** *auto*
have *P5*: ¬ *Col B C D*
using *assms*(*2*) *one-side-not-col124* **by** *auto*
have *P6*: ¬ *Col A B D*
using *assms*(*1*) *one-side-not-col124* **by** *auto*
{
assume *E* = *B*

```
  then have B A C LtA B D C
    using P2 P3 bet-col by blast
}
{
  assume P7: B Out E D
  have P8: A ≠ E
    using P6 P7 not-col-permutation-4 out-col by blast
  have P9: C ≠ E
    using P5 P7 out-col by blast
  have P10: B A C LtA B E C
  proof −
    have P10A: ¬ Col E A B
      by (metis Col-def P2 P3 P8 col-transitivity-1)
    then have P10B: E B A LtA B E C
      using P2 P9 Tarski-neutral-dimensionless.l11-41-aux Tarski-neutral-dimensionless-axioms by fastforce
    have P10C: E A B LtA B E C
      using P2 P9 P10A l11-41 by auto
    have P11: E A B CongA B A C
    proof −
      have P11A: A Out B B
        using assms(2) os-distincts out-trivial by auto
      have A Out C E
        using P2 P8 bet-out l6-6 by auto
      thus ?thesis
        using P11A conga-right-comm out2--conga by blast
    qed
    have P12: B E C CongA B E C
      by (metis Col-def P2 P3 P9 conga-refl)
    thus ?thesis
      using P11 P10C conga-preserves-lta by auto
  qed
  have B E C LtA B D C
  proof −
    have U1: E Out D B
    proof −
      obtain pp :: 'p ⇒ 'p ⇒ 'p where
        f1: ∀ p pa. p ≠ (pp p pa) ∧ pa ≠ (pp p pa) ∧ Col p pa (pp p pa)
        using diff-col-ex by moura
      then have ∀ p pa pb. Col pb pa p ∨ ¬ Col pb pa (pp p pa)
        by (meson l6-16-1)
      then have f2: ∀ p pa. Col pa p pa
        using f1 by metis
      have f3: (E = B ∨ D = E) ∨ Col D E B
        using f1 by (metis Col-def P2 col-out2-col l6-16-1 out-trivial)
      have ∀ p. (A = E ∨ Col p A C) ∨ ¬ Col p A E
        using Col-def P2 l6-16-1 by blast
      thus ?thesis
        using f3 f2 by (metis (no-types) Col-def assms(3) not-out-bet one-side-chara one-side-symmetry)
    qed
    have U2: D ≠ E
      using P2 P4 bet-col not-col-permutation-5 by blast
    have U3: ¬ Col D E C
      by (metis Col-def P2 P4 P9 col-transitivity-1)
    have U4: Bet E D B
      by (simp add: P7 U1 out2--bet)
    have D C E LtA C D B
      using P5 U3 U4 l11-41-aux not-col-distincts by blast
    have U5: D E C LtA C D B
      using P7 U4 U3 l11-41 out-diff2 by auto
    have D E C CongA B E C
      by (simp add: P9 U1 l6-6 out2--conga out-trivial)
    thus ?thesis
      by (metis U5 conga-preserves-lta conga-pseudo-refl lta-distincts)
  qed
  then have B A C LtA B D C
    using P10 lta-trans by blast
```

}
thus *?thesis*
using *P2* ‹$E = B \Longrightarrow B\ A\ C\ LtA\ B\ D\ C$› **by** *blast*
qed

lemma *bet-le--lt*:
assumes *Bet A D B* **and**
$A \neq D$ **and**
$D \neq B$ **and**
A C Le B C
shows *D C Lt B C*
proof −
have *P1*: $A \neq B$
using *assms*(*1*) *assms*(*2*) *between-identity* **by** *blast*
have *C D Lt C B*
proof *cases*
assume *P3*: *Col A B C*
thus *?thesis*
proof *cases*
assume *Bet C D B*
thus *?thesis*
by (*simp add*: *assms*(*3*) *bet--lt1213*)
next
assume ¬ *Bet C D B*
then have *D Out C B*
by (*metis Out-def P1 P3 assms*(*1*) *col-transitivity-2 not-col-permutation-3 not-out-bet out-col*)
thus *?thesis*
by (*smt Le-cases P3 assms*(*1*) *assms*(*2*) *assms*(*4*) *bet2-le2--le bet-le-eq bet-out-1 l6-6 l6-7 nle--lt or-bet-out out2--bet out-bet--out*)
qed
next
assume *Q0A*: ¬ *Col A B C*
then have *Q0B*: ¬ *Col B C D*
by (*meson Col-def assms*(*1*) *assms*(*3*) *col-transitivity-2*)
have *C B D LtA C D B*
proof −
have *Q1*: *B Out C C*
using *Q0A not-col-distincts out-trivial* **by** *force*
have *Q2*: *B Out A D*
using *Out-cases assms*(*1*) *assms*(*3*) *bet-out-1* **by** *blast*
have *Q3*: *A Out C C*
by (*metis Q0A col-trivial-3 out-trivial*)
have *Q4*: *A Out B B*
using *P1 out-trivial* **by** *auto*
have *C B A LeA C A B*
using *Col-perm Le-cases Q0A assms*(*4*) *l11-44-2bis* **by** *blast*
then have *T9*: *C B D LeA C A B*
using *Q1 Q2 Q3 Q4 lea-out4--lea* **by** *blast*
have *C A B LtA C D B*
proof −
have *U2*: ¬ *Col D A C*
using *Q0B assms*(*1*) *assms*(*2*) *bet-col col-transitivity-2 not-col-permutation-3 not-col-permutation-4* **by** *blast*
have *U3*: $D \neq C$
using *Q0B col-trivial-2* **by** *blast*
have *U4*: *D C A LtA C D B*
using *U2 assms*(*1*) *assms*(*3*) *l11-41-aux* **by** *auto*
have *U5*: *D A C LtA C D B*
by (*simp add*: *U2 assms*(*1*) *assms*(*3*) *l11-41*)
have *A Out B D*
using *Out-def P1 assms*(*1*) *assms*(*2*) **by** *auto*
then have *D A C CongA C A B*
using *Q3 conga-right-comm out2--conga* **by** *blast*
thus *?thesis*
by (*metis U5 U3 assms*(*3*) *conga-preserves-lta conga-refl*)
qed
thus *?thesis*

        using *T9 lea123456-lta--lta* **by** *blast*
      qed
      thus *?thesis*
        by (*simp add*: *l11-44-2-b*)
    qed
    thus *?thesis*
      using *Lt-cases* **by** *auto*
qed

lemma *cong2--ncol*:
  assumes $A \neq B$ **and**
    $B \neq C$ **and**
    $A \neq C$ **and**
    Cong A P B P **and**
    Cong A P C P
  shows ¬ *Col A B C*
proof −
  have *Cong B P C P*
    using *assms*(*4*) *assms*(*5*) *cong-inner-transitivity* **by** *blast*
  thus *?thesis* **using** *bet-le--lt*
   by (*metis assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *cong--le cong--nlt lt--nle not-col-permutation-5 third-point*)
qed

lemma *cong4-cop2--eq*:
  assumes $A \neq B$ **and**
    $B \neq C$ **and**
    $A \neq C$ **and**
    Cong A P B P **and**
    Cong A P C P **and**
    Coplanar A B C P **and**
    Cong A Q B Q **and**
    Cong A Q C Q **and**
    Coplanar A B C Q
  shows $P = Q$
proof −
  have *P1*: ¬ *Col A B C*
    using *assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *cong2--ncol* **by** *auto*
  {
    assume *P2*: $P \neq Q$
    have *P3*: ∀ *R*. *Col P Q R* ⟶ (*Cong A R B R* ∧ *Cong A R C R*)
      using *P2 assms*(*4*) *assms*(*5*) *assms*(*7*) *assms*(*8*) *l4-17 not-cong-4321* **by** *blast*
    obtain *D* **where** *P4*: *D Midpoint A B*
      using *midpoint-existence* **by** *auto*
    have *P5*: *Coplanar A B C D*
      using *P4 coplanar-perm-9 midpoint--coplanar* **by** *blast*
    have *P6*: *Col P Q D*
    proof −
      have *P6A*: *Coplanar P Q D A*
        using *P1 P5 assms*(*6*) *assms*(*9*) *coplanar-pseudo-trans ncop-distincts* **by** *blast*
      then have *P6B*: *Coplanar P Q D B*
        by (*metis P4 col-cop--cop midpoint-col midpoint-distinct-1*)
      have *P6D*: *Cong P A P B*
        using *assms*(*4*) *not-cong-2143* **by** *blast*
      have *P6E*: *Cong Q A Q B*
        using *assms*(*7*) *not-cong-2143* **by** *blast*
      have *Cong D A D B*
        using *Midpoint-def P4 not-cong-2134* **by** *blast*
      thus *?thesis* **using** *cong3-cop2--col P6A P6B assms*(*1*) *P6D P6E* **by** *blast*
    qed
    obtain *R1* **where** *P7*: $P \neq R1 \wedge Q \neq R1 \wedge D \neq R1$ ∧ *Col P Q R1*
      using *P6 diff-col-ex3* **by** *blast*
    obtain *R2* **where** *P8*: *Bet R1 D R2* ∧ *Cong D R2 R1 D*
      using *segment-construction* **by** *blast*
    have *P9*: *Col P Q R2*
      by (*metis P6 P7 P8 bet-col colx*)
    have *P9A*: *Cong R1 A R1 B*

using *P3 P7 not-cong-2143* **by** *blast*
then have *Per R1 D A*
using *P4 Per-def* **by** *auto*
then have *Per A D R1* **using** *l8-2* **by** *blast*
have *P10*: *Cong A R1 A R2*
proof −
have *f1*: *Bet R2 D R1* ∨*Bet R1 R2 D*
by (*metis* (*full-types*) *Col-def P7 P8 between-equality not-col-permutation-5*)
have *f2*: *Cong B R1 A R1*
using *Cong-perm* ‹*Cong R1 A R1 B*› **by** *blast*
have *Cong B R1 A R2* ∨ *Bet R1 R2 D*
using *f1 Cong-perm Midpoint-def P4 P8 l7-13* **by** *blast*
thus *?thesis*
using *f2 P8 bet-cong-eq cong-inner-transitivity* **by** *blast*
qed
have *Col A B C*
proof −
have *P11*: *Cong B R1 B R2*
by (*metis Cong-perm P10 P3 P9 P9A cong-inner-transitivity*)
have *P12*: *Cong C R1 C R2*
using *P10 P3 P7 P9 cong-inner-transitivity* **by** *blast*
have *P12A*: *Coplanar A B C R1*
using *P2 P7 assms*(*6*) *assms*(*9*) *col-cop2--cop* **by** *blast*
have *P12B*: *Coplanar A B C R2*
using *P2 P9 assms*(*6*) *assms*(*9*) *col-cop2--cop* **by** *blast*
have *R1* ≠ *R2*
using *P7 P8 between-identity* **by** *blast*
thus *?thesis*
using *P10 P11 P12A P12B P12 cong3-cop2--col* **by** *blast*
qed
then have *False*
by (*simp add*: *P1*)
}
thus *?thesis* **by** *auto*
qed

lemma *t18-18-aux*:
assumes *Cong A B D E* **and**
Cong A C D F **and**
F D E LtA C A B **and**
¬ *Col A B C* **and**
¬ *Col D E F* **and**
D F Le D E
shows *E F Lt B C*
proof −
obtain *G0* **where** *P1*: *C A B CongA F D G0* ∧ *F D OS G0 E*
using *angle-construction-1 assms*(*4*) *assms*(*5*) *not-col-permutation-2* **by** *blast*
then have *P2*: ¬ *Col F D G0*
using *col123--nos* **by** *auto*
then obtain *G* **where** *P3*: *D Out G0 G* ∧ *Cong D G A B*
by (*metis assms*(*4*) *bet-col between-trivial2 col-trivial-2 segment-construction-3*)
have *P4*: *C A B CongA F D G*
proof −
have *P4B*: *A Out C C*
by (*metis assms*(*4*) *col-trivial-3 out-trivial*)
have *P4C*: *A Out B B*
by (*metis assms*(*4*) *col-trivial-1 out-trivial*)
have *P4D*: *D Out F F*
using *P2 not-col-distincts out-trivial* **by** *blast*
have *D Out G G0*
by (*simp add*: *P3 l6-6*)
thus *?thesis* **using** *P1 P4B P4C P4D*
using *l11-10* **by** *blast*
qed
have *D Out G G0*
by (*simp add*: *P3 l6-6*)

then have *D F OS G G0*
  using *P2 not-col-permutation-4 out-one-side* **by** *blast*
then have *F D OS G G0*
  by (*simp add*: *invert-one-side*)
then have *P5*: *F D OS G E*
  using *P1 one-side-transitivity* **by** *blast*
have *P6*: ¬ *Col F D G*
  by (*meson P5 one-side-not-col123*)
have *P7*: *Cong C B F G*
  using *P3 P4 assms*(*2*) *cong2-conga-cong cong-commutativity cong-symmetry* **by** *blast*
have *P8*: *F E Lt F G*
proof −
  have *P9*: *F D E LtA F D G*
    by (*metis P4 assms*(*3*) *assms*(*5*) *col-trivial-1 col-trivial-3 conga-preserves-lta conga-refl*)
  have *P10*: *Cong D G D E*
    using *P3 assms*(*1*) *cong-transitivity* **by** *blast*
  {
    assume *P11*: *Col E F G*
    have *P12*: *F D E LeA F D G*
      by (*simp add*: *P9 lta--lea*)
    have *P13*: ¬ *F D E CongA F D G*
      using *P9 not-lta-and-conga* **by** *blast*
    have *F D E CongA F D G*
    proof −
      have *F Out E G*
        using *Col-cases P11 P5 col-one-side-out l6-6* **by** *blast*
      then have *E F D CongA G F D*
        by (*metis assms*(*5*) *conga-os--out conga-refl l6-6 not-col-distincts one-side-reflexivity out2--conga*)
      thus *?thesis*
        by (*meson P10 assms*(*2*) *assms*(*6*) *cong-4321 cong-inner-transitivity l11-52 le-comm*)
    qed
    then have *False*
      using *P13* **by** *blast*
  }
  then have *P15*: ¬ *Col E F G* **by** *auto*
  {
    assume *P20*: *Col D E G*
    have *P21*: *F D E LeA F D G*
      by (*simp add*: *P9 lta--lea*)
    have *P22*: ¬ *F D E CongA F D G*
      using *P9 not-lta-and-conga* **by** *blast*
    have *F D E CongA F D G*
    proof −
      have *D Out E G*
        by (*meson Out-cases P5 P20 col-one-side-out invert-one-side not-col-permutation-5*)
      thus *?thesis*
        using *P10 P15* ‹*D Out G G0*› *cong-inner-transitivity l6-11-uniqueness l6-7 not-col-distincts* **by** *blast*
    qed
    then have *False*
      by (*simp add*: *P22*)
  }
  then have *P16*: ¬ *Col D E G* **by** *auto*
  have *P17*: *E InAngle F D G*
    using *lea-in-angle* **by** (*simp add*: *P5 P9 lta--lea*)
  then obtain *H* **where** *P18*: *Bet F H G* ∧ (*H* = *D* ∨ *D Out H E*)
    using *InAngle-def* **by** *auto*
  {
    assume *H* = *D*
    then have *F G E LtA F E G*
      using *P18 P6 bet-col* **by** *blast*
  }
  {
    assume *P19*: *D Out H E*
    have *P20*: *H* ≠ *F*
      using *Out-cases P19 assms*(*5*) *out-col* **by** *blast*
    have *P21*: *H* ≠ *G*

using *P19 P16 l6-6 out-col* **by** *blast*
have *F D Le G D*
using *P10 assms*(*6*) *cong-pseudo-reflexivity l5-6 not-cong-4312* **by** *blast*
then have *H D Lt G D*
using *P18 P20 P21 bet-le--lt* **by** *blast*
then have *P22*: *D H Lt D E*
using *Lt-cases P10 cong2-lt--lt cong-reflexivity* **by** *blast*
then have *P23*: *D H Le D E* $\wedge$ $\neg$ *Cong D H D E*
using *Lt-def* **by** *blast*
have *P24*: *H* $\neq$ *E*
using *P23 cong-reflexivity* **by** *blast*
have *P25*: *Bet D H E*
by (*simp add*: *P19 P23 l6-13-1*)
have *P26*: *E G OS F D*
by (*metis InAngle-def P15 P16 P18 P25 bet-out-1 between-symmetry in-angle-one-side not-col-distincts not-col-permutation-1*)
have *F G E LtA F E G*
proof −
have *P27*: *F G E LtA D E G*
proof −
have *P28*: *D G E CongA D E G*
by (*metis P10 P16 l11-44-1-a not-col-distincts*)
have *F G E LtA D G E*
proof −
have *P29*: *F G E LeA D G E*
by (*metis OS-def P17 P26 P5 TS-def in-angle-one-side inangle--lea-1 invert-one-side l11-24 os2--inangle*)
{
assume *F G E CongA D G E*
then have *E G F CongA E G D*
by (*simp add*: *conga-comm*)
then have *G Out F D*
using *P26 conga-os--out* **by** *auto*
then have *False*
using *P6 not-col-permutation-2 out-col* **by** *blast*
}
then have $\neg$ *F G E CongA D G E* **by** *auto*
thus *?thesis*
by (*simp add*: *LtA-def P29*)
qed
thus *?thesis*
by (*metis P28 P6 col-trivial-3 conga-preserves-lta conga-refl*)
qed
have *G E D LtA G E F*
proof −
have *P30*: *G E D LeA G E F*
proof −
have *P31*: *D InAngle G E F*
by (*simp add*: *P16 P17 P26 assms*(*5*) *in-angle-two-sides l11-24 not-col-permutation-5 os-ts--inangle*)
have *G E D CongA G E D*
by (*metis P16 col-trivial-1 col-trivial-2 conga-refl*)
thus *?thesis*
using *P31 inangle--lea* **by** *auto*
qed
have $\neg$ *G E D CongA G E F*
by (*metis OS-def P26 P5 TS-def conga-os--out invert-one-side out-col*)
thus *?thesis*
by (*simp add*: *LtA-def P30*)
qed
then have *D E G LtA F E G*
using *lta-comm* **by** *blast*
thus *?thesis*
using *P27 lta-trans* **by** *blast*
qed
}
then have *F G E LtA F E G*
using *P18* ‹*H* = *D* ⟹ *F G E LtA F E G*› **by** *blast*
thus *?thesis*

```
      by (simp add: l11-44-2-b)
  qed
  then have E F Lt F G
    using lt-left-comm by blast
  thus ?thesis
    using P7 cong2-lt--lt cong-pseudo-reflexivity not-cong-4312 by blast
qed

lemma t18-18:
  assumes Cong A B D E and
    Cong A C D F and
    F D E LtA C A B
  shows E F Lt B C
proof −
  have P1: F ≠ D
    using assms(3) lta-distincts by blast
  have P2: E ≠ D
    using assms(3) lta-distincts by blast
  have P3: C ≠ A
    using assms(3) lta-distincts by auto
  have P4: B ≠ A
    using assms(3) lta-distincts by blast
  {
    assume P6: Col A B C
    {
      assume P7: Bet B A C
      obtain C' where P8:Bet E D C' ∧ Cong D C' A C
        using segment-construction by blast
      have P9: Cong E F E F
        by (simp add: cong-reflexivity)
      have P10: Cong E C' B C
        using P7 P8 assms(1) l2-11-b not-cong-4321 by blast
      have E F Lt E C'
      proof −
        have P11: Cong D F D C'
          using P8 assms(2) cong-transitivity not-cong-3412 by blast
        have ¬ Bet E D F
          using Bet-perm Col-def assms(3) col-lta--out not-bet-and-out by blast
        thus ?thesis
          using P11 P8 triangle-strict-inequality by blast
      qed
      then have E F Lt B C
        using P9 P10 cong2-lt--lt by blast
    }
    {
      assume ¬ Bet B A C
      then have E F Lt B C
        using P6 assms(3) between-symmetry col-lta--bet col-permutation-2 by blast
    }
    then have E F Lt B C
      using ‹Bet B A C ⟹ E F Lt B C› by auto
  }
  {
    assume P12: ¬ Col A B C
    {
      assume P13: Col D E F
      {
        assume P14: Bet F D E
        then have C A B LeA F D E
          by (simp add: P1 P2 P3 P4 l11-31-2)
        then have F D E LtA F D E
          using assms(3) lea--nlta by auto
        then have False
          by (simp add: nlta)
        then have E F Lt B C by auto
      }
```

```
      {
        assume ¬ Bet F D E
        then have P16: D Out F E
          using P13 not-col-permutation-1 not-out-bet by blast
        obtain F' where P17: A Out B F' ∧ Cong A F' A C
          using P3 P4 segment-construction-3 by fastforce
        then have P18: B F' Lt B C
       by (meson P12 Tarski-neutral-dimensionless.triangle-strict-reverse-inequality Tarski-neutral-dimensionless-axioms
not-cong-3412 out-col)
        have Cong B F' E F
          by (meson Out-cases P16 P17 assms(1) assms(2) cong-transitivity out-cong-cong)
        then have E F Lt B C
          using P18 cong2-lt--lt cong-reflexivity by blast
      }
      then have E F Lt B C
        using ‹Bet F D E ⟹ E F Lt B C› by blast
    }
    {
      assume P20: ¬ Col D E F
      {
        assume D F Le D E
        then have E F Lt B C
            by (meson P12 Tarski-neutral-dimensionless.t18-18-aux Tarski-neutral-dimensionless-axioms P20 assms(1)
assms(2) assms(3))
      }
      {
        assume D E Le D F
        then have E F Lt B C
       by (meson P12 P20 Tarski-neutral-dimensionless.lta-comm Tarski-neutral-dimensionless.t18-18-aux Tarski-neutral-dimensionless
assms(1) assms(2) assms(3) lt-comm not-col-permutation-5)
      }
      then have E F Lt B C
        using ‹D F Le D E ⟹ E F Lt B C› local.le-cases by blast
    }
    then have E F Lt B C
      using ‹Col D E F ⟹ E F Lt B C› by blast
  }
  thus ?thesis
    using ‹Col A B C ⟹ E F Lt B C› by auto
qed

lemma t18-19:
  assumes A ≠ B and
    A ≠ C and
    Cong A B D E and
    Cong A C D F and
    E F Lt B C
  shows F D E LtA C A B
proof −
  {
    assume P1: C A B LeA F D E
    {
      assume C A B CongA F D E
      then have False
        using Cong-perm assms(3) assms(4) assms(5) cong--nlt l11-49 by blast
    }
    {
      assume P2: ¬ C A B CongA F D E
      then have C A B LtA F E D
        by (metis P1 assms(3) assms(4) assms(5) cong-symmetry lea-distincts lta--nlea not-and-lt or-lta2-conga t18-18)
      then have B C Lt E F
        by (metis P1 P2 assms(3) assms(4) cong-symmetry lta--nlea lta-distincts or-lta2-conga t18-18)
      then have False
        using assms(5) not-and-lt by auto
    }
    then have False
```

```
    using ‹C A B CongA F D E ⟹ False› by auto
  }
  then have ¬ C A B LeA F D E by auto
  thus ?thesis
    using assms(1) assms(2) assms(3) assms(4) cong-identity nlea--lta by blast
qed

lemma acute-trivial:
  assumes A ≠ B
  shows Acute A B A
 by (metis Tarski-neutral-dimensionless.acute-distincts Tarski-neutral-dimensionless-axioms angle-partition assms l11-43)

lemma acute-not-per:
  assumes Acute A B C
  shows ¬ Per A B C
proof –
  obtain A' B' C' where P1: Per A' B' C' ∧ A B C LtA A' B' C'
    using Acute-def assms by auto
  thus ?thesis
    using acute-distincts acute-per--lta assms nlta by fastforce
qed

lemma angle-bisector:
  assumes A ≠ B and
    C ≠ B
  shows ∃ P. (P InAngle A B C ∧ P B A CongA P B C)
proof cases
  assume P1: Col A B C
  thus ?thesis
  proof cases
    assume P2: Bet A B C
    then obtain Q where P3: ¬ Col A B Q
      using assms(1) not-col-exists by auto
    then obtain P where P4: A B Perp P B ∧ A B OS Q P
      using P1 l10-15 os-distincts by blast
    then have P5: P InAngle A B C
      by (metis P2 assms(2) in-angle-line os-distincts)
    have P B A CongA P B C
    proof –
      have P9: P ≠ B
        using P4 os-distincts by blast
      have Per P B A
        by (simp add: P4 Perp-perm Tarski-neutral-dimensionless.perp-per-2 Tarski-neutral-dimensionless-axioms)
      thus ?thesis
        using P2 assms(1) assms(2) P9 l11-18-1 by auto
    qed
    thus ?thesis
      using P5 by auto
  next
    assume T1: ¬ Bet A B C
    then have T2: B Out A C
      by (simp add: P1 l6-4-2)
    have T3: C InAngle A B C
      by (simp add: assms(1) assms(2) inangle3123)
    have C B A CongA C B C
      using T2 between-trivial2 l6-6 out2--conga out2-bet-out by blast
    thus ?thesis
      using T3 by auto
  qed
next
  assume T4: ¬ Col A B C
  obtain C0 where T5: B Out C0 C ∧ Cong B C0 B A
    using assms(1) assms(2) l6-11-existence by fastforce
  obtain P where T6: P Midpoint A C0
    using midpoint-existence by auto
  have T6A: ¬ Col A B C0
```

by (*metis T4 T5 col3 l6-3-1 not-col-distincts out-col*)
have *T6B*: $P \neq B$
using *Col-def Midpoint-def T6 T6A* **by** *auto*
have *T6D*: $P \neq A$
using *T6 T6A is-midpoint-id not-col-distincts* **by** *blast*
have *P InAngle A B C0*
using *InAngle-def T5 T6 T6B assms*(*1*) *l6-3-1 midpoint-bet out-trivial* **by** *fastforce*
then have *T7*: *P InAngle A B C*
using *T5 T6B in-angle-trans2 l11-24 out341--inangle* **by** *blast*
have *T8*: $(P = B) \vee B$ *Out P P*
using *out-trivial* **by** *auto*
have *T9*: *B Out A A*
by (*simp add*: *assms*(*1*) *out-trivial*)
{
assume *T9A*: *B Out P P*
have *P B A CongA P B C0* $\wedge$ *B P A CongA B P C0* $\wedge$ *P A B CongA P C0 B*
proof −
have *T9B*: *Cong B P B P*
by (*simp add*: *cong-reflexivity*)
have *T9C*: *Cong B A B C0*
using *Cong-perm T5* **by** *blast*
have *Cong P A P C0*
using *Midpoint-def T6 not-cong-2134* **by** *blast*
thus *?thesis* **using** *l11-51 T6B assms*(*1*) *T9B T9C T6D* **by** *presburger*
qed
then have *P B A CongA P B C0* **by** *auto*
then have *P B A CongA P B C* **using** *l11-10 T9A T9*
by (*meson T5 l6-6*)
then have $\exists$ *P*. (*P InAngle A B C* $\wedge$ *P B A CongA P B C*)
using *T7* **by** *auto*
}
thus *?thesis*
using *T6B T8* **by** *blast*
qed

lemma *reflectl--conga*:
assumes $A \neq B$ **and**
$B \neq P$ **and**
P P′ ReflectL A B
shows *A B P CongA A B P′*
proof −
obtain *A′* **where** *P1*: *A′ Midpoint P′ P* $\wedge$ *Col A B A′* $\wedge$ (*A B Perp P′ P* $\vee$ $P = P'$)
using *ReflectL-def assms*(*3*) **by** *auto*
{
assume *P2*: *A B Perp P′ P*
then have *P3*: $P \neq P'$
using *perp-not-eq-2* **by** *blast*
then have *P4*: $A' \neq P'$
using *P1 is-midpoint-id* **by** *blast*
have *P5*: $A' \neq P$
using *P1 P3 is-midpoint-id-2* **by** *auto*
have *A B P CongA A B P′*
proof *cases*
assume *P6*: $A' = B$
then have *P8*: $B \neq P'$
using *P4* **by** *auto*
have *P9*: *Per A B P*
by (*smt P1 P3 P6 Perp-cases col-transitivity-2 midpoint-col midpoint-distinct-1 not-col-permutation-2 perp-col2-bis perp-per-2*)
have *Per A B P′*
by (*smt Mid-cases P1 P2 P6 P8 assms*(*1*) *col-trivial-3 midpoint-col not-col-permutation-3 perp-col4 perp-per-2*)
thus *?thesis*
using *l11-16 P4 P5 P6 P9 assms*(*1*) **by** *auto*
next
assume *T1*: $A' \neq B$
have *T2*: *B A′ P CongA B A′ P′*

```
      proof –
        have T2A: Cong B P B P′
          using assms(3) col-trivial-2 is-image-spec-col-cong l10-4-spec not-cong-4321 by blast
        then have T2B: A′ B P CongA A′ B P′
       by (metis Cong-perm Midpoint-def P1 P5 T1 Tarski-neutral-dimensionless.l11-51 Tarski-neutral-dimensionless-axioms
assms(2) cong-reflexivity)
        have A′ P B CongA A′ P′ B
          by (simp add: P5 T2A T2B cong-reflexivity conga-comm l11-49)
        thus ?thesis
          using P5 T2A T2B cong-reflexivity l11-49 by blast
      qed
      have T3: Cong A′ B A′ B
        by (simp add: cong-reflexivity)
      have Cong A′ P A′ P′
        using Midpoint-def P1 not-cong-4312 by blast
      then have T4: A′ B P CongA A′ B P′ ∧ A′ P B CongA A′ P′ B using l11-49
        using assms(2) T2 T3 by blast
      show ?thesis
      proof cases
        assume Bet A′ B A
        thus ?thesis
          using T4 assms(1) l11-13 by blast
      next
        assume ¬ Bet A′ B A
        then have T5: B Out A′ A
          using P1 not-col-permutation-3 or-bet-out by blast
        have T6: B ≠ P′
          using T4 conga-distinct by blast
        have T8: B Out A A′
          by (simp add: T5 l6-6)
        have T9: B Out P P
          using assms(2) out-trivial by auto
        have B Out P′ P′
          using T6 out-trivial by auto
        thus ?thesis
          using l11-10 T4 T8 T9 by blast
      qed
    qed
  }
  {
    assume P = P′
    then have A B P CongA A B P′
      using assms(1) assms(2) conga-refl by auto
  }
  thus ?thesis
    using P1 ‹A B Perp P′ P ⟹ A B P CongA A B P′› by blast
qed

lemma conga-cop-out-reflectl--out:
  assumes ¬ B Out A C and
    Coplanar A B C P and
    P B A CongA P B C and
    B Out A T and
    T T′ ReflectL B P
  shows B Out C T′
proof –
  have P1: P B T CongA P B T′
    by (metis assms(3) assms(4) assms(5) conga-distinct is-image-spec-rev out-distinct reflectl--conga)
  have P2: T T′ Reflect B P
    by (metis P1 assms(5) conga-distinct is-image-is-image-spec)
  have P3: B ≠ T′
    using CongA-def P1 by blast
  have P4: P B C CongA P B T′
  proof –
    have P5: P B C CongA P B A
      by (simp add: assms(3) conga-sym)
```

```
    have P B A CongA P B T'
    proof -
      have P7: B Out P P
        using assms(3) conga-diff45 out-trivial by blast
      have P8: B Out A T
        by (simp add: assms(4))
      have B Out T' T'
        using P3 out-trivial by auto
      thus ?thesis
        using P1 P7 P8 l11-10 by blast
    qed
    thus ?thesis
      using P5 not-conga by blast
  qed
  have P B OS C T'
  proof -
    have P9: P B TS A C
      using assms(1) assms(2) assms(3) conga-cop--or-out-ts coplanar-perm-20 by blast
    then have T ≠ T'
      by (metis Col-perm P2 P3 TS-def assms(4) col-transitivity-2 l10-8 out-col)
    then have P B TS T T'
      by (metis P2 P4 conga-diff45 invert-two-sides l10-14)
    then have P B TS A T'
      using assms(4) col-trivial-2 out-two-sides-two-sides by blast
    thus ?thesis
      using OS-def P9 l9-2 by blast
  qed
  thus ?thesis
    using P4 conga-os--out by auto
qed

lemma col-conga-cop-reflectl--col:
  assumes ¬ B Out A C and
    Coplanar A B C P and
    P B A CongA P B C and
    Col B A T and
    T T' ReflectL B P
  shows Col B C T'
proof cases
  assume B = T
  thus ?thesis
    using assms(5) col-image-spec--eq not-col-distincts by blast
next
  assume P1: B ≠ T
  thus ?thesis
  proof cases
    assume B Out A T
    thus ?thesis
      using out-col conga-cop-out-reflectl--out assms(1) assms(2) assms(3) assms(5) by blast
  next
    assume P2: ¬ B Out A T
    obtain A' where P3: Bet A B A' ∧ Cong B A' A B
      using segment-construction by blast
    obtain C' where P4: Bet C B C' ∧ Cong B C' C B
      using segment-construction by blast
    have P5: B Out C' T'
    proof -
      have P6: ¬ B Out A' C'
        by (metis P3 P4 assms(1) between-symmetry cong-diff-2 l6-2 out-diff1 out-diff2)
      have P7: Coplanar A' B C' P
      proof cases
        assume Col A B C
        thus ?thesis
         by (smt P3 P4 assms(1) assms(2) assms(3) bet-col bet-neq32--neq col2-cop--cop col-transitivity-1 colx conga-diff2
conga-diff56 l6-4-2 ncoplanar-perm-15 not-col-permutation-5)
      next
```

assume *P7B*: ¬ *Col A B C*
have *P7C*: *Coplanar A B C A′*
using *P3 bet-col ncop--ncols* **by** *blast*
have *P7D*: *Coplanar A B C B*
using *ncop-distincts* **by** *blast*
have *Coplanar A B C C′*
using *P4 bet--coplanar coplanar-perm-20* **by** *blast*
thus *?thesis*
using *P7B P7C P7D assms(2) coplanar-pseudo-trans* **by** *blast*
qed
have *P8*: *P B A′ CongA P B C′*
by (*metis CongA-def P3 P4 assms(3) cong-reverse-identity conga-left-comm l11-13 not-conga-sym*)
have *P9*: *B Out A′ T*
by (*smt Out-def P1 P2 P3 P8 assms(3) assms(4) conga-distinct l5-2 l6-4-2 not-col-permutation-4*)
thus *?thesis*
using *P6 P7 P8 P9 assms(5) conga-cop-out-reflectl--out* **by** *blast*
qed
thus *?thesis*
by (*metis Col-def P4 col-transitivity-1 out-col out-diff1*)
qed
qed

lemma *conga2-cop2--col*:
assumes ¬ *B Out A C* **and**
P B A CongA P B C **and**
P′ B A CongA P′ B C **and**
Coplanar A B P P′ **and**
Coplanar B C P P′
shows *Col B P P′*
proof −
obtain *C′* **where** *P1*: *B Out C′ C* ∧ *Cong B C′ B A*
by (*metis assms(2) conga-distinct l6-11-existence*)
have *P1A*: *Cong P A P C′* ∧ (*P* ≠ *A* ⟶ (*B P A CongA B P C′* ∧ *B A P CongA B C′ P*))
proof −
have *P2*: *P B A CongA P B C′*
proof −
have *P2A*: *B Out P P*
using *assms(2) conga-diff45 out-trivial* **by** *auto*
have *B Out A A*
using *assms(2) conga-distinct out-trivial* **by** *auto*
thus *?thesis*
using *P1 P2A assms(2) l11-10* **by** *blast*
qed
have *P3*: *Cong B P B P*
by (*simp add*: *cong-reflexivity*)
have *Cong B A B C′*
using *Cong-perm P1* **by** *blast*
thus *?thesis* **using** *l11-49 P2 cong-reflexivity* **by** *blast*
qed
have *P4*: *P′ B A CongA P′ B C′*
proof −
have *P4A*: *B Out P′ P′*
using *assms(3) conga-diff1 out-trivial* **by** *auto*
have *B Out A A*
using *assms(2) conga-distinct out-trivial* **by** *auto*
thus *?thesis*
using *P1 P4A assms(3) l11-10* **by** *blast*
qed
have *P5*: *Cong B P′ B P′*
by (*simp add*: *cong-reflexivity*)
have *P5A*: *Cong B A B C′*
using *Cong-perm P1* **by** *blast*
then have *P6*: *P′* ≠ *A* ⟶ (*B P′ A CongA B P′ C′* ∧ *B A P′ CongA B C′ P′*)
using *P4 P5 l11-49* **by** *blast*
have *P7*: *Coplanar B P P′ A*
using *assms(4) ncoplanar-perm-18* **by** *blast*

have *P8*: *Coplanar B P P′ C′*
by (*smt Col-cases P1 assms*(*5*) *col-cop--cop ncoplanar-perm-16 ncoplanar-perm-8 out-col out-diff2*)
have $A \neq C'$
using *P1 assms*(*1*) **by** *auto*
thus *?thesis*
using *P4 P5 P7 P8 P5A P1A cong3-cop2--col l11-49* **by** *blast*
qed

lemma *conga2-cop2--col-1*:
assumes ¬ *Col A B C* **and**
P B A CongA P B C **and**
P′ B A CongA P′ B C **and**
Coplanar A B C P **and**
Coplanar A B C P′
shows *Col B P P′*
proof −
have *P1*: ¬ *B Out A C*
using *Col-cases assms*(*1*) *out-col* **by** *blast*
have *P2*: *Coplanar A B P P′*
by (*meson assms*(*1*) *assms*(*4*) *assms*(*5*) *coplanar-perm-12 coplanar-trans-1 not-col-permutation-2*)
have *Coplanar B C P P′*
using *assms*(*1*) *assms*(*4*) *assms*(*5*) *coplanar-trans-1* **by** *auto*
thus *?thesis* **using** *P1 P2 conga2-cop2--col assms*(*2*) *assms*(*3*) *conga2-cop2--col* **by** *auto*
qed

lemma *col-conga--conga*:
assumes *P B A CongA P B C* **and**
Col B P P′ **and**
$B \neq P'$
shows *P′ B A CongA P′ B C*
proof *cases*
assume *Bet P B P′*
thus *?thesis*
using *assms*(*1*) *assms*(*3*) *l11-13* **by** *blast*
next
assume ¬ *Bet P B P′*
then have *P1*: *B Out P P′*
using *Col-cases assms*(*2*) *or-bet-out* **by** *blast*
then have *P2*: *B Out P′ P*
by (*simp add*: *l6-6*)
have *P3*: *B Out A A*
using *CongA-def assms*(*1*) *out-trivial* **by** *auto*
have *B Out C C*
using *assms*(*1*) *conga-diff56 out-trivial* **by** *blast*
thus *?thesis*
using *P2 P3 assms*(*1*) *l11-10* **by** *blast*
qed

lemma *cop-inangle--ex-col-inangle*:
assumes ¬ *B Out A C* **and**
P InAngle A B C **and**
Coplanar A B C Q
shows ∃ *R*. (*R InAngle A B C* ∧ $P \neq R$ ∧ *Col P Q R*)
proof −
have *P1*: $A \neq B$
using *assms*(*2*) *inangle-distincts* **by** *blast*
then have *P4*: $A \neq C$
using *assms*(*1*) *out-trivial* **by** *blast*
have *P2*: $C \neq B$
using *assms*(*2*) *inangle-distincts* **by** *auto*
have *P3*: $P \neq B$
using *InAngle-def assms*(*2*) **by** *auto*
thus *?thesis*
proof *cases*
assume $P = Q$
thus *?thesis*

using *P1 P2 P4 col-trivial-1 inangle1123 inangle3123* **by** *blast*
next
assume *P5*: $P \neq Q$
thus *?thesis*
proof *cases*
assume *P6*: *Col B P Q*
obtain *R* **where** *P7*: *Bet B P R* ∧ *Cong P R B P*
using *segment-construction* **by** *blast*
have *P8*: *R InAngle A B C*
using *Out-cases P1 P2 P3 P7 assms*(*2*) *bet-out l11-25 out-trivial* **by** *blast*
have $P \neq R$
using *P3 P7 cong-reverse-identity* **by** *blast*
thus *?thesis*
by (*metis P3 P6 P7 P8 bet-col col-transitivity-2*)
next
assume *T1*: ¬ *Col B P Q*
thus *?thesis*
proof *cases*
assume *T2*: *Col A B C*
have *T3*: *Q InAngle A B C*
by (*metis P1 P2 T1 T2 assms*(*1*) *in-angle-line l6-4-2 not-col-distincts*)
thus *?thesis*
using *P5 col-trivial-2* **by** *blast*
next
assume *Q1*: ¬ *Col A B C*
thus *?thesis*
proof *cases*
assume *Q2*: *Col B C P*
have *Q3*: ¬ *Col B A P*
using *Col-perm P3 Q1 Q2 col-transitivity-2* **by** *blast*
have *Q4*: *Coplanar B P Q A*
using *P2 Q2 assms*(*3*) *col2-cop--cop col-trivial-3 ncoplanar-perm-22 ncoplanar-perm-3* **by** *blast*
have *Q5*: $Q \neq P$
using *P5* **by** *auto*
have *Q6*: *Col B P P*
using *not-col-distincts* **by** *blast*
have *Q7*: *Col Q P P*
using *not-col-distincts* **by** *auto*
have ¬ *Col B P A*
using *Col-cases Q3* **by** *auto*
then obtain *Q0* **where** *P10*: *Col Q P Q0* ∧ *B P OS A Q0*
using *cop-not-par-same-side Q4 Q5 Q6 Q7 T1* **by** *blast*
have *P13*: $P \neq Q0$
using *P10 os-distincts* **by** *auto*
{
assume *B A OS P Q0*
then have *?thesis*
using *P10 P13 assms*(*2*) *in-angle-trans not-col-permutation-4 os2--inangle* **by** *blast*
}
{
assume *V1*: ¬ *B A OS P Q0*
have ∃ *R*. *Bet P R Q0* ∧ *Col P Q R* ∧ *Col B A R*
proof *cases*
assume *V3*: *Col B A Q0*
have *Col P Q Q0*
using *Col-cases P10* **by** *auto*
thus *?thesis*
using *V3 between-trivial* **by** *auto*
next
assume *V4*: ¬ *Col B A Q0*
then have *V5*: ¬ *Col Q0 B A*
using *Col-perm* **by** *blast*
have ¬ *Col P B A*
using *Col-cases Q3* **by** *blast*
then obtain *R* **where** *V8*: *Col R B A* ∧ *Bet P R Q0*
using *cop-nos--ts V1 V5*

by (*meson P10 TS-def ncoplanar-perm-2 os--coplanar*)
thus *?thesis*
by (*metis Col-def P10 P13 col-transitivity-2*)
qed
then obtain *R* **where** *V9*: *Bet P R Q0* ∧ *Col P Q R* ∧ *Col B A R* **by** *auto*
have *V10*: *P* ≠ *R*
using *Q3 V9* **by** *blast*
have *R InAngle A B C*
proof −
have *W1*: ¬ *Col B P Q0*
using *P10 P13 T1 col2--eq* **by** *blast*
have *P Out Q0 R*
using *V10 V9 bet-out l6-6* **by** *auto*
then have *B P OS Q0 R*
using *Q6 W1 out-one-side-1* **by** *blast*
then have *B P OS A R*
using *P10 one-side-transitivity* **by** *blast*
then have *B Out A R*
using *V9 col-one-side-out* **by** *auto*
thus *?thesis*
by (*simp add*: *P2 out321--inangle*)
qed
then have *?thesis*
using *V10 V9* **by** *blast*
}
thus *?thesis*
using ‹*B A OS P Q0* ⟹ ∃ *R*. *R InAngle A B C* ∧ *P* ≠ *R* ∧ *Col P Q R*› **by** *blast*
next
assume *Z1*: ¬ *Col B C P*
then have *Z6*: ¬ *Col B P C*
by (*simp add*: *not-col-permutation-5*)
have *Z3*: *Col B P P*
by (*simp add*: *col-trivial-2*)
have *Z4*: *Col Q P P*
by (*simp add*: *col-trivial-2*)
have *Coplanar A B C P*
using *Q1 assms*(*2*) *inangle--coplanar ncoplanar-perm-18* **by** *blast*
then have *Coplanar B P Q C*
using *Q1 assms*(*3*) *coplanar-trans-1 ncoplanar-perm-5* **by** *blast*
then obtain *Q0* **where** *Z5*: *Col Q P Q0* ∧ *B P OS C Q0*
using *cop-not-par-same-side* **by** (*metis Z3 Z4 T1 Z6*)
thus *?thesis*
proof *cases*
assume *B C OS P Q0*
thus *?thesis*
proof −
have ∀ *p*. *p InAngle C B A* ∨ ¬ *p InAngle C B P*
using *assms*(*2*) *in-angle-trans l11-24* **by** *blast*
then show *?thesis*
by (*metis Col-perm Z5* ‹*B C OS P Q0*› *l11-24 os2--inangle os-distincts*)
qed
next
assume *Z6*: ¬ *B C OS P Q0*
have *Z7*: ∃ *R*. *Bet P R Q0* ∧ *Col P Q R* ∧ *Col B C R*
proof *cases*
assume *Col B C Q0*
thus *?thesis*
using *Col-def Col-perm Z5 between-trivial* **by** *blast*
next
assume *Z8*: ¬ *Col B C Q0*
have ∃ *R*. *Col R B C* ∧ *Bet P R Q0*
proof −
have *Z10*: *Coplanar B C P Q0*
using *Z5 ncoplanar-perm-2 os--coplanar* **by** *blast*
have *Z11*: ¬ *Col P B C*
using *Col-cases Z1* **by** *blast*

have ¬ *Col Q0 B C*
using *Col-perm Z8* **by** *blast*
thus *?thesis*
using *cop-nos--ts Z6 Z10 Z11* **by** (*simp add*: *TS-def*)
qed
then obtain *R* **where** *Col R B C* ∧ *Bet P R Q0* **by** *blast*
thus *?thesis*
by (*smt Z5 bet-col col2--eq col-permutation-1 os-distincts*)
qed
then obtain *R* **where** *Z12*: *Bet P R Q0* ∧ *Col P Q R* ∧ *Col B C R* **by** *blast*
have *Z13*: *P* ≠ *R*
using *Z1 Z12* **by** *auto*
have *Z14*: ¬ *Col B P Q0*
using *Z5 one-side-not-col124* **by** *blast*
have *P Out Q0 R*
using *Z12 Z13 bet-out l6-6* **by** *auto*
then have *B P OS Q0 R*
using *Z14 Z3 out-one-side-1* **by** *blast*
then have *B P OS C R*
using *Z5 one-side-transitivity* **by** *blast*
then have *B Out C R*
using *Z12 col-one-side-out* **by** *blast*
then have *R InAngle A B C*
using *P1 out341--inangle* **by** *auto*
thus *?thesis*
using *Z12 Z13* **by** *auto*
qed
qed
qed
qed
qed
qed

lemma *col-inangle2--out*:
assumes ¬ *Bet A B C* **and**
P InAngle A B C **and**
Q InAngle A B C **and**
Col B P Q
shows *B Out P Q*
proof *cases*
assume *Col A B C*
thus *?thesis*
by (*meson assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *bet-in-angle-bet bet-out--bet in-angle-out l6-6 not-col-permutation-4 or-bet-out*)
next
assume *P1*: ¬ *Col A B C*
thus *?thesis*
proof *cases*
assume *Col B A P*
thus *?thesis*
by (*meson assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *bet-in-angle-bet bet-out--bet l6-6 not-col-permutation-4 or-bet-out*)
next
assume *P2*: ¬ *Col B A P*
have ¬ *Col B A Q*
using *P2 assms*(*3*) *assms*(*4*) *col2--eq col-permutation-4 inangle-distincts* **by** *blast*
then have *B A OS P Q*
using *P1 P2 assms*(*2*) *assms*(*3*) *inangle-one-side invert-one-side not-col-permutation-4* **by** *auto*
thus *?thesis*
using *assms*(*4*) *col-one-side-out* **by** *auto*
qed
qed

lemma *inangle2--lea*:
assumes *P InAngle A B C* **and**
Q InAngle A B C
shows *P B Q LeA A B C*

proof −
  have *P1*: *P InAngle C B A*
    by (*simp add*: *assms*(*1*) *l11-24*)
  have *P2*: *Q InAngle C B A*
    by (*simp add*: *assms*(*2*) *l11-24*)
  have *P3*: *A* ≠ *B*
    using *assms*(*1*) *inangle-distincts* **by** *auto*
  have *P4*: *C* ≠ *B*
    using *assms*(*1*) *inangle-distincts* **by** *blast*
  have *P5*: *P* ≠ *B*
    using *assms*(*1*) *inangle-distincts* **by** *auto*
  have *P6*: *Q* ≠ *B*
    using *assms*(*2*) *inangle-distincts* **by** *auto*
  thus *?thesis*
  proof *cases*
    assume *P7*: *Col A B C*
    thus *?thesis*
    proof *cases*
      assume *Bet A B C*
      thus *?thesis*
        by (*simp add*: *P3 P4 P5 P6 l11-31-2*)
    next
      assume ¬ *Bet A B C*
      then have *B Out A C*
        using *P7 not-out-bet* **by** *blast*
      then have *B Out P Q*
        using *Out-cases assms*(*1*) *assms*(*2*) *in-angle-out l6-7* **by** *blast*
      thus *?thesis*
        by (*simp add*: *P3 P4 l11-31-1*)
    qed
  next
    assume *T1*: ¬ *Col A B C*
    thus *?thesis*
    proof *cases*
      assume *T2*: *Col B P Q*
      have ¬ *Bet A B C*
        using *T1 bet-col* **by** *auto*
      then have *B Out P Q*
        using *T2 assms*(*1*) *assms*(*2*) *col-inangle2--out* **by** *auto*
      thus *?thesis*
        by (*simp add*: *P3 P4 l11-31-1*)
    next
      assume *T3*: ¬ *Col B P Q*
      thus *?thesis*
      proof *cases*
        assume *Col B A P*
        then have *B Out A P*
          using *Col-def T1 assms*(*1*) *col-in-angle-out* **by** *blast*
        then have *P B Q CongA A B Q*
          using *P6 out2--conga out-trivial* **by** *auto*
        thus *?thesis*
          using *LeA-def assms*(*2*) **by** *blast*
      next
        assume *W0*: ¬ *Col B A P*
        show *?thesis*
        proof *cases*
          assume *Col B C P*
          then have *B Out C P*
            by (*metis P1 P3 T1 bet-out-1 col-in-angle-out out-col*)
          thus *?thesis*
              by (*smt P3 P4 P6 Tarski-neutral-dimensionless.lea-left-comm Tarski-neutral-dimensionless.lea-out4--lea Tarski-neutral-dimensionless-axioms assms*(*2*) *inangle--lea-1 out-trivial*)
        next
          assume *W0A*: ¬ *Col B C P*
          show *?thesis*
          proof *cases*

assume *Col B A Q*
then have *B Out A Q*
using *Col-def T1 assms(2) col-in-angle-out* **by** *blast*
thus *?thesis*
by (*smt P3 P4 P5 Tarski-neutral-dimensionless.lea-left-comm Tarski-neutral-dimensionless.lea-out4--lea Tarski-neutral-dimensionless-axioms assms(1) inangle--lea out-trivial*)
next
assume *W0AA*: ¬ *Col B A Q*
thus *?thesis*
proof *cases*
assume *Col B C Q*
then have *B Out C Q*
using *Bet-cases P2 T1 bet-col col-in-angle-out* **by** *blast*
thus *?thesis*
by (*smt P1 P3 P4 P5 Tarski-neutral-dimensionless.lea-comm Tarski-neutral-dimensionless.lea-out4--lea Tarski-neutral-dimensionless-axioms inangle--lea out-trivial*)
next
assume *W0B*: ¬ *Col B C Q*
have *W1*: *Coplanar B P A Q*
by (*metis Col-perm T1 assms(1) assms(2) col--coplanar inangle-one-side ncoplanar-perm-13 os--coplanar*)
have *W2*: ¬ *Col A B P*
by (*simp add*: *W0 not-col-permutation-4*)
have *W3*: ¬ *Col Q B P*
using *Col-perm T3* **by** *blast*
then have *W4*: *B P TS A Q* ∨ *B P OS A Q*
using *cop--one-or-two-sides*
by (*simp add*: *W1 W2*)
{
assume *W4A*: *B P TS A Q*
have *Q InAngle P B C*
proof −
have *W5*: *P B OS C Q*
using *OS-def P1 W0 W0A W4A in-angle-two-sides invert-two-sides l9-2* **by** *blast*
have *C B OS P Q*
by (*meson P1 P2 T1 W0A W0B inangle-one-side not-col-permutation-3 not-col-permutation-4*)
thus *?thesis*
by (*simp add*: *W5 invert-one-side os2--inangle*)
qed
then have *P B Q LeA A B C*
by (*meson assms(1) inangle--lea inangle--lea-1 lea-trans*)
}
{
assume *W6*: *B P OS A Q*
have *B A OS P Q*
using *Col-perm T1 W2 W0AA assms(1) assms(2) inangle-one-side invert-one-side* **by** *blast*
then have *Q InAngle P B A*
by (*simp add*: *W6 os2--inangle*)
then have *P B Q LeA A B C*
by (*meson P1 inangle--lea inangle--lea-1 lea-right-comm lea-trans*)
}
thus *?thesis*
using *W4* ‹*B P TS A Q* ⟹ *P B Q LeA A B C*› **by** *blast*
qed
qed
qed
qed
qed
qed
qed

lemma *conga-inangle-per--acute*:
assumes *Per A B C* **and**
P InAngle A B C **and**
P B A CongA P B C
shows *Acute A B P*
proof −

have *P1*: ¬ *Col A B C*
using *assms*(*1*) *assms*(*3*) *conga-diff2 conga-diff56 l8-9* **by** *blast*
have *P2*: *A B P LeA A B C*
by (*simp add*: *assms*(*2*) *inangle--lea*)
{
assume *A B P CongA A B C*
then have *P3*: *Per A B P*
by (*meson Tarski-neutral-dimensionless.l11-17 Tarski-neutral-dimensionless.not-conga-sym Tarski-neutral-dimensionless-axioms assms*(*1*))
have *P4*: *Coplanar P C A B*
using *assms*(*2*) *inangle--coplanar ncoplanar-perm-3* **by** *blast*
have *P5*: *P* ≠ *B*
using *assms*(*2*) *inangle-distincts* **by** *blast*
have *Per C B P*
using *P3 Per-cases assms*(*3*) *l11-17* **by** *blast*
then have *False*
using *P1 P3 P4 P5 col-permutation-1 cop-per2--col* **by** *blast*
}
then have ¬ *A B P CongA A B C* **by** *auto*
then have *A B P LtA A B C*
by (*simp add*: *LtA-def P2*)
thus *?thesis*
using *Acute-def assms*(*1*) **by** *blast*
qed

lemma *conga-inangle2-per--acute*:
assumes *Per A B C* **and**
P InAngle A B C **and**
P B A CongA P B C **and**
Q InAngle A B C
shows *Acute P B Q*
proof −
have *P1*: *P InAngle C B A*
using *assms*(*2*) *l11-24* **by** *auto*
have *P2*: *Q InAngle C B A*
using *assms*(*4*) *l11-24* **by** *blast*
have *P3*: *A* ≠ *B*
using *assms*(*3*) *conga-diff2* **by** *auto*
have *P5*: *P* ≠ *B*
using *assms*(*2*) *inangle-distincts* **by** *blast*
have *P7*: ¬ *Col A B C*
using *assms*(*1*) *assms*(*3*) *conga-distinct l8-9* **by** *blast*
have *P8*: *Acute A B P*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *conga-inangle-per--acute* **by** *auto*
{
assume *Col P B A*
then have *Col P B C*
using *assms*(*3*) *col-conga-col* **by** *blast*
then have *False*
using *Col-perm P5 P7* ‹*Col P B A*› *col-transitivity-2* **by** *blast*
}
then have *P9*: ¬ *Col P B A* **by** *auto*
have *P10*: ¬ *Col P B C*
using ‹*Col P B A* ⟹ *False*› *assms*(*3*) *ncol-conga-ncol* **by** *blast*
have *P11*: ¬ *Bet A B C*
using *P7 bet-col* **by** *blast*
show *?thesis*
proof *cases*
assume *Col B A Q*
then have *B Out A Q*
using *P11 assms*(*4*) *col-in-angle-out* **by** *auto*
thus *?thesis*
using *Out-cases P5 P8 acute-out2--acute acute-sym out-trivial* **by** *blast*
next
assume *S0*: ¬ *Col B A Q*
show *?thesis*

```
proof cases
  assume S1: Col B C Q
  then have B Out C Q
    using P11 P2 between-symmetry col-in-angle-out by blast
  then have S2: B Out Q C
    using l6-6 by blast
  have S3: B Out P P
    by (simp add: P5 out-trivial)
  have B Out A A
    by (simp add: P3 out-trivial)
  then have A B P CongA P B Q
    using S2 conga-left-comm l11-10 S3 assms(3) by blast
  thus ?thesis
    using P8 acute-conga--acute by blast
next
  assume S4: ¬ Col B C Q
  show ?thesis
  proof cases
    assume Col B P Q
    thus ?thesis
      using out--acute col-inangle2--out P11 assms(2) assms(4) by blast
  next
    assume S5: ¬ Col B P Q
    have S6: Coplanar B P A Q
        by (metis Col-perm P7 assms(2) assms(4) coplanar-trans-1 inangle--coplanar ncoplanar-perm-12 ncopla-
nar-perm-21)
    have S7: ¬ Col A B P
      using Col-cases P9 by auto
    have ¬ Col Q B P
      using Col-perm S5 by blast
    then have S8: B P TS A Q ∨ B P OS A Q
      using cop--one-or-two-sides S6 S7 by blast
    {
      assume S9: B P TS A Q
      have S10: Acute P B C
        using P8 acute-conga--acute acute-sym assms(3) by blast
      have Q InAngle P B C
      proof -
        have S11: P B OS C Q
          by (metis Col-perm OS-def P1 P10 P9 S9 in-angle-two-sides invert-two-sides l9-2)
        have C B OS P Q
          by (meson P1 P10 P2 P7 S4 inangle-one-side not-col-permutation-3 not-col-permutation-4)
        thus ?thesis
          by (simp add: S11 invert-one-side os2--inangle)
      qed
      then have P B Q LeA P B C
        by (simp add: inangle--lea)
      then have Acute P B Q
        using S10 acute-lea-acute by blast
    }
    {
      assume S12: B P OS A Q
      have B A OS P Q
        using Col-perm P7 S7 S0 assms(2) assms(4) inangle-one-side invert-one-side by blast
      then have Q InAngle P B A
        by (simp add: S12 os2--inangle)
      then have Q B P LeA P B A
        by (simp add: P3 P5 inangle1123 inangle2--lea)
      then have P B Q LeA A B P
        by (simp add: lea-comm)
      then have Acute P B Q
        using P8 acute-lea-acute by blast
    }
    thus ?thesis
      using ‹B P TS A Q ⟹ Acute P B Q› S8 by blast
  qed
```

qed
qed
qed

lemma *lta-os--ts*:
assumes
A O1 P LtA A O1 B **and**
O1 A OS B P
shows *O1 P TS A B*
proof −
have *A O1 P LeA A O1 B*
by (*simp add*: *assms*(*1*) *lta--lea*)
then have ∃ *P0*. *P0 InAngle A O1 B* ∧ *A O1 P CongA A O1 P0*
by (*simp add*: *LeA-def*)
then obtain *P′* **where** *P1*: *P′ InAngle A O1 B* ∧ *A O1 P CongA A O1 P′* **by** *blast*
have *P2*: ¬ *Col A O1 B*
using *assms*(*2*) *col123--nos not-col-permutation-4* **by** *blast*
obtain *R* **where** *P3*: *O1 A TS B R* ∧ *O1 A TS P R*
using *OS-def assms*(*2*) **by** *blast*
{
assume *Col B O1 P*
then have *Bet B O1 P*
by (*metis Tarski-neutral-dimensionless.out2--conga Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*2*) *between-trivial col-trivial-2 lta-not-conga one-side-chara or-bet-out out-trivial*)
then have *O1 A TS B P*
using *assms*(*2*) *col-trivial-1 one-side-chara* **by** *blast*
then have *P6*: ¬ *O1 A OS B P*
using *l9-9-bis* **by** *auto*
then have *False*
using *P6 assms*(*2*) **by** *auto*
}
then have *P4*: ¬ *Col B O1 P* **by** *auto*
thus *?thesis*
by (*meson P3 assms*(*1*) *inangle--lta l9-8-1 not-and-lta not-col-permutation-4 os-ts--inangle two-sides-cases*)
qed

lemma *bet--suppa*:
assumes *A* ≠ *B* **and**
B ≠ *C* **and**
B ≠ *A′* **and**
Bet A B A′
shows *A B C SuppA C B A′*
proof −
have *C B A′ CongA C B A′*
using *assms*(*2*) *assms*(*3*) *conga-refl* **by** *auto*
thus *?thesis* **using** *assms*(*4*) *assms*(*1*) *SuppA-def* **by** *auto*
qed

lemma *ex-suppa*:
assumes *A* ≠ *B* **and**
B ≠ *C*
shows ∃ *D E F*. *A B C SuppA D E F*
proof −
obtain *A′* **where** *Bet A B A′* ∧ *Cong B A′ A B*
using *segment-construction* **by** *blast*
thus *?thesis*
by (*meson assms*(*1*) *assms*(*2*) *bet--suppa point-construction-different*)
qed

lemma *suppa-distincts*:
assumes *A B C SuppA D E F*
shows *A* ≠ *B* ∧ *B* ≠ *C* ∧ *D* ≠ *E* ∧ *E* ≠ *F*
using *CongA-def SuppA-def assms* **by** *auto*

lemma *suppa-right-comm*:
assumes *A B C SuppA D E F*

```
  shows A B C SuppA F E D
  using SuppA-def assms conga-left-comm by auto

lemma suppa-left-comm:
  assumes A B C SuppA D E F
  shows C B A SuppA D E F
proof −
  obtain A′ where P1: Bet A B A′ ∧ D E F CongA C B A′
    using SuppA-def assms by auto
  obtain C′ where P2: Bet C B C′ ∧ Cong B C′ C B
    using segment-construction by blast
  then have C B A′ CongA A B C′
    by (metis Bet-cases P1 SuppA-def assms cong-diff-3 conga-diff45 conga-diff56 conga-left-comm l11-14)
  then have D E F CongA A B C′
    using P1 conga-trans by blast
  thus ?thesis
    by (metis CongA-def P1 P2 SuppA-def)
qed

lemma suppa-comm:
  assumes A B C SuppA D E F
  shows C B A SuppA F E D
  using assms suppa-left-comm suppa-right-comm by blast

lemma suppa-sym:
  assumes A B C SuppA D E F
  shows D E F SuppA A B C
proof −
  obtain A′ where P1: Bet A B A′ ∧ D E F CongA C B A′
    using SuppA-def assms by auto
  obtain D′ where P2: Bet D E D′ ∧ Cong E D′ D E
    using segment-construction by blast
  have A′ B C CongA D E F
    using P1 conga-right-comm not-conga-sym by blast
  then have A B C CongA F E D′
   by (metis P1 P2 Tarski-neutral-dimensionless.conga-right-comm Tarski-neutral-dimensionless.l11-13 Tarski-neutral-dimensionless.su
Tarski-neutral-dimensionless-axioms assms between-symmetry cong-diff-3)
  thus ?thesis
    by (metis CongA-def P1 P2 SuppA-def)
qed

lemma conga2-suppa--suppa:
  assumes A B C CongA A′ B′ C′ and
    D E F CongA D′ E′ F′ and
    A B C SuppA D E F
  shows A′ B′ C′ SuppA D′ E′ F′
proof −
  obtain A0 where P1: Bet A B A0 ∧ D E F CongA C B A0
    using SuppA-def assms(3) by auto
  then have A B C SuppA D′ E′ F′
   by (metis Tarski-neutral-dimensionless.SuppA-def Tarski-neutral-dimensionless-axioms assms(2) assms(3) conga-sym
conga-trans)
  then have P2: D′ E′ F′ SuppA A B C
    by (simp add: suppa-sym)
  then obtain D0 where P3: Bet D′ E′ D0 ∧ A B C CongA F′ E′ D0
    using P2 SuppA-def by auto
  have P5: A′ B′ C′ CongA F′ E′ D0
    using P3 assms(1) not-conga not-conga-sym by blast
  then have D′ E′ F′ SuppA A′ B′ C′
    using P2 P3 SuppA-def by auto
  thus ?thesis
    by (simp add: suppa-sym)
qed

lemma suppa2--conga456:
  assumes A B C SuppA D E F and
```

```
    A B C SuppA D' E' F'
  shows D E F CongA D' E' F'
proof -
  obtain A' where P1: Bet A B A' ∧ D E F CongA C B A'
    using SuppA-def assms(1) by auto
  obtain A'' where P2: Bet A B A'' ∧ D' E' F' CongA C B A''
    using SuppA-def assms(2) by auto
  have C B A' CongA C B A''
  proof -
    have P3: B Out C C using P1
      by (simp add: CongA-def out-trivial)
    have B Out A'' A' using P1 P2 l6-2
      by (metis assms(1) between-symmetry conga-distinct suppa-distincts)
    thus ?thesis
      by (simp add: P3 out2--conga)
  qed
  then have C B A' CongA D' E' F'
    using P2 not-conga not-conga-sym by blast
  thus ?thesis
    using P1 not-conga by blast
qed

lemma suppa2--conga123:
  assumes A B C SuppA D E F and
    A' B' C' SuppA D E F
  shows A B C CongA A' B' C'
  using assms(1) assms(2) suppa2--conga456 suppa-sym by blast

lemma bet-out--suppa:
  assumes A ≠ B and
    B ≠ C and
    Bet A B C and
    E Out D F
  shows A B C SuppA D E F
proof -
  have D E F CongA C B C
    using assms(2) assms(4) l11-21-b out-trivial by auto
  thus ?thesis
    using SuppA-def assms(1) assms(3) by blast
qed

lemma bet-suppa--out:
  assumes Bet A B C and
    A B C SuppA D E F
  shows E Out D F
proof -
  have A B C SuppA C B C
    using assms(1) assms(2) bet--suppa suppa-distincts by auto
  then have C B C CongA D E F
    using assms(2) suppa2--conga456 by auto
  thus ?thesis
    using eq-conga-out by auto
qed

lemma out-suppa--bet:
  assumes B Out A C and
    A B C SuppA D E F
  shows Bet D E F
proof -
  obtain B' where P1: Bet A B B' ∧ Cong B B' A B
    using segment-construction by blast
  have A B C SuppA A B B'
    by (metis P1 assms(1) assms(2) bet--suppa bet-cong-eq bet-out--bet suppa-distincts suppa-left-comm)
  then have A B B' CongA D E F
    using assms(2) suppa2--conga456 by auto
  thus ?thesis
```

using *P1 bet-conga--bet* **by** *blast*
qed

lemma *per-suppa--per*:
assumes *Per A B C* **and**
A B C SuppA D E F
shows *Per D E F*
proof −
obtain A' **where** *P1*: *Bet A B* A' ∧ *D E F CongA C B* A'
using *SuppA-def assms*(*2*) **by** *auto*
have *Per C B* A'
proof −
have *P2*: *A* ≠ *B*
using *assms*(*2*) *suppa-distincts* **by** *auto*
have *P3*: *Per C B A*
by (*simp add*: *assms*(*1*) *l8-2*)
have *Col B A* A'
using *P1 Col-cases Col-def* **by** *blast*
thus *?thesis*
by (*metis P2 P3 per-col*)
qed
thus *?thesis*
using *P1 l11-17 not-conga-sym* **by** *blast*
qed

lemma *per2--suppa*:
assumes *A* ≠ *B* **and**
B ≠ *C* **and**
D ≠ *E* **and**
E ≠ *F* **and**
Per A B C **and**
Per D E F
shows *A B C SuppA D E F*
proof −
obtain D' E' F' **where** *P1*: *A B C SuppA* D' E' F'
using *assms*(*1*) *assms*(*2*) *ex-suppa* **by** *blast*
have D' E' F' *CongA D E F*
using *P1 assms*(*3*) *assms*(*4*) *assms*(*5*) *assms*(*6*) *l11-16 per-suppa--per suppa-distincts* **by** *blast*
thus *?thesis*
by (*meson P1 conga2-suppa--suppa suppa2--conga123*)
qed

lemma *suppa--per*:
assumes *A B C SuppA A B C*
shows *Per A B C*
proof −
obtain A' **where** *P1*: *Bet A B* A' ∧ *A B C CongA C B* A'
using *SuppA-def assms* **by** *auto*
then have *C B A CongA C B* A'
by (*simp add*: *conga-left-comm*)
thus *?thesis*
using *P1 Per-perm l11-18-2* **by** *blast*
qed

lemma *acute-suppa--obtuse*:
assumes *Acute A B C* **and**
A B C SuppA D E F
shows *Obtuse D E F*
proof −
obtain A' **where** *P1*: *Bet A B* A' ∧ *D E F CongA C B* A'
using *SuppA-def assms*(*2*) **by** *auto*
then have *Obtuse C B* A'
by (*metis Tarski-neutral-dimensionless.obtuse-sym Tarski-neutral-dimensionless-axioms acute-bet--obtuse assms*(*1*) *conga-distinct*)
thus *?thesis*
by (*meson P1 Tarski-neutral-dimensionless.conga-obtuse--obtuse Tarski-neutral-dimensionless.not-conga-sym Tarski-neutral-dimensi

qed

lemma *obtuse-suppa--acute*:
assumes *Obtuse A B C* **and**
A B C SuppA D E F
shows *Acute D E F*
proof −
obtain *A'* **where** *P1*: *Bet A B A'* ∧ *D E F CongA C B A'*
using *SuppA-def assms*(*2*) **by** *auto*
then have *Acute C B A'*
using *acute-sym assms*(*1*) *bet-obtuse--acute conga-distinct* **by** *blast*
thus *?thesis*
using *P1 acute-conga--acute not-conga-sym* **by** *blast*
qed

lemma *lea-suppa2--lea*:
assumes *A B C SuppA A' B' C'* **and**
D E F SuppA D' E' F'
A B C LeA D E F
shows *D' E' F' LeA A' B' C'*
proof −
obtain *A0* **where** *P1*: *Bet A B A0* ∧ *A' B' C' CongA C B A0*
using *SuppA-def assms*(*1*) **by** *auto*
obtain *D0* **where** *P2*: *Bet D E D0* ∧ *D' E' F' CongA F E D0*
using *SuppA-def assms*(*2*) **by** *auto*
have *F E D0 LeA C B A0*
proof −
have *P3*: *D0* ≠ *E*
using *CongA-def P2* **by** *auto*
have *P4*: *A0* ≠ *B*
using *CongA-def P1* **by** *blast*
have *P6*: *Bet D0 E D*
by (*simp add*: *P2 between-symmetry*)
have *Bet A0 B A*
by (*simp add*: *P1 between-symmetry*)
thus *?thesis*
by (*metis P3 P4 P6 assms*(*3*) *l11-36-aux2 lea-comm lea-distincts*)
qed
thus *?thesis*
by (*meson P1 P2 Tarski-neutral-dimensionless.l11-30 Tarski-neutral-dimensionless.not-conga-sym Tarski-neutral-dimensionless-axio*
qed

lemma *lta-suppa2--lta*:
assumes *A B C SuppA A' B' C'*
and *D E F SuppA D' E' F'*
and *A B C LtA D E F*
shows *D' E' F' LtA A' B' C'*
proof −
obtain *A0* **where** *P1*: *Bet A B A0* ∧ *A' B' C' CongA C B A0*
using *SuppA-def assms*(*1*) **by** *auto*
obtain *D0* **where** *P2*: *Bet D E D0* ∧ *D' E' F' CongA F E D0*
using *SuppA-def assms*(*2*) **by** *auto*
have *F E D0 LtA C B A0*
proof −
have *P5*: *A0* ≠ *B*
using *CongA-def P1* **by** *blast*
have *D0* ≠ *E*
using *CongA-def P2* **by** *auto*
thus *?thesis*
using *assms*(*3*) *P1 P5 P2 bet2-lta--lta lta-comm* **by** *blast*
qed
thus *?thesis*
using *P1 P2 conga-preserves-lta not-conga-sym* **by** *blast*
qed

lemma *suppa-dec*:

A B C SuppA D E F ∨ ¬ *A B C SuppA D E F*
by *simp*

lemma *acute-one-side-aux*:
assumes *C A OS P B* **and**
Acute A C P **and**
C A Perp B C
shows *C B OS A P*
proof −
obtain *R* **where** *T1*: *C A TS P R* ∧ *C A TS B R*
using *OS-def assms*(*1*) **by** *blast*
obtain *A′ B′ C′* **where** *P1*: *Per A′ B′ C′* ∧ *A C P LtA A′ B′ C′*
using *Acute-def assms*(*2*) **by** *auto*
have *P2*: *Per A C B*
by (*simp add*: *assms*(*3*) *perp-per-1*)
then have *P3*: *A′ B′ C′ CongA A C B*
using *P1 assms*(*1*) *l11-16 lta-distincts os-distincts* **by** *blast*
have *P4*: *A C P LtA A C B*
by (*metis P2 acute-per--lta assms*(*1*) *assms*(*2*) *os-distincts*)
{
assume *P4A*: *Col P C B*
have *Per A C P*
proof −
have *P4B*: *C* ≠ *B*
using *assms*(*1*) *os-distincts* **by** *blast*
have *P4C*: *Per A C B*
by (*simp add*: *P2*)
have *Col C B P*
using *P4A Col-cases* **by** *auto*
thus *?thesis* **using** *per-col P4B P4C* **by** *blast*
qed
then have *False*
using *acute-not-per assms*(*2*) **by** *auto*
}
then have *P5*: ¬ *Col P C B* **by** *auto*
have *P6*: ¬ *Col A C P*
using *assms*(*1*) *col123--nos not-col-permutation-4* **by** *blast*
have *P7*: *C B TS A P* ∨ *C B OS A P*
using *P5 assms*(*1*) *not-col-permutation-4 os-ts1324--os two-sides-cases* **by** *blast*
{
assume *P8*: *C B TS A P*
then obtain *T* **where** *P9*: *Col T C B* ∧ *Bet A T P*
using *TS-def* **by** *blast*
then have *P10*: *C* ≠ *T*
using *Col-def P6 P9* **by** *auto*
have *T InAngle A C P*
by (*meson P4 P5 P8 Tarski-neutral-dimensionless.inangle--lta Tarski-neutral-dimensionless-axioms assms*(*1*) *not-and-lta not-col-permutation-3 os-ts--inangle*)
then have *C A OS T P*
by (*metis P10 P9 T1 TS-def col123--nos in-angle-one-side invert-one-side l6-16-1 one-side-reflexivity*)
then have *P13*: *C A OS T B*
using *assms*(*1*) *one-side-transitivity* **by** *blast*
have *C B OS A P*
by (*meson P4 Tarski-neutral-dimensionless.lta-os--ts Tarski-neutral-dimensionless-axioms assms*(*1*) *one-side-symmetry os-ts1324--os*)
}
thus *?thesis*
using *P7* **by** *blast*
qed

lemma *acute-one-side-aux0*:
assumes *Col A C P* **and**
Acute A C P **and**
C A Perp B C
shows *C B OS A P*
proof −

have *Per A C B*
by (*simp add*: *assms*(*3*) *perp-per-1*)
then have *P1*: *A C P LtA A C B*
using *Tarski-neutral-dimensionless.acute-per--lta Tarski-neutral-dimensionless-axioms acute-distincts assms*(*2*) *assms*(*3*) *perp-not-eq-2* **by** *fastforce*
have *P2*: *C Out A P*
using *acute-col--out assms*(*1*) *assms*(*2*) **by** *auto*
thus *?thesis*
using *Perp-cases assms*(*3*) *out-one-side perp-not-col* **by** *blast*
qed

lemma *acute-cop-perp--one-side*:
assumes *Acute A C P* **and**
C A Perp B C **and**
Coplanar A B C P
shows *C B OS A P*
proof *cases*
assume *Col A C P*
thus *?thesis*
by (*simp add*: *acute-one-side-aux0 assms*(*1*) *assms*(*2*))
next
assume *P1*: ¬ *Col A C P*
have *P2*: *C A TS P B* ∨ *C A OS P B*
using *Col-cases P1 assms*(*2*) *assms*(*3*) *cop-nos--ts coplanar-perm-13 perp-not-col* **by** *blast*
{
assume *P3*: *C A TS P B*
obtain *Bs* **where** *P4*: *C Midpoint B Bs*
using *symmetric-point-construction* **by** *auto*
have *C A TS Bs B*
by (*metis P3 P4 assms*(*2*) *bet--ts l9-2 midpoint-bet midpoint-distinct-2 perp-not-col ts-distincts*)
then have *P6*: *C A OS P Bs*
using *P3 l9-8-1* **by** *auto*
have *C Bs Perp A C*
proof −
have *P6A*: *C* ≠ *Bs*
using *P6 os-distincts* **by** *blast*
have *Col C B Bs*
using *Bet-cases Col-def P4 midpoint-bet* **by** *blast*
thus *?thesis*
using *Perp-cases P6A assms*(*2*) *perp-col* **by** *blast*
qed
then have *Bs C Perp C A*
using *Perp-perm* **by** *blast*
then have *C A Perp Bs C*
using *Perp-perm* **by** *blast*
then have *C B OS A P* **using** *acute-one-side-aux*
by (*metis P4 P6 assms*(*1*) *assms*(*2*) *col-one-side midpoint-col not-col-permutation-5 perp-distinct*)
}
{
assume *C A OS P B*
then have *C B OS A P* **using** *acute-one-side-aux*
using *assms*(*1*) *assms*(*2*) **by** *blast*
}
thus *?thesis*
using *P2* ‹*C A TS P B* ⟹ *C B OS A P*› **by** *auto*
qed

lemma *acute--not-obtuse*:
assumes *Acute A B C*
shows ¬ *Obtuse A B C*
using *acute-obtuse--lta assms nlta* **by** *blast*

3.10.2 Sum of angles

lemma *suma-distincts*:
assumes *A B C D E F SumA G H I*

shows *A* ≠ *B* ∧ *B* ≠*C* ∧ *D* ≠ *E* ∧ *E* ≠ *F* ∧ *G* ≠ *H* ∧ *H* ≠ *I*
proof −
obtain *J* **where** *C B J CongA D E F* ∧ ¬ *B C OS A J* ∧ *Coplanar A B C J* ∧ *A B J CongA G H I*
using *SumA-def assms* **by** *auto*
thus *?thesis*
using *CongA-def* **by** *blast*
qed

lemma *trisuma-distincts*:
assumes *A B C TriSumA D E F*
shows *A* ≠ *B* ∧ *B* ≠ *C* ∧ *A* ≠ *C* ∧ *D* ≠ *E* ∧ *E* ≠ *F*
proof −
obtain *G H I* **where** *A B C B C A SumA G H I* ∧ *G H I C A B SumA D E F*
using *TriSumA-def assms* **by** *auto*
thus *?thesis*
using *suma-distincts* **by** *blast*
qed

lemma *ex-suma*:
assumes *A* ≠ *B* **and**
B ≠ *C* **and**
D ≠ *E* **and**
E ≠ *F*
shows ∃ *G H I*. *A B C D E F SumA G H I*
proof −
have ∃ *I*. *A B C D E F SumA A B I*
proof *cases*
assume *P1*: *Col A B C*
obtain *J* **where** *P2*: *D E F CongA C B J* ∧ *Coplanar C B J A* **using** *angle-construction-4*
using *assms*(*2*) *assms*(*3*) *assms*(*4*) **by** *presburger*
have *P3*: *J* ≠ *B*
using *CongA-def P2* **by** *blast*
have ¬ *B C OS A J*
by (*metis P1 between-trivial2 one-side-chara*)
then have *A B C D E F SumA A B J*
by (*meson P2 P3 SumA-def assms*(*1*) *conga-refl ncoplanar-perm-15 not-conga-sym*)
thus *?thesis* **by** *blast*
next
assume *T1*: ¬ *Col A B C*
show *?thesis*
proof *cases*
assume *T2*: *Col D E F*
show *?thesis*
proof *cases*
assume *T3*: *Bet D E F*
obtain *J* **where** *T4*: *B Midpoint C J*
using *symmetric-point-construction* **by** *blast*
have *A B C D E F SumA A B J*
proof −
have *C B J CongA D E F*
by (*metis T3 T4 assms*(*2*) *assms*(*3*) *assms*(*4*) *conga-line midpoint-bet midpoint-distinct-2*)
moreover have ¬ *B C OS A J*
by (*simp add*: *T4 col124--nos midpoint-col*)
moreover have *Coplanar A B C J*
using *T3 bet--coplanar bet-conga--bet calculation*(*1*) *conga-sym ncoplanar-perm-15* **by** *blast*
moreover have *A B J CongA A B J*
using *CongA-def assms*(*1*) *calculation*(*1*) *conga-refl* **by** *auto*
ultimately show *?thesis*
using *SumA-def* **by** *blast*
qed
then show *?thesis*
by *auto*
next
assume *T5*: ¬ *Bet D E F*
have *A B C D E F SumA A B C*
proof −

have *E Out D F*
using *T2 T5 l6-4-2* **by** *auto*
then have *C B C CongA D E F*
using *assms(2) l11-21-b out-trivial* **by** *auto*
moreover have ¬ *B C OS A C*
using *os-distincts* **by** *blast*
moreover have *Coplanar A B C C*
using *ncop-distincts* **by** *auto*
moreover have *A B C CongA A B C*
using *assms(1) assms(2) conga-refl* **by** *auto*
ultimately show *?thesis*
using *SumA-def* **by** *blast*
qed
then show *?thesis*
by *auto*
qed
next
assume *T6*: ¬ *Col D E F*
then obtain *J* **where** *T7*: *D E F CongA C B J* ∧ *C B TS J A*
using *T1 ex-conga-ts not-col-permutation-4 not-col-permutation-5* **by** *presburger*
then show *?thesis*
proof −
have *C B J CongA D E F*
using *T7 not-conga-sym* **by** *blast*
moreover have ¬ *B C OS A J*
by (*simp add*: *T7 invert-two-sides l9-2 l9-9*)
moreover have *Coplanar A B C J*
using *T7 ncoplanar-perm-15 ts--coplanar* **by** *blast*
moreover have *A B J CongA A B J*
using *T7 assms(1) conga-diff56 conga-refl* **by** *blast*
ultimately show *?thesis*
using *SumA-def* **by** *blast*
qed
qed
qed
then show *?thesis*
by *auto*
qed

lemma *suma2--conga*:
assumes *A B C D E F SumA G H I* **and**
A B C D E F SumA G′ H′ I′
shows *G H I CongA G′ H′ I′*
proof −
obtain *J* **where** *P1*: *C B J CongA D E F* ∧ ¬ *B C OS A J* ∧ *Coplanar A B C J* ∧ *A B J CongA G H I*
using *SumA-def assms(1)* **by** *blast*
obtain *J′* **where** *P2*: *C B J′ CongA D E F* ∧ ¬ *B C OS A J′* ∧ *Coplanar A B C J′* ∧ *A B J′ CongA G′ H′ I′*
using *SumA-def assms(2)* **by** *blast*
have *P3*: *C B J CongA C B J′*
proof −
have *C B J CongA D E F*
by (*simp add*: *P1*)
moreover have *D E F CongA C B J′*
by (*simp add*: *P2 conga-sym*)
ultimately show *?thesis*
using *not-conga* **by** *blast*
qed
have *P4*: *A B J CongA A B J′*
proof *cases*
assume *P5*: *Col A B C*
then show *?thesis*
proof *cases*
assume *P6*: *Bet A B C*
show *?thesis*
proof −
have *C B J CongA C B J′*

by (*simp add*: *P3*)
moreover have *Bet C B A*
by (*simp add*: *P6 between-symmetry*)
moreover have $A \neq B$
using *assms*(*1*) *suma-distincts* **by** *blast*
ultimately show *?thesis*
using *l11-13* **by** *blast*
qed
next
assume *P7*: ¬ *Bet A B C*
moreover have *B Out A C*
by (*simp add*: *P5 calculation l6-4-2*)
moreover have $B \neq J$
using *CongA-def P3* **by** *blast*
then moreover have *B Out J J*
using *out-trivial* **by** *auto*
moreover have $B \neq J'$
using *CongA-def P3* **by** *blast*
then moreover have *B Out J′ J′*
using *out-trivial* **by** *auto*
ultimately show *?thesis*
using *P3 l11-10* **by** *blast*
qed
next
assume *P8*: ¬ *Col A B C*
show *?thesis*
proof *cases*
assume *P9*: *Col D E F*
have *B Out J′ J*
proof *cases*
assume *P10*: *Bet D E F*
show *?thesis*
proof −
have *D E F CongA J′ B C*
using *P2 conga-right-comm not-conga-sym* **by** *blast*
then have *Bet J′ B C*
using *P10 bet-conga--bet* **by** *blast*
moreover have *D E F CongA J B C*
by (*simp add*: *P1 conga-right-comm conga-sym*)
then moreover have *Bet J B C*
using *P10 bet-conga--bet* **by** *blast*
ultimately show *?thesis*
by (*metis CongA-def P3 l6-2*)
qed
next
assume *P11*: ¬ *Bet D E F*
have *P12*: *E Out D F*
by (*simp add*: *P11 P9 l6-4-2*)
show *?thesis*
proof −
have *B Out J′ C*
proof −
have *D E F CongA J′ B C*
using *P2 conga-right-comm conga-sym* **by** *blast*
then show *?thesis*
using *l11-21-a P12* **by** *blast*
qed
moreover have *B Out C J*
by (*metis P3 P8 bet-conga--bet calculation col-conga-col col-out2-col l6-4-2 l6-6 not-col-distincts not-conga-sym out-bet-out-1 out-trivial*)
ultimately show *?thesis*
using *l6-7* **by** *blast*
qed
qed
then show *?thesis*
using *P8 not-col-distincts out2--conga out-trivial* **by** *blast*

```
    next
      assume P13: ¬ Col D E F
      show ?thesis
      proof –
        have B C TS A J
        proof –
          have Coplanar B C A J
            using P1 coplanar-perm-8 by blast
          moreover have ¬ Col A B C
            by (simp add: P8)
          moreover have ¬ B C OS A J
            using P1 by simp
          moreover have ¬ Col J B C
          proof –
            have D E F CongA J B C
              using P1 conga-right-comm not-conga-sym by blast
            then show ?thesis
              using P13 ncol-conga-ncol by blast
          qed
          ultimately show ?thesis
            using cop--one-or-two-sides by blast
        qed
        moreover have B C TS A J′
        proof –
          have Coplanar B C A J′
            using P2 coplanar-perm-8 by blast
          moreover have ¬ Col A B C
            by (simp add: P8)
          moreover have ¬ B C OS A J′
            using P2 by simp
          moreover have ¬ Col J′ B C
          proof –
            have D E F CongA J′ B C
              using P2 conga-right-comm not-conga-sym by blast
            then show ?thesis
              using P13 ncol-conga-ncol by blast
          qed
          ultimately show ?thesis
            using cop-nos--ts by blast
        qed
        moreover have A B C CongA A B C
          by (metis P8 conga-pseudo-refl conga-right-comm not-col-distincts)
        moreover have C B J CongA C B J′
          by (simp add: P3)
        ultimately show ?thesis
          using l11-22a by blast
      qed
    qed
  qed
  then show ?thesis
    by (meson P1 P2 not-conga not-conga-sym)
qed

lemma suma-sym:
  assumes A B C D E F SumA G H I
  shows D E F A B C SumA G H I
proof –
  obtain J where P1: C B J CongA D E F ∧ ¬ B C OS A J ∧ Coplanar A B C J ∧ A B J CongA G H I
    using SumA-def assms(1) by blast
  show ?thesis
  proof cases
    assume P2: Col A B C
    then show ?thesis
    proof cases
      assume P3: Bet A B C
      obtain K where P4: Bet F E K ∧ Cong F E E K
```

```
      using Cong-perm segment-construction by blast
    show ?thesis
    proof -
      have P5: F E K CongA A B C
        by (metis CongA-def P1 P3 P4 cong-diff conga-line)
      moreover have ¬ E F OS D K
        using P4 bet-col col124--nos invert-one-side by blast
      moreover have Coplanar D E F K
        using P4 bet--coplanar ncoplanar-perm-15 by blast
      moreover have D E K CongA G H I
      proof -
        have D E K CongA A B J
        proof -
          have F E D CongA C B J
            by (simp add: P1 conga-left-comm conga-sym)
          moreover have Bet F E K
            by (simp add: P4)
          moreover have K ≠ E
            using P4 calculation(1) cong-identity conga-diff1 by blast
          moreover have Bet C B A
            by (simp add: Bet-perm P3)
          moreover have A ≠ B
            using CongA-def P5 by blast
          ultimately show ?thesis
            using conga-right-comm l11-13 not-conga-sym by blast
        qed
        then show ?thesis
          using P1 not-conga by blast
      qed
      ultimately show ?thesis
        using SumA-def by blast
    qed
  next
    assume T1: ¬ Bet A B C
    then have T2: B Out A C
      by (simp add: P2 l6-4-2)
    show ?thesis
    proof -
      have F E F CongA A B C
        by (metis T2 assms l11-21-b out-trivial suma-distincts)
      moreover have ¬ E F OS D F
        using os-distincts by auto
      moreover have Coplanar D E F F
        using ncop-distincts by auto
      moreover have D E F CongA G H I
      proof -
        have A B J CongA D E F
        proof -
          have C B J CongA D E F
            by (simp add: P1)
          moreover have B Out A C
            by (simp add: T2)
          moreover have J ≠ B
            using calculation(1) conga-distinct by auto
          moreover have D ≠ E
            using calculation(1) conga-distinct by blast
          moreover have F ≠ E
            using calculation(1) conga-distinct by blast
          ultimately show ?thesis
            by (meson Out-cases not-conga out2--conga out-trivial)
        qed
        then have D E F CongA A B J
          using not-conga-sym by blast
        then show ?thesis
          using P1 not-conga by blast
      qed
```

ultimately show *?thesis*
using *SumA-def* **by** *blast*
qed
qed
next
assume *Q1*: $\neg$ *Col A B C*
show *?thesis*
proof *cases*
assume *Q2*: *Col D E F*
obtain *K* **where** *Q3*: *A B C CongA F E K*
using *P1 angle-construction-3 conga-diff1 conga-diff56* **by** *fastforce*
show *?thesis*
proof −
have *F E K CongA A B C*
by (*simp add*: *Q3 conga-sym*)
moreover have $\neg$ *E F OS D K*
using *Col-cases Q2 one-side-not-col123* **by** *blast*
moreover have *Coplanar D E F K*
by (*simp add*: *Q2 col--coplanar*)
moreover have *D E K CongA G H I*
proof −
have *D E K CongA A B J*
proof *cases*
assume *Bet D E F*
then have *J B A CongA D E K*
by (*metis P1 bet-conga--bet calculation*(*1*) *conga-diff45 conga-right-comm l11-13 not-conga-sym*)
then show *?thesis*
using *conga-right-comm not-conga-sym* **by** *blast*
next
assume $\neg$ *Bet D E F*
then have *W2*: *E Out D F*
using *Q2 or-bet-out* **by** *blast*
have *A B J CongA D E K*
proof −
have *A B C CongA F E K*
by (*simp add*: *Q3*)
moreover have $A \neq B$
using *Q1 col-trivial-1* **by** *auto*
moreover have *E Out D F*
by (*simp add*: *W2*)
moreover have *B Out J C*
proof −
have *D E F CongA J B C*
by (*simp add*: *P1 conga-left-comm conga-sym*)
then show *?thesis*
using *W2 out-conga-out* **by** *blast*
qed
moreover have $K \neq E$
using *CongA-def Q3* **by** *blast*
ultimately show *?thesis*
using *l11-10 out-trivial* **by** *blast*
qed
then show *?thesis*
using *not-conga-sym* **by** *blast*
qed
then show *?thesis*
using *P1 not-conga* **by** *blast*
qed
ultimately show *?thesis*
using *SumA-def* **by** *blast*
qed
next
assume *W3*: $\neg$ *Col D E F*
then obtain *K* **where** *W4*: *A B C CongA F E K* $\wedge$ *F E TS K D*
using *Q1 ex-conga-ts not-col-permutation-3* **by** *blast*
show *?thesis*

```
proof −
  have F E K CongA A B C
    using W4 not-conga-sym by blast
  moreover have ¬ E F OS D K
  proof −
    have E F TS D K
      using W4 invert-two-sides l9-2 by blast
    then show ?thesis
      using l9-9 by auto
  qed
  moreover have Coplanar D E F K
  proof −
    have E F TS D K
      using W4 invert-two-sides l9-2 by blast
    then show ?thesis
      using ncoplanar-perm-8 ts--coplanar by blast
  qed
  moreover have D E K CongA G H I
  proof −
    have A B J CongA K E D
    proof −
      have B C TS A J
      proof −
        have Coplanar B C A J
          using P1 ncoplanar-perm-12 by blast
        moreover have ¬ Col A B C
          by (simp add: Q1)
        moreover have ¬ B C OS A J
          using P1 by simp
        moreover have ¬ Col J B C
        proof −
          {
            assume Col J B C
            have Col D E F
            proof −
              have Col C B J
                using Col-perm ‹Col J B C› by blast
              moreover have C B J CongA D E F
                by (simp add: P1)
              ultimately show ?thesis
                using col-conga-col by blast
            qed
            then have False
              by (simp add: W3)
          }
          then show ?thesis by blast
        qed
        ultimately show ?thesis
          using cop-nos--ts by blast
      qed
      moreover have E F TS K D
        using W4 invert-two-sides by blast
      moreover have A B C CongA K E F
        by (simp add: W4 conga-right-comm)
      moreover have C B J CongA F E D
        by (simp add: P1 conga-right-comm)
      ultimately show ?thesis
        using l11-22a by auto
    qed
    then have D E K CongA A B J
      using conga-left-comm not-conga-sym by blast
    then show ?thesis
      using P1 not-conga by blast
  qed
  ultimately show ?thesis
    using SumA-def by blast
```

```
      qed
    qed
  qed
qed

lemma conga3-suma--suma:
  assumes A B C D E F SumA G H I and
    A B C CongA A′ B′ C′ and
    D E F CongA D′ E′ F′ and
    G H I CongA G′ H′ I′
  shows A′ B′ C′ D′ E′ F′ SumA G′ H′ I′
proof −
  have D′ E′ F′ A B C SumA G′ H′ I′
  proof −
    obtain J where P1: C B J CongA D E F ∧ ¬ B C OS A J ∧ Coplanar A B C J ∧ A B J CongA G H I
      using SumA-def assms(1) by blast
    have A B C D′ E′ F′ SumA G′ H′ I′
    proof −
      have C B J CongA D′ E′ F′
        using P1 assms(3) not-conga by blast
      moreover have ¬ B C OS A J
        using P1 by simp
      moreover have Coplanar A B C J
        using P1 by simp
      moreover have A B J CongA G′ H′ I′
        using P1 assms(4) not-conga by blast
      ultimately show ?thesis
        using SumA-def by blast
    qed
    then show ?thesis
      by (simp add: suma-sym)
  qed
  then obtain J where P2: F′ E′ J CongA A B C ∧ ¬ E′ F′ OS D′ J ∧ Coplanar D′ E′ F′ J ∧ D′ E′ J CongA G′
H′ I′
    using SumA-def by blast
  have D′ E′ F′ A′ B′ C′ SumA G′ H′ I′
  proof −
    have F′ E′ J CongA A′ B′ C′
    proof −
      have F′ E′ J CongA A B C
        by (simp add: P2)
      moreover have A B C CongA A′ B′ C′
        by (simp add: assms(2))
      ultimately show ?thesis
        using not-conga by blast
    qed
    moreover have ¬ E′ F′ OS D′ J
      using P2 by simp
    moreover have Coplanar D′ E′ F′ J
      using P2 by simp
    moreover have D′ E′ J CongA G′ H′ I′
      by (simp add: P2)
    ultimately show ?thesis
      using SumA-def by blast
  qed
  then show ?thesis
    by (simp add: suma-sym)
qed

lemma out6-suma--suma:
  assumes A B C D E F SumA G H I and
    B Out A A′ and
    B Out C C′ and
    E Out D D′ and
    E Out F F′ and
    H Out G G′ and
```

H Out I I'
shows *A' B C' D' E F' SumA G' H I'*
proof −
have *A B C CongA A' B C'*
using *Out-cases assms(2) assms(3) out2--conga* **by** *blast*
moreover have *D E F CongA D' E F'*
using *Out-cases assms(4) assms(5) out2--conga* **by** *blast*
moreover have *G H I CongA G' H I'*
by (*simp add*: *assms(6) assms(7) l6-6 out2--conga*)
ultimately show *?thesis*
using *assms(1) conga3-suma--suma* **by** *blast*
qed

lemma *out546-suma--conga*:
assumes *A B C D E F SumA G H I* **and**
E Out D F
shows *A B C CongA G H I*
proof −
have *A B C D E F SumA A B C*
proof −
have *C B C CongA D E F*
by (*metis assms(1) assms(2) l11-21-b out-trivial suma-distincts*)
moreover have ¬ *B C OS A C*
using *os-distincts* **by** *auto*
moreover have *Coplanar A B C C*
using *ncop-distincts* **by** *auto*
moreover have *A B C CongA A B C*
by (*metis Tarski-neutral-dimensionless.suma-distincts Tarski-neutral-dimensionless-axioms assms(1) conga-pseudo-refl conga-right-comm*)
ultimately show *?thesis*
using *SumA-def* **by** *blast*
qed
then show *?thesis* **using** *suma2--conga assms(1)* **by** *blast*
qed

lemma *out546--suma*:
assumes $A \neq B$ **and**
$B \neq C$ **and**
E Out D F
shows *A B C D E F SumA A B C*
proof −
have *P1*: $D \neq E$
using *assms(3) out-diff1* **by** *auto*
have *P2*: $F \neq E$
using *Out-def assms(3)* **by** *auto*
then obtain *G H I* **where** *P3*: *A B C D E F SumA G H I*
using *P1 assms(1) assms(2) ex-suma* **by** *presburger*
then have *G H I CongA A B C*
by (*meson Tarski-neutral-dimensionless.conga-sym Tarski-neutral-dimensionless.out546-suma--conga Tarski-neutral-dimensionless-a assms(3)*)
then show *?thesis*
using *P1 P2 P3 assms(1) assms(2) assms(3) conga3-suma--suma conga-refl out-diff1* **by** *auto*
qed

lemma *out213-suma--conga*:
assumes *A B C D E F SumA G H I* **and**
B Out A C
shows *D E F CongA G H I*
using *assms(1) assms(2) out546-suma--conga suma-sym* **by** *blast*

lemma *out213--suma*:
assumes $D \neq E$ **and**
$E \neq F$ **and**
B Out A C
shows *A B C D E F SumA D E F*
by (*simp add*: *assms(1) assms(2) assms(3) out546--suma suma-sym*)

lemma *suma-left-comm*:
  assumes *A B C D E F SumA G H I*
  shows *C B A D E F SumA G H I*
proof −
  have *A B C CongA C B A*
    using *assms conga-pseudo-refl suma-distincts* **by** *fastforce*
  moreover have *D E F CongA D E F*
    by (*metis assms conga-refl suma-distincts*)
  moreover have *G H I CongA G H I*
    by (*metis assms conga-refl suma-distincts*)
  ultimately show *?thesis*
    using *assms conga3-suma--suma* **by** *blast*
qed

lemma *suma-middle-comm*:
  assumes *A B C D E F SumA G H I*
  shows *A B C F E D SumA G H I*
  using *assms suma-left-comm suma-sym* **by** *blast*

lemma *suma-right-comm*:
  assumes *A B C D E F SumA G H I*
  shows *A B C D E F SumA I H G*
proof −
  have *A B C CongA A B C*
    using *assms conga-refl suma-distincts* **by** *fastforce*
  moreover have *D E F CongA D E F*
    by (*metis assms conga-refl suma-distincts*)
  moreover have *G H I CongA I H G*
   by (*meson Tarski-neutral-dimensionless.conga-right-comm Tarski-neutral-dimensionless.suma2--conga Tarski-neutral-dimensionless-assms*)
  ultimately show *?thesis*
    using *assms conga3-suma--suma* **by** *blast*
qed

lemma *suma-comm*:
  assumes *A B C D E F SumA G H I*
  shows *C B A F E D SumA I H G*
  by (*simp add*: *assms suma-left-comm suma-middle-comm suma-right-comm*)

lemma *ts--suma*:
  assumes *A B TS C D*
  shows *C B A A B D SumA C B D*
proof −
  have *A B D CongA A B D*
   by (*metis Tarski-neutral-dimensionless.conga-right-comm Tarski-neutral-dimensionless-axioms assms conga-pseudo-refl ts-distincts*)
  moreover have ¬ *B A OS C D*
    using *assms invert-one-side l9-9* **by** *blast*
  moreover have *Coplanar C B A D*
    using *assms ncoplanar-perm-14 ts--coplanar* **by** *blast*
  moreover have *C B D CongA C B D*
    by (*metis assms conga-refl ts-distincts*)
  ultimately show *?thesis*
    using *SumA-def* **by** *blast*
qed

lemma *ts--suma-1*:
  assumes *A B TS C D*
  shows *C A B B A D SumA C A D*
  by (*simp add*: *assms invert-two-sides ts--suma*)

lemma *inangle--suma*:
  assumes *P InAngle A B C*
  shows *A B P P B C SumA A B C*
proof −

```
  have Coplanar A B P C
    by (simp add: assms coplanar-perm-8 inangle--coplanar)
  moreover have ¬ B P OS A C
    by (meson assms col123--nos col124--nos in-angle-two-sides invert-two-sides l9-9-bis not-col-permutation-5)
  ultimately show ?thesis
    using SumA-def assms conga-refl inangle-distincts by blast
qed

lemma bet--suma:
  assumes A ≠ B and
    B ≠ C and
    P ≠ B and Bet A B C
  shows A B P P B C SumA A B C
proof −
  have P InAngle A B C
    using assms(1) assms(2) assms(3) assms(4) in-angle-line by auto
  then show ?thesis
    by (simp add: inangle--suma)
qed

lemma sams-chara:
  assumes A ≠ B and
    A′ ≠ B and
    Bet A B A′
  shows SAMS A B C D E F ⟷ D E F LeA C B A′
proof −
  {
    assume T1: SAMS A B C D E F
    obtain J where T2: C B J CongA D E F ∧ ¬ B C OS A J ∧ ¬ A B TS C J ∧ Coplanar A B C J
      using SAMS-def T1 by auto
    have T3: A ≠ A′
      using assms(2) assms(3) between-identity by blast
    have T4: C ≠ B
      using T2 conga-distinct by blast
    have T5: J ≠ B
      using T2 conga-diff2 by blast
    have T6: D ≠ E
      using CongA-def T2 by auto
    have T7: F ≠ E
      using CongA-def T2 by blast
    {
      assume E Out D F
      then have D E F LeA C B A′
        by (simp add: T4 assms(2) l11-31-1)
    }
    {
      assume T8: ¬ Bet A B C
      have D E F LeA C B A′
      proof cases
        assume Col A B C
        then have Bet C B A′
          using T8 assms(1) assms(3) between-exchange3 outer-transitivity-between2 third-point by blast
        then show ?thesis
          by (simp add: T4 T6 T7 assms(2) l11-31-2)
      next
        assume T9: ¬ Col A B C
        show ?thesis
        proof cases
          assume T10: Col D E F
          show ?thesis
          proof cases
            assume T11: Bet D E F
            have D E F CongA C B J
              by (simp add: T2 conga-sym)
            then have T12: Bet C B J
              using T11 bet-conga--bet by blast
```

have *A B TS C J*
proof −
have ¬ *Col J A B*
using *T5 T9 T12 bet-col col2--eq col-permutation-1* **by** *blast*
moreover have ∃ *T. Col T A B* ∧ *Bet C T J*
using *T12 col-trivial-3* **by** *blast*
ultimately show *?thesis*
using *T9 TS-def col-permutation-1* **by** *blast*
qed
then have *False*
using *T2* **by** *simp*
then show *?thesis* **by** *simp*
next
assume ¬ *Bet D E F*
then show *?thesis*
using *T10* ‹*E Out D F* ⟹ *D E F LeA C B A*′› *or-bet-out* **by** *auto*
qed
next
assume *T13*: ¬ *Col D E F*
show *?thesis*
proof −
have *C B J LeA C B A*′
proof −
have *J InAngle C B A*′
proof −
have *A*′ ≠ *B*
by (*simp add*: *assms*(*2*))
moreover have *Bet A B A*′
by (*simp add*: *assms*(*3*))
moreover have *C InAngle A B J*
proof −
have ¬ *Col J B C*
proof −
have ¬ *Col D E F*
by (*simp add*: *T13*)
moreover have *D E F CongA J B C*
using *T2 conga-left-comm not-conga-sym* **by** *blast*
ultimately show *?thesis*
using *ncol-conga-ncol* **by** *blast*
qed
then have *B C TS A J*
by (*simp add*: *T2 T9 cop-nos--ts coplanar-perm-8*)
then obtain *X* **where** *T14*: *Col X B C* ∧ *Bet A X J*
using *TS-def* **by** *blast*
{
assume *T15*: *X* ≠ *B*
have *B Out X C*
proof −
have *Col B X C*
by (*simp add*: *Col-perm T14*)
moreover have *B A OS X C*
proof −
have *A B OS X C*
proof −
have *A B OS X J*
by (*smt T14 T9 T15 bet-out calculation col-transitivity-2 col-trivial-2 l6-21 out-one-side*)
moreover have *A B OS J C*
by (*metis T14 T2 T9 calculation cop-nts--os l5-2 not-col-permutation-2 one-side-chara one-side-symmetry*)
ultimately show *?thesis*
using *one-side-transitivity* **by** *blast*
qed
then show *?thesis*
by (*simp add*: *invert-one-side*)
qed
ultimately show *?thesis*

```
                using col-one-side-out by auto
              qed
            }
            then have Bet A X J ∧ (X = B ∨ B Out X C)
              using T14 by blast
            then show ?thesis
              using InAngle-def T4 T5 assms(1) by auto
          qed
          ultimately show ?thesis
            using in-angle-reverse l11-24 by blast
        qed
        moreover have C B J CongA C B J
          by (simp add: T4 T5 conga-refl)
        ultimately show ?thesis
          by (simp add: inangle--lea)
      qed
      moreover have D E F LeA C B J
        by (simp add: T2 conga--lea456123)
      ultimately show ?thesis
        using lea-trans by blast
    qed
  qed
qed
}
then have D E F LeA C B A′
  using SAMS-def T1 ‹E Out D F ⟹ D E F LeA C B A′› by blast
}
{
  assume P1: D E F LeA C B A′
  have P2: A ≠ A′
    using assms(2) assms(3) between-identity by blast
  have P3: C ≠ B
    using P1 lea-distincts by auto
  have P4: D ≠ E
    using P1 lea-distincts by auto
  have P5: F ≠ E
    using P1 lea-distincts by auto
  have SAMS A B C D E F
  proof cases
    assume P6: Col A B C
    show ?thesis
    proof cases
      assume P7: Bet A B C
      have E Out D F
      proof −
        have B Out C A′
          by (meson Bet-perm P3 P7 assms(1) assms(2) assms(3) l6-2)
        moreover have C B A′ CongA D E F
          using P1 calculation l11-21-b out-lea--out by blast
        ultimately show ?thesis
          using out-conga-out by blast
      qed
      moreover have C B C CongA D E F
        using P3 calculation l11-21-b out-trivial by auto
      moreover have ¬ B C OS A C
        using os-distincts by auto
      moreover have ¬ A B TS C C
        by (simp add: not-two-sides-id)
      moreover have Coplanar A B C C
        using ncop-distincts by auto
      ultimately show ?thesis
        using SAMS-def assms(1) by blast
    next
      assume P8: ¬ Bet A B C
      have P9: B Out A C
        by (simp add: P6 P8 l6-4-2)
```

```
      obtain J where P10: D E F CongA C B J
        using P3 P4 P5 angle-construction-3 by blast
      show ?thesis
      proof -
        have C B J CongA D E F
          using P10 not-conga-sym by blast
        moreover have ¬ B C OS A J
          using Col-cases P6 one-side-not-col123 by blast
        moreover have ¬ A B TS C J
          using Col-cases P6 TS-def by blast
        moreover have Coplanar A B C J
          using P6 col--coplanar by auto
        ultimately show ?thesis
          using P8 SAMS-def assms(1) by blast
      qed
    qed
  next
    assume P11: ¬ Col A B C
    have P12: ¬ Col A' B C
      using P11 assms(2) assms(3) bet-col bet-col1 colx by blast
    show ?thesis
    proof cases
      assume P13: Col D E F
      have P14: E Out D F
      proof -
        {
          assume P14: Bet D E F
          have D E F LeA C B A'
            by (simp add: P1)
          then have Bet C B A'
            using P14 bet-lea--bet by blast
          then have Col A' B C
            using Col-def Col-perm by blast
          then have False
            by (simp add: P12)
        }
        then have ¬ Bet D E F by auto
        then show ?thesis
          by (simp add: P13 l6-4-2)
      qed
      show ?thesis
      proof -
        have C B C CongA D E F
          by (simp add: P3 P14 l11-21-b out-trivial)
        moreover have ¬ B C OS A C
          using os-distincts by auto
        moreover have ¬ A B TS C C
          by (simp add: not-two-sides-id)
        moreover have Coplanar A B C C
          using ncop-distincts by auto
        ultimately show ?thesis
          using P14 SAMS-def assms(1) by blast
      qed
    next
      assume P15: ¬ Col D E F
      obtain J where P16: D E F CongA C B J ∧ C B TS J A
        using P11 P15 ex-conga-ts not-col-permutation-3 by presburger
      show ?thesis
      proof -
        have C B J CongA D E F
          by (simp add: P16 conga-sym)
        moreover have ¬ B C OS A J
        proof -
          have C B TS A J
            using P16 by (simp add: l9-2)
          then show ?thesis
```

using *invert-one-side l9-9* **by** *blast*
qed
moreover have ¬ *A B TS C J* ∧ *Coplanar A B C J*
proof *cases*
assume *Col A B J*
then show *?thesis*
using *TS-def ncop--ncols not-col-permutation-1* **by** *blast*
next
assume *P17*: ¬ *Col A B J*
have ¬ *A B TS C J*
proof −
have *A′ B OS J C*
proof −
have ¬ *Col A′ B C*
by (*simp add*: *P12*)
moreover have ¬ *Col B A′ J*
proof −
{
assume *Col B A′ J*
then have *False*
by (*metis P17 assms*(*2*) *assms*(*3*) *bet-col col-trivial-2 colx*)
}
then show *?thesis* **by** *auto*
qed
moreover have *J InAngle A′ B C*
proof −
obtain *K* **where** *P20*: *K InAngle C B A′* ∧ *D E F CongA C B K*
using *LeA-def P1* **by** *blast*
have *J InAngle C B A′*
proof −
have *C B A′ CongA C B A′*
by (*simp add*: *P3 assms*(*2*) *conga-pseudo-refl conga-right-comm*)
moreover have *C B K CongA C B J*
proof −
have *C B K CongA D E F*
using *P20 not-conga-sym* **by** *blast*
moreover have *D E F CongA C B J*
by (*simp add*: *P16*)
ultimately show *?thesis*
using *not-conga* **by** *blast*
qed
moreover have *K InAngle C B A′*
using *P20* **by** *simp*
moreover have *C B OS J A′*
proof −
have *C B TS J A* **using** *P16*
by *simp*
moreover have *C B TS A′ A*
using *Col-perm P12 assms*(*3*) *bet--ts between-symmetry calculation invert-two-sides ts-distincts* **by** *blast*
ultimately show *?thesis*
using *OS-def* **by** *auto*
qed
ultimately show *?thesis*
using *conga-preserves-in-angle* **by** *blast*
qed
then show *?thesis*
by (*simp add*: *l11-24*)
qed
ultimately show *?thesis*
by (*simp add*: *in-angle-one-side*)
qed
then have *A′ B OS C J*
by (*simp add*: *one-side-symmetry*)
then have ¬ *A′ B TS C J*
by (*simp add*: *l9-9-bis*)

```
          then show ?thesis
            using assms(2) assms(3) bet-col bet-col1 col-preserves-two-sides by blast
        qed
        moreover have Coplanar A B C J
        proof -
          have C B TS J A
            using P16 by simp
          then show ?thesis
            by (simp add: coplanar-perm-20 ts--coplanar)
        qed
        ultimately  show ?thesis by auto
      qed
      ultimately show ?thesis
        using P11 SAMS-def assms(1) bet-col by auto
    qed
  qed
 qed
}
then show ?thesis
  using ‹SAMS A B C D E F ⟹ D E F LeA C B A'› by blast
qed

lemma sams-distincts:
  assumes SAMS A B C D E F
  shows A ≠ B ∧ B ≠ C ∧ D ≠ E ∧ E ≠ F
proof -
  obtain J where P1: C B J CongA D E F ∧ ¬ B C OS A J ∧ ¬ A B TS C J ∧ Coplanar A B C J
    using SAMS-def assms by auto
  then show ?thesis
    by (metis SAMS-def assms conga-distinct)
qed

lemma sams-sym:
  assumes SAMS A B C D E F
  shows SAMS D E F A B C
proof -
  have P1: A ≠ B
    using assms sams-distincts by blast
  have P3: D ≠ E
    using assms sams-distincts by blast
  obtain D' where P5: E Midpoint D D'
    using symmetric-point-construction by blast
  obtain A' where P6: B Midpoint A A'
    using symmetric-point-construction by blast
  have P8: E ≠ D'
    using P3 P5 is-midpoint-id-2 by blast
  have P9: A ≠ A'
    using P1 P6 l7-3 by auto
  then have P10: B ≠ A'
    using P6 P9 midpoint-not-midpoint by auto
  then have D E F LeA C B A'
    using P1 P6 assms midpoint-bet sams-chara by fastforce
  then have D E F LeA A' B C
    by (simp add: lea-right-comm)
  then have A B C LeA D' E F
    by (metis Mid-cases P1 P10 P3 P5 P6 P8 l11-36 midpoint-bet)
  then have A B C LeA F E D'
    by (simp add: lea-right-comm)
  moreover have D ≠ E
    by (simp add: P3)
  moreover have D' ≠ E
    using P8 by auto
  moreover have Bet D E D'
    by (simp add: P5 midpoint-bet)
  then show ?thesis
    using P3 P8 calculation(1) sams-chara by fastforce
```

```
qed

lemma sams-right-comm:
  assumes SAMS A B C D E F
  shows SAMS A B C F E D
proof -
  have P1: E Out D F ∨ ¬ Bet A B C
    using SAMS-def assms by blast
  obtain J where P2: C B J CongA D E F ∧ ¬ B C OS A J ∧ ¬ A B TS C J ∧ Coplanar A B C J
    using SAMS-def assms by auto
  {
    assume E Out D F
    then have E Out F D ∨ ¬ Bet A B C
      by (simp add: l6-6)
  }
  {
    assume ¬ Bet A B C
    then have E Out F D ∨ ¬ Bet A B C by auto
  }
  then have E Out F D ∨ ¬ Bet A B C
    using ‹E Out D F ⟹ E Out F D ∨ ¬ Bet A B C› P1 by auto
  moreover have C B J CongA F E D
  proof -
    have C B J CongA D E F
      by (simp add: P2)
    then show ?thesis
      by (simp add: conga-right-comm)
  qed
  ultimately show ?thesis
    using P2 SAMS-def assms by auto
qed

lemma sams-left-comm:
  assumes SAMS A B C D E F
  shows SAMS C B A D E F
proof -
  have SAMS D E F A B C
    by (simp add: assms sams-sym)
  then have SAMS D E F C B A
    using sams-right-comm by blast
  then show ?thesis
    using sams-sym by blast
qed

lemma sams-comm:
  assumes SAMS A B C D E F
  shows SAMS C B A F E D
  using assms sams-left-comm sams-right-comm by blast

lemma conga2-sams--sams:
  assumes A B C CongA A' B' C' and
    D E F CongA D' E' F' and
    SAMS A B C D E F
  shows SAMS A' B' C' D' E' F'
proof -
  obtain A0 where B Midpoint A A0
    using symmetric-point-construction by auto
  obtain A'0 where B' Midpoint A' A'0
    using symmetric-point-construction by blast
  have D' E' F' LeA C' B' A'0
  proof -
    have D E F LeA C B A0
      by (metis ‹B Midpoint A A0› assms(1) assms(3) conga-distinct midpoint-bet midpoint-distinct-2 sams-chara)
    moreover have D E F CongA D' E' F'
      by (simp add: assms(2))
    moreover have C B A0 CongA C' B' A'0
```

proof −
have *A0 B C CongA A′0 B′ C′*
by (*metis* ‹*B Midpoint A A0*› ‹*B′ Midpoint A′ A′0*› *assms*(*1*) *calculation*(*1*) *conga-diff45 l11-13 lea-distincts midpoint-bet midpoint-not-midpoint*)
then show *?thesis*
using *conga-comm* **by** *blast*
qed
ultimately show *?thesis*
using *l11-30* **by** *blast*
qed
then show *?thesis*
by (*metis* ‹*B′ Midpoint A′ A′0*› *assms*(*1*) *conga-distinct lea-distincts midpoint-bet sams-chara*)
qed

lemma *out546--sams*:
assumes *A* ≠ *B* **and**
B ≠ *C* **and**
E Out D F
shows *SAMS A B C D E F*
proof −
obtain *A′* **where** *Bet A B A′* ∧ *Cong B A′ A B*
using *segment-construction* **by** *blast*
then have *D E F LeA C B A′*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *cong-diff-3 l11-31-1* **by** *fastforce*
then show *?thesis*
using ‹*Bet A B A′* ∧ *Cong B A′ A B*› *assms*(*1*) *lea-distincts sams-chara* **by** *blast*
qed

lemma *out213--sams*:
assumes *D* ≠ *E* **and**
E ≠ *F* **and**
B Out A C
shows *SAMS A B C D E F*
by (*simp add*: *Tarski-neutral-dimensionless.sams-sym Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*2*) *assms*(*3*) *out546--sams*)

lemma *bet-suma--sams*:
assumes *A B C D E F SumA G H I* **and**
Bet G H I
shows *SAMS A B C D E F*
proof −
obtain *A′* **where** *P1*: *C B A′ CongA D E F* ∧ ¬ *B C OS A A′* ∧ *Coplanar A B C A′* ∧ *A B A′ CongA G H I*
using *SumA-def assms*(*1*) **by** *auto*
then have *G H I CongA A B A′*
using *not-conga-sym* **by** *blast*
then have *Bet A B A′*
using *assms*(*2*) *bet-conga--bet* **by** *blast*
show *?thesis*
proof −
have *E Out D F* ∨ ¬ *Bet A B C*
proof −
{
assume *Bet A B C*
then have *E Out D F*
proof −
have *B Out C A′*
proof −
have *C* ≠ *B*
using *assms*(*1*) *suma-distincts* **by** *blast*
moreover have *A′* ≠ *B*
using *CongA-def* ‹*G H I CongA A B A′*› **by** *blast*
moreover have *A* ≠ *B*
using *CongA-def* ‹*G H I CongA A B A′*› **by** *blast*
moreover have *Bet C B A*
by (*simp add*: *Bet-perm* ‹*Bet A B C*›)
ultimately show *?thesis*

          using *Out-def* ‹*Bet A B A'*› ‹*Bet A B C*› *l5-2* **by** *auto*
        qed
        moreover have *C B A' CongA D E F*
          using *P1* **by** *simp*
        ultimately show *?thesis*
          using *l11-21-a* **by** *blast*
      qed
    }
    then show *?thesis*
      by *blast*
  qed
  moreover have ∃ *J*. (*C B J CongA D E F* ∧ ¬ *B C OS A J* ∧ ¬ *A B TS C J* ∧ *Coplanar A B C J*)
  proof −
    have *C B A' CongA D E F*
      by (*simp add*: *P1*)
    moreover have ¬ *B C OS A A'*
      by (*simp add*: *P1*)
    moreover have ¬ *A B TS C A'*
      using *Col-def TS-def* ‹*Bet A B A'*› **by** *blast*
    moreover have *Coplanar A B C A'*
      by (*simp add*: *P1*)
    ultimately show *?thesis*
      by *blast*
  qed
  ultimately show *?thesis*
    using *CongA-def SAMS-def* ‹*C B A' CongA D E F* ∧ ¬ *B C OS A A'* ∧ *Coplanar A B C A'* ∧ *A B A' CongA G H I*› **by** *auto*
 qed
qed

lemma *bet--sams*:
 assumes *A* ≠ *B* **and**
  B ≠ *C* **and**
  P ≠ *B* **and**
  Bet A B C
 shows *SAMS A B P P B C*
 by (*meson assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *bet--suma bet-suma--sams*)

lemma *suppa--sams*:
 assumes *A B C SuppA D E F*
 shows *SAMS A B C D E F*
proof −
 obtain *A'* **where** *P1*: *Bet A B A'* ∧ *D E F CongA C B A'*
  using *SuppA-def assms* **by** *auto*
 then have *SAMS A B C C B A'*
  by (*metis assms bet--sams conga-diff45 conga-diff56 suppa2--conga123*)
 thus *?thesis*
  by (*meson P1 assms conga2-sams--sams not-conga-sym suppa2--conga123*)
qed

lemma *os-ts--sams*:
 assumes *B P TS A C* **and**
  A B OS P C
 shows *SAMS A B P P B C*
proof −
 have *B Out P C* ∨ ¬ *Bet A B P*
  using *assms*(*2*) *bet-col col123--nos* **by** *blast*
 moreover have ∃ *J*. (*P B J CongA P B C* ∧ ¬ *B P OS A J* ∧ ¬ *A B TS P J* ∧ *Coplanar A B P J*)
  by (*metis assms*(*1*) *assms*(*2*) *conga-refl l9-9 os--coplanar os-distincts*)
 ultimately show *?thesis*
  using *SAMS-def assms*(*2*) *os-distincts* **by** *auto*
qed

lemma *os2--sams*:
 assumes *A B OS P C* **and**
  C B OS P A

shows *SAMS A B P P B C*
by (*simp add*: *Tarski-neutral-dimensionless.os-ts--sams Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*2*) *invert-one-side l9-31*)

lemma *inangle--sams*:
assumes *P InAngle A B C*
shows *SAMS A B P P B C*
proof −
have *Bet A B C* ∨ *B Out A C* ∨ ¬ *Col A B C*
using *l6-4-2* **by** *blast*
{
assume *Bet A B C*
then have *SAMS A B P P B C*
using *assms bet--sams inangle-distincts* **by** *fastforce*
}
{
assume *B Out A C*
then have *SAMS A B P P B C*
by (*metis assms in-angle-out inangle-distincts out213--sams*)
}
{
assume ¬ *Col A B C*
then have ¬ *Bet A B C*
using *Col-def* **by** *auto*
{
assume *Col B A P*
have *SAMS A B P P B C*
by (*metis* ‹*Col B A P*› ‹¬ *Bet A B C*› *assms col-in-angle-out inangle-distincts out213--sams*)
}
{
assume ¬ *Col B A P*
{
assume *Col B C P*
have *SAMS A B P P B C*
by (*metis Tarski-neutral-dimensionless.sams-comm Tarski-neutral-dimensionless-axioms* ‹*Col B C P*› ‹¬ *Bet A B C*› *assms between-symmetry col-in-angle-out inangle-distincts l11-24 out546--sams*)
}
{
assume ¬ *Col B C P*
have *SAMS A B P P B C*
proof −
have *B P TS A C*
by (*simp add*: ‹¬ *Col B A P*› ‹¬ *Col B C P*› *assms in-angle-two-sides invert-two-sides*)
moreover have *A B OS P C*
by (*simp add*: ‹¬ *Col A B C*› ‹¬ *Col B A P*› *assms in-angle-one-side*)
ultimately show *?thesis*
by (*simp add*: *os-ts--sams*)
qed
}
then have *SAMS A B P P B C*
using ‹*Col B C P* ⟹ *SAMS A B P P B C*› **by** *blast*
}
then have *SAMS A B P P B C*
using ‹*Col B A P* ⟹ *SAMS A B P P B C*› **by** *blast*
}
thus *?thesis*
using ‹*B Out A C* ⟹ *SAMS A B P P B C*› ‹*Bet A B C* ⟹ *SAMS A B P P B C*› ‹*Bet A B C* ∨ *B Out A C* ∨ ¬ *Col A B C*› **by** *blast*
qed

lemma *sams-suma--lea123789*:
assumes *A B C D E F SumA G H I* **and**
SAMS A B C D E F
shows *A B C LeA G H I*
proof *cases*
assume *Col A B C*

show *?thesis*
proof *cases*
assume *Bet A B C*
have *P1*: ($A \neq B \wedge$ (*E Out D F* $\vee \neg$ *Bet A B C*)) $\wedge$ ($\exists$ *J*. (*C B J CongA D E F* $\wedge \neg$ (*B C OS A J*) $\wedge \neg$ (*A B TS C J*) $\wedge$ *Coplanar A B C J*))
using *SAMS-def assms*(*2*) **by** *auto*
{
assume *E Out D F*
then have *A B C CongA G H I*
using *assms*(*1*) *out546-suma--conga* **by** *auto*
then have *A B C LeA G H I*
by (*simp add*: *conga--lea*)
}
thus *?thesis*
using *P1* ‹*Bet A B C*› **by** *blast*
next
assume ¬ *Bet A B C*
then have *B Out A C*
using ‹*Col A B C*› *or-bet-out* **by** *auto*
thus *?thesis*
by (*metis assms*(*1*) *l11-31-1 suma-distincts*)
qed
next
assume ¬ *Col A B C*
show *?thesis*
proof *cases*
assume *Col D E F*
show *?thesis*
proof *cases*
assume *Bet D E F*
have *SAMS D E F A B C*
using *assms*(*2*) *sams-sym* **by** *auto*
then have *B Out A C*
using *SAMS-def* ‹*Bet D E F*› **by** *blast*
thus *?thesis* **using** *l11-31-1*
by (*metis assms*(*1*) *suma-distincts*)
next
assume ¬ *Bet D E F*
have *A B C CongA G H I*
proof −
have *A B C D E F SumA G H I*
by (*simp add*: *assms*(*1*))
moreover have *E Out D F*
using ‹*Col D E F*› ‹¬ *Bet D E F*› *l6-4-2* **by** *auto*
ultimately show *?thesis*
using *out546-suma--conga* **by** *auto*
qed
show *?thesis*
by (*simp add*: ‹*A B C CongA G H I*› *conga--lea*)
qed
next
assume ¬ *Col D E F*
show *?thesis*
proof *cases*
assume *Col G H I*
show *?thesis*
proof *cases*
assume *Bet G H I*
thus *?thesis*
by (*metis assms*(*1*) *l11-31-2 suma-distincts*)
next
assume ¬ *Bet G H I*
then have *H Out G I*
by (*simp add*: ‹*Col G H I*› *l6-4-2*)
have *E Out D F* $\vee$ ¬ *Bet A B C*
using ‹¬ *Col A B C*› *bet-col* **by** *auto*

{
assume ¬ *Bet A B C*
then obtain *J* where *P2*: *C B J CongA D E F* ∧ ¬ *B C OS A J* ∧ *Coplanar A B C J* ∧ *A B J CongA G H I*
using *SumA-def assms*(*1*) by *blast*
have *G H I CongA A B J*
using *P2 not-conga-sym* by *blast*
then have *B Out A J*
using ‹*H Out G I*› *out-conga-out* by *blast*
then have *B C OS A J*
using *Col-perm* ‹¬ *Col A B C*› *out-one-side* by *blast*
then have *False*
using ‹*C B J CongA D E F* ∧ ¬ *B C OS A J* ∧ *Coplanar A B C J* ∧ *A B J CongA G H I*› by *linarith*
}
then have *False*
using *Col-def* ‹¬ *Col A B C*› by *blast*
thus *?thesis* by *blast*
qed
next
assume ¬ *Col G H I*
obtain *J* where *P4*: *C B J CongA D E F* ∧ ¬ *B C OS A J* ∧ ¬ *A B TS C J* ∧ *Coplanar A B C J*
using *SAMS-def assms*(*2*) by *auto*
{
assume *Col J B C*
have *J B C CongA D E F*
by (*simp add*: *P4 conga-left-comm*)
then have *Col D E F*
using *col-conga-col* ‹*Col J B C*› by *blast*
then have *False*
using ‹¬ *Col D E F*› by *blast*
}
then have ¬ *Col J B C* by *blast*
have *A B J CongA G H I*
proof −
have *A B C D E F SumA A B J*
proof −
have *C B J CongA D E F*
using *P4* by *simp*
moreover have ¬ *B C OS A J*
by (*simp add*: *P4*)
moreover have *Coplanar A B C J*
by (*simp add*: *P4*)
moreover have *A B J CongA A B J*
by (*metis* ‹¬ *Col A B C*› ‹¬ *Col J B C*› *col-trivial-1 conga-refl*)
ultimately show *?thesis*
using *SumA-def* by *blast*
qed
then show *?thesis*
using *assms*(*1*) *suma2--conga* by *auto*
qed
have ¬ *Col J B A*
using ‹*A B J CongA G H I*› ‹¬ *Col G H I*› *col-conga-col not-col-permutation-3* by *blast*
have *A B C LeA A B J*
proof −
have *C InAngle A B J*
by (*metis Col-perm P4* ‹¬ *Col A B C*› ‹¬ *Col J B A*› ‹¬ *Col J B C*› *cop-nos--ts coplanar-perm-7 coplanar-perm-8 invert-two-sides l9-2 os-ts--inangle*)
moreover have *A B C CongA A B C*
using *calculation in-angle-asym inangle3123 inangle-distincts* by *auto*
ultimately show *?thesis*
using *inangle--lea* by *blast*
qed
thus *?thesis*
using ‹*A B J CongA G H I*› *conga--lea lea-trans* by *blast*
qed
qed
qed

```
lemma sams-suma--lea456789:
  assumes A B C D E F SumA G H I and
    SAMS A B C D E F
  shows D E F LeA G H I
proof -
  have D E F A B C SumA G H I
    by (simp add: assms(1) suma-sym)
  moreover have SAMS D E F A B C
    using assms(2) sams-sym by blast
  ultimately show ?thesis
    using sams-suma--lea123789 by auto
qed

lemma sams-lea2--sams:
  assumes SAMS A' B' C' D' E' F' and
    A B C LeA A' B' C' and
    D E F LeA D' E' F'
  shows SAMS A B C D E F
proof -
  obtain A0 where B Midpoint A A0
    using symmetric-point-construction by auto
  obtain A'0 where B' Midpoint A' A'0
    using symmetric-point-construction by auto
  have D E F LeA C B A0
  proof -
    have D' E' F' LeA C B A0
    proof -
      have D' E' F' LeA C' B' A'0
        by (metis ‹B' Midpoint A' A'0› assms(1) assms(2) lea-distincts midpoint-bet midpoint-distinct-2 sams-chara)
      moreover have C' B' A'0 LeA C B A0
       by (metis Mid-cases ‹B Midpoint A A0› ‹B' Midpoint A' A'0› assms(2) l11-36-aux2 l7-3-2 lea-comm lea-distincts
midpoint-bet sym-preserve-diff)
      ultimately show ?thesis
        using lea-trans by blast
    qed
    moreover have D E F LeA D' E' F'
      using assms(3) by auto
    ultimately show ?thesis
      using ‹D' E' F' LeA C B A0› assms(3) lea-trans by blast
  qed
  then show ?thesis
    by (metis ‹B Midpoint A A0› assms(2) lea-distincts midpoint-bet sams-chara)
qed

lemma sams-lea456-suma2--lea:
  assumes D E F LeA D' E' F' and
    SAMS A B C D' E' F' and
    A B C D E F SumA G H I and
    A B C D' E' F' SumA G' H' I'
  shows G H I LeA G' H' I'
proof cases
  assume E' Out D' F'
  have G H I CongA G' H' I'
  proof -
    have G H I CongA A B C
    proof -
      have A B C D E F SumA G H I
        by (simp add: assms(3))
      moreover have E Out D F
        using ‹E' Out D' F'› assms(1) out-lea--out by blast
      ultimately show ?thesis
        using conga-sym out546-suma--conga by blast
    qed
    moreover have A B C CongA G' H' I'
      using ‹E' Out D' F'› assms(4) out546-suma--conga by blast
```

```
      ultimately show ?thesis
        using conga-trans by blast
    qed
    thus ?thesis
      by (simp add: conga--lea)
  next
    assume T1: ¬ E′ Out D′ F′
    show ?thesis
    proof cases
      assume T2: Col A B C
      have E′ Out D′ F′ ∨ ¬ Bet A B C
        using assms(2) SAMS-def by simp
      {
        assume ¬ Bet A B C
        then have B Out A C
          by (simp add: T2 l6-4-2)
        have G H I LeA G′ H′ I′
        proof −
          have D E F LeA D′ E′ F′
            by (simp add: assms(1))
          moreover have D E F CongA G H I
            using ‹B Out A C› assms(3) out213-suma--conga by auto
          moreover have D′ E′ F′ CongA G′ H′ I′
            using ‹B Out A C› assms(4) out213-suma--conga by auto
          ultimately show ?thesis
            using l11-30 by blast
        qed
      }
      thus ?thesis
        using T1 ‹E′ Out D′ F′ ∨ ¬ Bet A B C› by auto
    next
      assume ¬ Col A B C
      show ?thesis
      proof cases
        assume Col D′ E′ F′
        have SAMS D′ E′ F′ A B C
          by (simp add: assms(2) sams-sym)
        {
          assume ¬ Bet D′ E′ F′
          then have G H I LeA G′ H′ I′
            using not-bet-out T1 ‹Col D′ E′ F′› by auto
        }
        thus ?thesis
          by (metis assms(2) assms(3) assms(4) bet-lea--bet l11-31-2 sams-suma--lea456789 suma-distincts)
      next
        assume ¬ Col D′ E′ F′
        show ?thesis
        proof cases
          assume Col D E F
          have ¬ Bet D E F
            using bet-lea--bet Col-def ‹¬ Col D′ E′ F′› assms(1) by blast
          thus ?thesis
          proof −
            have A B C LeA G′ H′ I′
              using assms(2) assms(4) sams-suma--lea123789 by auto
            moreover have A B C CongA G H I
              by (meson ‹Col D E F› ‹¬ Bet D E F› assms(3) or-bet-out out213-suma--conga suma-sym)
            moreover have G′ H′ I′ CongA G′ H′ I′
            proof −
              have G′ ≠ H′
                using calculation(1) lea-distincts by blast
              moreover have H′ ≠ I′
                using ‹A B C LeA G′ H′ I′› lea-distincts by blast
              ultimately show ?thesis
                using conga-refl by auto
            qed
```

ultimately show *?thesis*
using *l11-30* **by** *blast*
qed
next
assume ¬ *Col D E F*
show *?thesis*
proof *cases*
assume *Col G′ H′ I′*
show *?thesis*
proof *cases*
assume *Bet G′ H′ I′*
show *?thesis*
proof −
have $G \neq H$
using *assms*(*3*) *suma-distincts* **by** *auto*
moreover have $I \neq H$
using *assms*(*3*) *suma-distincts* **by** *blast*
moreover have $G' \neq H'$
using *assms*(*4*) *suma-distincts* **by** *auto*
moreover have $I' \neq H'$
using *assms*(*4*) *suma-distincts* **by** *blast*
ultimately show *?thesis*
by (*simp add*: ‹*Bet G′ H′ I′*› *l11-31-2*)
qed
next
assume ¬ *Bet G′ H′ I′*
have *B Out A C*
proof −
have *H′ Out G′ I′*
using ‹*Col G′ H′ I′*› *l6-4-2* **by** (*simp add*: ‹¬ *Bet G′ H′ I′*›)
moreover have *A B C LeA G′ H′ I′* **using** *sams-suma--lea123789*
using *assms*(*2*) *assms*(*4*) **by** *auto*
ultimately show *?thesis*
using *out-lea--out* **by** *blast*
qed
then have *Col A B C*
using *Col-perm out-col* **by** *blast*
then have *False*
using ‹¬ *Col A B C*› **by** *auto*
thus *?thesis* **by** *blast*
qed
next
assume ¬ *Col G′ H′ I′*
obtain *F′1* **where** *P5*: *C B F′1 CongA D′ E′ F′* ∧ ¬ *B C OS A F′1* ∧ ¬ *A B TS C F′1* ∧ *Coplanar A B C F′1*
using *SAMS-def assms*(*2*) **by** *auto*
have *P6*: *D E F LeA C B F′1*
proof −
have *D E F CongA D E F*
using ‹¬ *Col D E F*› *conga-refl not-col-distincts* **by** *fastforce*
moreover have *D′ E′ F′ CongA C B F′1*
by (*simp add*: *P5 conga-sym*)
ultimately show *?thesis*
using *assms*(*1*) *l11-30* **by** *blast*
qed
then obtain *F1* **where** *P6*: *F1 InAngle C B F′1* ∧ *D E F CongA C B F1*
using *LeA-def* **by** *auto*
have *A B F′1 CongA G′ H′ I′*
proof −
have *A B C D′ E′ F′ SumA A B F′1*
proof −
have *C B F′1 CongA D′ E′ F′*
using *P5* **by** *blast*
moreover have ¬ *B C OS A F′1*
using *P5* **by** *auto*
moreover have *Coplanar A B C F′1*

```
        by (simp add: P5)
      moreover have A B F'1 CongA A B F'1
      proof -
        have A ≠ B
          using ‹¬ Col A B C› col-trivial-1 by blast
        moreover have B ≠ F'1
          using P6 inangle-distincts by auto
        ultimately show ?thesis
          using conga-refl by auto
      qed
      ultimately show ?thesis
        using SumA-def by blast
    qed
    moreover have A B C D' E' F' SumA G' H' I'
      by (simp add: assms(4))
    ultimately show ?thesis
      using suma2--conga by auto
  qed
  have ¬ Col A B F'1
    using ‹A B F'1 CongA G' H' I'› ‹¬ Col G' H' I'› col-conga-col by blast
  have ¬ Col C B F'1
  proof -
    have ¬ Col D' E' F'
      by (simp add: ‹¬ Col D' E' F'›)
    moreover have D' E' F' CongA C B F'1
      using P5 not-conga-sym by blast
    ultimately show ?thesis
      using ncol-conga-ncol by blast
  qed
  show ?thesis
  proof -
    have A B F1 LeA A B F'1
    proof -
      have F1 InAngle A B F'1
      proof -
        have F1 InAngle F'1 B A
        proof -
          have F1 InAngle F'1 B C
            by (simp add: P6 l11-24)
          moreover have C InAngle F'1 B A
          proof -
            have B C TS A F'1
              using Col-perm P5 ‹¬ Col A B C› ‹¬ Col C B F'1› cop-nos--ts ncoplanar-perm-12 by blast
            obtain X where Col X B C ∧ Bet A X F'1
              using TS-def ‹B C TS A F'1› by auto
            have Bet F'1 X A
              using Bet-perm ‹Col X B C ∧ Bet A X F'1› by blast
            moreover have (X = B) ∨ (B Out X C)
            proof -
              have B A OS X C
              proof -
                have A B OS X F'1
                    by (metis ‹Col X B C ∧ Bet A X F'1› ‹¬ Col A B C› ‹¬ Col A B F'1› bet-out-1 calculation
out-one-side)
                moreover have A B OS F'1 C
                  using Col-perm P5 ‹¬ Col A B C› ‹¬ Col A B F'1› cop-nos--ts one-side-symmetry by blast
                ultimately show ?thesis
                  using invert-one-side one-side-transitivity by blast
              qed
              thus ?thesis
                using Col-cases ‹Col X B C ∧ Bet A X F'1› col-one-side-out by blast
            qed
            ultimately show ?thesis
              by (metis InAngle-def ‹¬ Col A B C› ‹¬ Col A B F'1› not-col-distincts)
          qed
          ultimately show ?thesis
```

```
        using in-angle-trans by blast
      qed
      then show ?thesis
        using l11-24 by blast
    qed
    moreover have A B F1 CongA A B F1
    proof -
      have A ≠ B
        using ‹¬ Col A B C› col-trivial-1 by blast
      moreover have B ≠ F1
        using P6 conga-diff56 by blast
      ultimately show ?thesis
        using conga-refl by auto
    qed
    ultimately show ?thesis
      by (simp add: inangle--lea)
  qed
  moreover have A B F1 CongA G H I
  proof -
    have A B C D E F SumA A B F1
    proof -
      have B C TS F1 A
      proof -
        have B C TS F'1 A
        proof -
          have B C TS A F'1
            using Col-perm P5 ‹¬ Col A B C› ‹¬ Col C B F'1› cop-nos--ts ncoplanar-perm-12 by blast
          thus ?thesis
            using l9-2 by blast
        qed
        moreover have B C OS F'1 F1
        proof -
          have ¬ Col C B F'1
            by (simp add: ‹¬ Col C B F'1›)
          moreover have ¬ Col B C F1
          proof -
            have ¬ Col D E F
              using ‹¬ Col D E F› by auto
            moreover have D E F CongA C B F1
              by (simp add: P6)
            ultimately show ?thesis
              using ncol-conga-ncol not-col-permutation-4 by blast
          qed
          moreover have F1 InAngle C B F'1 using P6 by blast
          ultimately show ?thesis
            using in-angle-one-side invert-one-side one-side-symmetry by blast
        qed
        ultimately show ?thesis
          using l9-8-2 by blast
      qed
      thus ?thesis
      proof -
        have C B F1 CongA D E F
          using P6 not-conga-sym by blast
        moreover have ¬ B C OS A F1
          using ‹B C TS F1 A› l9-9 one-side-symmetry by blast
        moreover have Coplanar A B C F1
          using ‹B C TS F1 A› ncoplanar-perm-9 ts--coplanar by blast
        moreover have A B F1 CongA A B F1
        proof -
          have A ≠ B
            using ‹¬ Col A B C› col-trivial-1 by blast
          moreover have B ≠ F1
            using P6 conga-diff56 by blast
          ultimately show ?thesis
            using conga-refl by auto
```

```
                qed
                ultimately show ?thesis
                  using SumA-def by blast
              qed
            qed
            moreover have A B C D E F SumA G H I
              by (simp add: assms(3))
            ultimately show ?thesis
              using suma2--conga by auto
          qed
          ultimately show ?thesis
            using ‹A B F'1 CongA G' H' I'› l11-30 by blast
        qed
      qed
    qed
  qed
 qed
qed

lemma sams-lea123-suma2--lea:
  assumes A B C LeA A' B' C' and
    SAMS A' B' C' D E F and
    A B C D E F SumA G H I and
    A' B' C' D E F SumA G' H' I'
  shows G H I LeA G' H' I'
  by (meson assms(1) assms(2) assms(3) assms(4) sams-lea456-suma2--lea sams-sym suma-sym)

lemma sams-lea2-suma2--lea:
  assumes A B C LeA A' B' C' and
    D E F LeA D' E' F' and
    SAMS A' B' C' D' E' F' and
    A B C D E F SumA G H I and
    A' B' C' D' E' F' SumA G' H' I'
  shows G H I LeA G' H' I'
proof -
  obtain G'' H'' I'' where A B C D' E' F' SumA G'' H'' I''
    using assms(4) assms(5) ex-suma suma-distincts by presburger
  have G H I LeA G'' H'' I''
  proof -
    have D E F LeA D' E' F'
      by (simp add: assms(2))
    moreover have SAMS A B C D' E' F'
    proof -
      have SAMS A' B' C' D' E' F'
        by (simp add: assms(3))
      moreover have A B C LeA A' B' C'
        using assms(1) by auto
      moreover have D' E' F' LeA D' E' F'
        using assms(2) lea-distincts lea-refl by blast
      ultimately show ?thesis
        using sams-lea2--sams by blast
    qed
    moreover have A B C D E F SumA G H I
      by (simp add: assms(4))
    moreover have A B C D' E' F' SumA G'' H'' I''
      by (simp add: ‹A B C D' E' F' SumA G'' H'' I''›)
    ultimately show ?thesis
      using sams-lea456-suma2--lea by blast
  qed
  moreover have G'' H'' I'' LeA G' H' I'
    using sams-lea123-suma2--lea assms(3) assms(5) ‹A B C D' E' F' SumA G'' H'' I''› assms(1) by blast
  ultimately show ?thesis
    using lea-trans by blast
qed

lemma sams2-suma2--conga456:
```

assumes *SAMS A B C D E F* **and**
SAMS A B C D′ E′ F′ **and**
A B C D E F SumA G H I **and**
A B C D′ E′ F′ SumA G H I
shows *D E F CongA D′ E′ F′*
proof *cases*
assume *Col A B C*
show *?thesis*
proof *cases*
assume *P2*: *Bet A B C*
then have *E Out D F*
using *assms*(*1*) *SAMS-def* **by** *blast*
moreover have *E′ Out D′ F′*
using *P2 assms*(*2*) *SAMS-def* **by** *blast*
ultimately show *?thesis*
by (*simp add*: *l11-21-b*)
next
assume ¬ *Bet A B C*
then have *B Out A C*
using ‹*Col A B C*› *or-bet-out* **by** *blast*
show *?thesis*
proof −
have *D E F CongA G H I*
proof −
have *A B C D E F SumA G H I*
by (*simp add*: *assms*(*3*))
thus *?thesis*
using ‹*B Out A C*› *out213-suma--conga* **by** *auto*
qed
moreover have *G H I CongA D′ E′ F′*
proof −
have *A B C D′ E′ F′ SumA G H I*
by (*simp add*: *assms*(*4*))
then have *D′ E′ F′ CongA G H I*
using ‹*B Out A C*› *out213-suma--conga* **by** *auto*
thus *?thesis*
using *not-conga-sym* **by** *blast*
qed
ultimately show *?thesis*
using *not-conga* **by** *blast*
qed
qed
next
assume ¬ *Col A B C*
obtain *J* **where** *T1*: *C B J CongA D E F* ∧ ¬ *B C OS A J* ∧ ¬ *A B TS C J* ∧ *Coplanar A B C J*
using *assms*(*1*) *SAMS-def* **by** *blast*
have *T1A*: *C B J CongA D E F*
using *T1* **by** *simp*
have *T1B*: ¬ *B C OS A J*
using *T1* **by** *simp*
have *T1C*: ¬ *A B TS C J*
using *T1* **by** *simp*
have *T1D*: *Coplanar A B C J*
using *T1* **by** *simp*
obtain *J′* **where** *T2*: *C B J′ CongA D′ E′ F′* ∧ ¬ *B C OS A J′* ∧ ¬ *A B TS C J′* ∧ *Coplanar A B C J′*
using *assms*(*2*) *SAMS-def* **by** *blast*
have *T2A*: *C B J′ CongA D′ E′ F′*
using *T2* **by** *simp*
have *T2B*: ¬ *B C OS A J′*
using *T2* **by** *simp*
have *T2C*: ¬ *A B TS C J′*
using *T2* **by** *simp*
have *T2D*: *Coplanar A B C J′*
using *T2* **by** *simp*
have *T3*: *C B J CongA C B J′*
proof −

```
    have A B J CongA A B J′
    proof −
      have A B J CongA G H I
      proof −
        have A B C D E F SumA A B J
          using SumA-def T1A T1B T1D ‹¬ Col A B C› conga-distinct conga-refl not-col-distincts by auto
        thus ?thesis
          using assms(3) suma2--conga by blast
      qed
      moreover have G H I CongA A B J′
      proof −
        have A B C D′ E′ F′ SumA G H I
          by (simp add: assms(4))
        moreover have A B C D′ E′ F′ SumA A B J′
          using SumA-def T2A T2B T2D ‹¬ Col A B C› conga-distinct conga-refl not-col-distincts by auto
        ultimately show ?thesis
          using suma2--conga by auto
      qed
      ultimately show ?thesis
        using conga-trans by blast
    qed
    have B Out J J′ ∨ A B TS J J′
    proof −
      have Coplanar A B J J′
        using T1D T2D ‹¬ Col A B C› coplanar-trans-1 ncoplanar-perm-8 not-col-permutation-2 by blast
      moreover have A B J CongA A B J′
        by (simp add: ‹A B J CongA A B J′›)
      ultimately show ?thesis
        by (simp add: conga-cop--or-out-ts)
    qed
    {
      assume B Out J J′
      then have C B J CongA C B J′
        by (metis Out-cases ‹¬ Col A B C› bet-out between-trivial not-col-distincts out2--conga)
    }
    {
      assume A B TS J J′
      then have A B OS J C
        by (meson T1C T1D TS-def ‹¬ Col A B C› cop-nts--os not-col-permutation-2 one-side-symmetry)
      then have A B TS C J′
        using ‹A B TS J J′› l9-8-2 by blast
      then have False
        by (simp add: T2C)
      then have C B J CongA C B J′
        by blast
    }
    thus ?thesis
      using ‹B Out J J′ ⟹ C B J CongA C B J′› ‹B Out J J′ ∨ A B TS J J′› by blast
  qed
  then have C B J CongA D′ E′ F′
    using T2A not-conga by blast
  thus ?thesis
    using T1A not-conga not-conga-sym by blast
qed

lemma sams2-suma2--conga123:
  assumes SAMS A B C D E F and
    SAMS A′ B′ C′ D E F and
    A B C D E F SumA G H I and
    A′ B′ C′ D E F SumA G H I
  shows A B C CongA A′ B′ C′
proof −
  have SAMS D E F A B C
    by (simp add: assms(1) sams-sym)
  moreover have SAMS D E F A′ B′ C′
    by (simp add: assms(2) sams-sym)
```

```
  moreover have D E F  A B C SumA G H I
    by (simp add: assms(3) suma-sym)
  moreover have D E F A′ B′ C′ SumA G H I
    using assms(4) suma-sym by blast
  ultimately show ?thesis
    using sams2-suma2--conga456 by auto
qed

lemma suma-assoc-1:
  assumes SAMS A B C D E F and
    SAMS D E F G H I and
    A B C D E F SumA A′ B′ C′ and
    D E F G H I SumA D′ E′ F′ and
    A′ B′ C′ G H I SumA K L M
  shows A B C D′ E′ F′ SumA K L M
proof −
  obtain A0 where P1: Bet A B A0 ∧ Cong A B B A0
    using Cong-perm segment-construction by blast
  obtain D0 where P2: Bet D E D0 ∧ Cong D E E D0
    using Cong-perm segment-construction by blast
  show ?thesis
  proof cases
    assume Col A B C
    show ?thesis
    proof cases
      assume Bet A B C
      then have E Out D F
        using SAMS-def assms(1) by simp
      show ?thesis
      proof −
        have A′ B′ C′ CongA A B C
          using assms(3) ‹E Out D F› conga-sym out546-suma--conga by blast
        moreover have G H I CongA D′ E′ F′
          using assms(4) ‹E Out D F› out213-suma--conga by auto
        ultimately show ?thesis
       by (meson Tarski-neutral-dimensionless.conga3-suma--suma Tarski-neutral-dimensionless.suma2--conga Tarski-neutral-dimensio
assms(5))
      qed
    next
      assume ¬ Bet A B C
      then have B Out A C
        using ‹Col A B C› l6-4-2 by auto
      have A ≠ B
        using ‹B Out A C› out-distinct by auto
      have B ≠ C
        using ‹¬ Bet A B C› between-trivial by auto
      have D′ ≠ E′
        using assms(4) suma-distincts by blast
      have E′ ≠ F′
        using assms(4) suma-distincts by auto
      show ?thesis
      proof −
        obtain K0 L0 M0 where P3:A B C D′ E′ F′ SumA K0 L0 M0
          using ex-suma ‹A ≠ B› ‹B ≠ C› ‹D′ ≠ E′› ‹E′ ≠ F′› by presburger
        moreover have A B C CongA A B C
          using ‹A ≠ B› ‹B ≠ C› conga-refl by auto
        moreover have D′ E′ F′ CongA D′ E′ F′
          using ‹D′ ≠ E′› ‹E′ ≠ F′› conga-refl by auto
        moreover have K0 L0 M0 CongA K L M
        proof −
          have K0 L0 M0 CongA D′ E′ F′
            using P3 ‹B Out A C› conga-sym out213-suma--conga by blast
          moreover have D′ E′ F′ CongA K L M
          proof −
            have D E F G H I SumA D′ E′ F′
              by (simp add: assms(4))
```

moreover have *D E F G H I SumA K L M*
by (*meson Tarski-neutral-dimensionless.conga3-suma--suma Tarski-neutral-dimensionless.out213-suma--conga Tarski-neutral-dimensionless.sams2-suma2--conga456 Tarski-neutral-dimensionless.suma2--conga Tarski-neutral-dimensionless-axioms* ‹*B Out A C*› *assms*(*2*) *assms*(*3*) *assms*(*5*) *calculation not-conga-sym*)
ultimately show *?thesis*
using *suma2--conga* **by** *auto*
qed
ultimately show *?thesis*
using *not-conga* **by** *blast*
qed
ultimately show *?thesis*
using *conga3-suma--suma* **by** *blast*
qed
qed
next
assume *T1*: ¬ *Col A B C*
have ¬ *Col C B A0*
by (*metis Col-def P1* ‹¬ *Col A B C*› *cong-diff l6-16-1*)
show *?thesis*
proof *cases*
assume *Col D E F*
show *?thesis*
proof *cases*
assume *Bet D E F*
have *H Out G I* **using** *SAMS-def assms*(*2*) ‹*Bet D E F*› **by** *blast*
have *A B C D E F SumA A′ B′ C′*
by (*simp add*: *assms*(*3*))
moreover have *A B C CongA A B C*
by (*metis* ‹¬ *Col A B C*› *conga-pseudo-refl conga-right-comm not-col-distincts*)
moreover have *D E F CongA D′ E′ F′*
using ‹*H Out G I*› *assms*(*4*) *out546-suma--conga* **by** *auto*
moreover have *A′ B′ C′ CongA K L M*
using ‹*H Out G I*› *assms*(*5*) *out546-suma--conga* **by** *auto*
ultimately show *?thesis*
using *conga3-suma--suma* **by** *blast*
next
assume ¬ *Bet D E F*
then have *E Out D F*
using *not-bet-out* **by** (*simp add*: ‹*Col D E F*›)
show *?thesis*
proof −
have *A′ B′ C′ CongA A B C*
using *assms*(*3*) ‹*E Out D F*› *conga-sym out546-suma--conga* **by** *blast*
moreover have *G H I CongA D′ E′ F′*
using *out546-suma--conga* ‹*E Out D F*› *assms*(*4*) *out213-suma--conga* **by** *auto*
moreover have *K L M CongA K L M*
proof −
have *K* ≠ *L* ∧ *L* ≠ *M*
using *assms*(*5*) *suma-distincts* **by** *blast*
thus *?thesis*
using *conga-refl* **by** *auto*
qed
ultimately show *?thesis*
using *assms*(*5*) *conga3-suma--suma* **by** *blast*
qed
qed
next
assume ¬ *Col D E F*
then have ¬ *Col F E D0*
by (*metis Col-def P2 cong-diff l6-16-1 not-col-distincts*)
show *?thesis*
proof *cases*
assume *Col G H I*
show *?thesis*
proof *cases*
assume *Bet G H I*

```
      have SAMS G H I D E F
        by (simp add: assms(2) sams-sym)
      then have E Out D F
        using SAMS-def ‹Bet G H I› by blast
      then have Col D E F
        using Col-perm out-col by blast
      then have False
        using ‹¬ Col D E F› by auto
      thus ?thesis by simp
    next
      assume ¬ Bet G H I
      then have H Out G I
        using SAMS-def by (simp add: ‹Col G H I› l6-4-2)
      show ?thesis
      proof -
        have A B C CongA A B C
          by (metis ‹¬ Col A B C› conga-refl not-col-distincts)
        moreover have D E F CongA D' E' F'
          using assms(4) out546-suma--conga ‹H Out G I› by auto
        moreover have A' B' C' CongA K L M
          using ‹H Out G I› assms(5) out546-suma--conga by auto
        ultimately show ?thesis
          using assms(3) conga3-suma--suma by blast
      qed
    qed
  next
    assume ¬ Col G H I
    have ¬ B C OS A A0
      using P1 col-trivial-1 one-side-chara by blast
    have E F TS D D0
      by (metis P2 ‹¬ Col D E F› ‹¬ Col F E D0› bet--ts bet-col between-trivial not-col-permutation-1)
    show ?thesis
    proof cases
      assume Col A' B' C'
      show ?thesis
      proof cases
        assume Bet A' B' C'
        show ?thesis
        proof cases
          assume Col D' E' F'
          show ?thesis
          proof cases
            assume Bet D' E' F'
            have A B C CongA G H I
            proof -
              have A B C CongA D0 E F
              proof -
                have SAMS A B C D E F
                  by (simp add: assms(1))
                moreover have SAMS D0 E F D E F
                      by (metis P2 ‹¬ Col D E F› ‹¬ Col F E D0› bet--sams between-symmetry not-col-distincts
sams-right-comm)
                moreover have A B C D E F SumA A' B' C'
                  by (simp add: assms(3))
                moreover have D0 E F D E F SumA A' B' C'
                proof -
                  have D E F D0 E F SumA A' B' C'
                  proof -
                    have F E D0 CongA D0 E F
                      by (metis ‹¬ Col F E D0› col-trivial-1 col-trivial-2 conga-pseudo-refl)
                    moreover have ¬ E F OS D D0
                      using P2 col-trivial-1 one-side-chara by blast
                    moreover have Coplanar D E F D0
                      by (meson P2 bet--coplanar ncoplanar-perm-1)
                    moreover have D E D0 CongA A' B' C'
                     using assms(3) P2 ‹Bet A' B' C'› ‹¬ Col F E D0› conga-line not-col-distincts suma-distincts by
```

auto
ultimately show *?thesis*
using *SumA-def* **by** *blast*
qed
thus *?thesis*
by (*simp add*: ‹*D E F D0 E F SumA A′ B′ C′*› *suma-sym*)
qed
ultimately show *?thesis*
using *sams2-suma2--conga123* **by** *blast*
qed
moreover have *D0 E F CongA G H I*
proof −
have *SAMS D E F D0 E F*
using *P2* ‹¬ *Col D E F*› ‹¬ *Col F E D0*› *bet--sams not-col-distincts sams-right-comm* **by** *auto*
moreover have *D E F D0 E F SumA D′ E′ F′*
proof −
have *F E D0 CongA D0 E F*
by (*metis* (*no-types*) ‹¬ *Col F E D0*› *col-trivial-1 col-trivial-2 conga-pseudo-refl*)
moreover have ¬ *E F OS D D0*
using *P2 col-trivial-1 one-side-chara* **by** *blast*
moreover have *Coplanar D E F D0*
using *P2 bet--coplanar ncoplanar-perm-1* **by** *blast*
moreover have *D E D0 CongA D′ E′ F′*
using *assms*(*3*) *P2* ‹*Bet D′ E′ F′*› ‹¬ *Col F E D0*› *assms*(*4*) *conga-line not-col-distincts suma-distincts*
by *auto*
ultimately show *?thesis*
using *SumA-def* **by** *blast*
qed
ultimately show *?thesis*
using *assms*(*2*) *assms*(*4*) *sams2-suma2--conga456* **by** *auto*
qed
ultimately show *?thesis*
using *conga-trans* **by** *blast*
qed
then have *G H I CongA A B C*
using *not-conga-sym* **by** *blast*
have *D′ E′ F′ A B C SumA K L M*
proof −
have *A′ B′ C′ CongA D′ E′ F′*
by (*metis Tarski-neutral-dimensionless.suma-distincts Tarski-neutral-dimensionless-axioms* ‹*Bet A′ B′ C′*› ‹*Bet D′ E′ F′*› *assms*(*4*) *assms*(*5*) *conga-line*)
then show *?thesis*
by (*meson Tarski-neutral-dimensionless.conga3-suma--suma Tarski-neutral-dimensionless.suma2--conga Tarski-neutral-dimensionless-axioms* ‹*G H I CongA A B C*› *assms*(*5*))
qed
thus *?thesis*
by (*simp add*: *suma-sym*)
next
assume ¬ *Bet D′ E′ F′*
then have *E′ Out D′ F′*
by (*simp add*: ‹*Col D′ E′ F′*› *l6-4-2*)
have *D E F LeA D′ E′ F′*
using *assms*(*2*) *assms*(*4*) *sams-suma--lea123789* **by** *blast*
then have *E Out D F*
using ‹*E′ Out D′ F′*› *out-lea--out* **by** *blast*
then have *Col D E F*
using *Col-perm out-col* **by** *blast*
then have *False*
using ‹¬ *Col D E F*› **by** *auto*
thus *?thesis* **by** *simp*
qed
next
assume ¬ *Col D′ E′ F′*
have *D E F CongA C B A0*
proof −
have *SAMS A B C D E F*

by (*simp add*: *assms*(*1*))
moreover have *SAMS A B C C B A0*
using *P1* ‹¬ *Col A B C*› ‹¬ *Col C B A0*› *bet--sams not-col-distincts* **by** *auto*
moreover have *A B C D E F SumA A′ B′ C′*
by (*simp add*: *assms*(*3*))
moreover have *A B C C B A0 SumA A′ B′ C′*
proof −
have *A B C C B A0 SumA A B A0*
by (*metis P1* ‹¬ *Col A B C*› ‹¬ *Col C B A0*› *bet--suma col-trivial-1 col-trivial-2*)
moreover have *A B C CongA A B C*
using ‹*SAMS A B C C B A0*› *calculation sams2-suma2--conga123* **by** *auto*
moreover have *C B A0 CongA C B A0*
using ‹*SAMS A B C C B A0*› *calculation*(*1*) *sams2-suma2--conga456* **by** *auto*
moreover have *A B A0 CongA A′ B′ C′*
using *P1* ‹*Bet A′ B′ C′*› ‹¬ *Col C B A0*› *assms*(*3*) *conga-line not-col-distincts suma-distincts* **by** *auto*
ultimately show *?thesis*
using *conga3-suma--suma* **by** *blast*
qed
ultimately show *?thesis*
using *sams2-suma2--conga456* **by** *blast*
qed
have *SAMS C B A0 G H I*
proof −
have *D E F CongA C B A0*
by (*simp add*: ‹*D E F CongA C B A0*›)
moreover have *G H I CongA G H I*
using ‹¬ *Col G H I*› *conga-refl not-col-distincts* **by** *fastforce*
moreover have *SAMS D E F G H I*
by (*simp add*: *assms*(*2*))
ultimately show *?thesis*
using *conga2-sams--sams* **by** *blast*
qed
then obtain *J* **where** *P3*: *A0 B J CongA G H I* ∧ ¬ *B A0 OS C J* ∧ ¬ *C B TS A0 J* ∧ *Coplanar C B A0 J*
using *SAMS-def* **by** *blast*
obtain *F1* **where** *P4*: *F E F1 CongA G H I* ∧ ¬ *E F OS D F1* ∧ ¬ *D E TS F F1* ∧ *Coplanar D E F F1*
using *SAMS-def assms*(*2*) **by** *auto*
have *C B J CongA D′ E′ F′*
proof −
have *C B J CongA D E F1*
proof −
have (*B A0 TS C J* ∧ *E F TS D F1*) ∨ (*B A0 OS C J* ∧ *E F OS D F1*)
proof −
have *B A0 TS C J*
proof −
have *Coplanar B A0 C J*
using *P3 ncoplanar-perm-12* **by** *blast*
moreover have ¬ *Col C B A0*
by (*simp add*: ‹¬ *Col C B A0*›)
moreover have ¬ *Col J B A0*
using *P3* ‹¬ *Col G H I*› *col-conga-col not-col-permutation-3* **by** *blast*
moreover have ¬ *B A0 OS C J*
using *P3* **by** *simp*
ultimately show *?thesis*
by (*simp add*: *cop-nos--ts*)
qed
moreover have *E F TS D F1*
proof −
have *Coplanar E F D F1*
using *P4 ncoplanar-perm-12* **by** *blast*
moreover have ¬ *Col D E F*
by (*simp add*: ‹¬ *Col D E F*›)
moreover have ¬ *Col F1 E F*
using *P4* ‹¬ *Col G H I*› *col-conga-col col-permutation-3* **by** *blast*
moreover have ¬ *E F OS D F1*
using *P4* **by** *auto*

```
        ultimately show ?thesis
          by (simp add: cop-nos--ts)
      qed
      ultimately show ?thesis
        by simp
    qed
    moreover have C B A0 CongA D E F
      using ‹D E F CongA C B A0› not-conga-sym by blast
    moreover have A0 B J CongA F E F1
    proof -
      have A0 B J CongA G H I
        by (simp add: P3)
      moreover have G H I CongA F E F1
        using P4 not-conga-sym by blast
      ultimately show ?thesis
        using conga-trans by blast
    qed
    ultimately show ?thesis
      using l11-22 by auto
  qed
  moreover have D E F1 CongA D' E' F'
  proof -
    have D E F G H I SumA D E F1
      using P4 SumA-def ‹¬ Col D E F› conga-distinct conga-refl not-col-distincts by auto
    moreover have D E F G H I SumA D' E' F'
      by (simp add: assms(4))
    ultimately show ?thesis
      using suma2--conga by auto
  qed
  ultimately show ?thesis
    using conga-trans by blast
qed
show ?thesis
proof -
  have A B C D' E' F' SumA A B J
  proof -
    have C B TS J A
    proof -
      have C B TS A0 A
      proof -
        have B ≠ A0
          using ‹¬ Col C B A0› not-col-distincts by blast
        moreover have ¬ Col B C A
          using Col-cases ‹¬ Col A B C› by auto
        moreover have Bet A B A0
          by (simp add: P1)
        ultimately show ?thesis
          by (metis Bet-cases Col-cases ‹¬ Col C B A0› bet--ts invert-two-sides not-col-distincts)
      qed
      moreover have C B OS A0 J
      proof -
        have ¬ Col J C B
          using ‹C B J CongA D' E' F'› ‹¬ Col D' E' F'› col-conga-col not-col-permutation-2 by blast
        moreover have ¬ Col A0 C B
          using Col-cases ‹¬ Col C B A0› by blast
        ultimately show ?thesis
          using P3 cop-nos--ts by blast
      qed
      ultimately show ?thesis
        using l9-8-2 by blast
    qed
    moreover have C B J CongA D' E' F'
      by (simp add: ‹C B J CongA D' E' F'›)
    moreover have ¬ B C OS A J
      using calculation(1) invert-one-side l9-9-bis one-side-symmetry by blast
    moreover have Coplanar A B C J
```

```
          using calculation(1) ncoplanar-perm-15 ts--coplanar by blast
        moreover have A B J CongA A B J
        proof -
          have A ≠ B
            using ‹¬ Col A B C› col-trivial-1 by auto
          moreover have B ≠ J
            using ‹C B TS J A› ts-distincts by blast
          ultimately show ?thesis
            using conga-refl by auto
        qed
        ultimately show ?thesis
          using SumA-def by blast
      qed
      moreover have A B J CongA K L M
      proof -
        have A' B' C' G H I SumA A B J
        proof -
          have A B A0 CongA A' B' C'
               using P1 ‹Bet A' B' C'› ‹¬ Col A B C› ‹¬ Col C B A0› assms(5) conga-line not-col-distincts
suma-distincts by auto
          moreover have A0 B J CongA G H I
            by (simp add: P3)
          moreover have A B A0 A0 B J SumA A B J
          proof -
            have A0 B J CongA A0 B J
            proof -
              have A0 ≠ B
                using ‹¬ Col C B A0› col-trivial-2 by auto
              moreover have B ≠ J
                using CongA-def ‹A0 B J CongA G H I› by blast
              ultimately show ?thesis
                using conga-refl by auto
            qed
            moreover have ¬ B A0 OS A J
              by (simp add: Col-def P1 col123--nos)
            moreover have Coplanar A B A0 J
              using P1 bet--coplanar by auto
            moreover have A B J CongA A B J
              using P1 ‹¬ Col A B C› between-symmetry calculation(1) l11-13 not-col-distincts by blast
            ultimately show ?thesis
              using SumA-def by blast
          qed
          ultimately show ?thesis
            by (meson conga3-suma--suma suma2--conga)
        qed
        moreover have A' B' C' G H I SumA K L M
          by (simp add: assms(5))
        ultimately show ?thesis
          using suma2--conga by auto
      qed
      ultimately show ?thesis
      proof -
        have A B C CongA A B C ∧ D' E' F' CongA D' E' F'
          using CongA-def ‹A B J CongA K L M› ‹C B J CongA D' E' F'› conga-refl by presburger
        then show ?thesis
          using ‹A B C D' E' F' SumA A B J› ‹A B J CongA K L M› conga3-suma--suma by blast
      qed
    qed
  qed
next
  assume ¬ Bet A' B' C'
  have B Out A C
  proof -
    have A B C LeA A' B' C' using assms(1) assms(3) sams-suma--lea123789 by auto
    moreover have B' Out A' C'
      using ‹Col A' B' C'› ‹¬ Bet A' B' C'› or-bet-out by blast
```

ultimately show *?thesis*
using *out-lea--out* **by** *blast*
qed
then have *Col A B C*
using *Col-perm out-col* **by** *blast*
then have *False*
using ‹¬ *Col A B C*› **by** *auto*
thus *?thesis* **by** *simp*
qed
next
assume ¬ *Col A′ B′ C′*
obtain *C1* **where** *P6*: *C B C1 CongA D E F* ∧ ¬ *B C OS A C1* ∧ ¬ *A B TS C C1* ∧ *Coplanar A B C C1*
using *SAMS-def assms*(*1*) **by** *auto*
have *P6A*: *C B C1 CongA D E F*
using *P6* **by** *simp*
have *P6B*: ¬ *B C OS A C1*
using *P6* **by** *simp*
have *P6C*: ¬ *A B TS C C1*
using *P6* **by** *simp*
have *P6D*: *Coplanar A B C C1*
using *P6* **by** *simp*
have *A′ B′ C′ CongA A B C1*
proof −
have *A B C D E F SumA A B C1*
using *P6A P6B P6D SumA-def* ‹¬ *Col A B C*› *conga-distinct conga-refl not-col-distincts* **by** *auto*
moreover have *A B C D E F SumA A′ B′ C′*
by (*simp add*: *assms*(*3*))
ultimately show *?thesis*
using *suma2--conga* **by** *auto*
qed
have *B C1 OS C A*
proof −
have *B A OS C C1*
proof −
have *A B OS C C1*
proof −
have ¬ *Col C A B*
using *Col-perm* ‹¬ *Col A B C*› **by** *blast*
moreover have ¬ *Col C1 A B*
using ‹¬ *Col A′ B′ C′*› ‹*A′ B′ C′ CongA A B C1*› *col-permutation-1 ncol-conga-ncol* **by** *blast*
ultimately show *?thesis*
using *P6C P6D cop-nos--ts* **by** *blast*
qed
thus *?thesis*
by (*simp add*: *invert-one-side*)
qed
moreover have *B C TS A C1*
proof −
have *Coplanar B C A C1*
using *P6D ncoplanar-perm-12* **by** *blast*
moreover have ¬ *Col C1 B C*
proof −
have *D E F CongA C1 B C*
using *P6A conga-left-comm not-conga-sym* **by** *blast*
thus *?thesis*
using ‹¬ *Col D E F*› *ncol-conga-ncol* **by** *blast*
qed
ultimately show *?thesis*
using *T1 P6B cop-nos--ts* **by** *blast*
qed
ultimately show *?thesis*
using *os-ts1324--os one-side-symmetry* **by** *blast*
qed
show *?thesis*
proof *cases*
assume *Col D′ E′ F′*

show *?thesis*
proof *cases*
assume *Bet D′ E′ F′*
obtain *C0* **where** *P7*: *Bet C B C0* ∧ *Cong C B B C0*
using *Cong-perm segment-construction* **by** *blast*
have *B C1 TS C C0*
by (*metis P7* ‹*B C1 OS C A*› *bet--ts cong-diff-2 not-col-distincts one-side-not-col123*)
show *?thesis*
proof −
have *A B C C B C0 SumA A B C0*
proof −
have *C B C0 CongA C B C0*
by (*metis P7 T1 cong-diff conga-line not-col-distincts*)
moreover have ¬ *B C OS A C0*
using *P7 bet-col col124--nos invert-one-side* **by** *blast*
moreover have *Coplanar A B C C0*
using *P7 bet--coplanar ncoplanar-perm-15* **by** *blast*
moreover have *A B C0 CongA A B C0*
by (*metis P7 T1 cong-diff conga-refl not-col-distincts*)
ultimately show *?thesis*
using *SumA-def* **by** *blast*
qed
moreover have *A B C0 CongA K L M*
proof −
have *A′ B′ C′ G H I SumA A B C0*
proof −
have *A B C1 C1 B C0 SumA A B C0*
using ‹*B C1 TS C C0*› ‹*B C1 OS C A*› *l9-8-2 ts--suma-1* **by** *blast*
moreover have *A B C1 CongA A′ B′ C′*
by (*simp add*: *P6* ‹*A′ B′ C′ CongA A B C1*› *conga-sym*)
moreover have *C1 B C0 CongA G H I*
proof −
have *SAMS C B C1 C1 B C0*
by (*metis P7* ‹*B C1 TS C C0*› *bet--sams ts-distincts*)
moreover have *SAMS C B C1 G H I*
proof −
have *D E F CongA C B C1*
using *P6A not-conga-sym* **by** *blast*
moreover have *G H I CongA G H I*
using ‹¬ *Col G H I*› *conga-refl not-col-distincts* **by** *fastforce*
moreover have *SAMS D E F G H I*
by (*simp add*: *assms*(*2*))
ultimately show *?thesis*
using *conga2-sams--sams* **by** *blast*
qed
moreover have *C B C1 C1 B C0 SumA C B C0*
by (*simp add*: ‹*B C1 TS C C0*› *ts--suma-1*)
moreover have *C B C1 G H I SumA C B C0*
proof −
have *D E F G H I SumA D′ E′ F′*
by (*simp add*: *assms*(*4*))
moreover have *D E F CongA C B C1*
using *P6A not-conga-sym* **by** *blast*
moreover have *G H I CongA G H I*
using ‹¬ *Col G H I*› *conga-refl not-col-distincts* **by** *fastforce*
moreover have *D′ E′ F′ CongA C B C0* **using** *P7 assms*(*4*)
by (*metis P6A Tarski-neutral-dimensionless.suma-distincts Tarski-neutral-dimensionless-axioms* ‹*Bet D′ E′ F′*› *cong-diff conga-diff1 conga-line*)
ultimately show *?thesis*
using *conga3-suma--suma* **by** *blast*
qed
ultimately show *?thesis*
using *sams2-suma2--conga456* **by** *auto*
qed
moreover have *A B C0 CongA A B C0*
by (*metis P7 T1 cong-diff conga-refl not-col-distincts*)

```
          ultimately show ?thesis
            using conga3-suma--suma by blast
        qed
        thus ?thesis
          using assms(5) suma2--conga by auto
      qed
      moreover have A B C CongA A B C
      proof –
        have A ≠ B ∧ B ≠ C
          using T1 col-trivial-1 col-trivial-2 by auto
        thus ?thesis
          using conga-refl by auto
      qed
      moreover have C B C0 CongA D′ E′ F′
      proof –
        have C ≠ B
          using T1 col-trivial-2 by blast
        moreover have B ≠ C0
          using ‹B C1 TS C C0› ts-distincts by blast
        moreover have D′ ≠ E′
          using assms(4) suma-distincts by blast
        moreover have E′ ≠ F′
          using assms(4) suma-distincts by auto
        ultimately show ?thesis
          by (simp add: P7 ‹Bet D′ E′ F′› conga-line)
      qed
      ultimately show ?thesis
        using conga3-suma--suma by blast
    qed
  next
    assume ¬ Bet D′ E′ F′
    then have E′ Out D′ F′
      by (simp add: ‹Col D′ E′ F′› l6-4-2)
    have D E F LeA D′ E′ F′
      using sams-suma--lea123789 assms(2) assms(4) by auto
    then have E Out D F
      using ‹E′ Out D′ F′› out-lea--out by blast
    then have False
      using Col-cases ‹¬ Col D E F› out-col by blast
    thus ?thesis by simp
  qed
next
  assume ¬ Col D′ E′ F′
  have SAMS C B C1 G H I
  proof –
    have D E F CongA C B C1
      using P6A not-conga-sym by blast
    moreover have G H I CongA G H I
      using ‹¬ Col G H I› conga-refl not-col-distincts by fastforce
    ultimately show ?thesis
      using assms(2) conga2-sams--sams by blast
  qed
  then obtain J where P7: C1 B J CongA G H I ∧ ¬ B C1 OS C J ∧ ¬ C B TS C1 J ∧ Coplanar C B C1 J
    using SAMS-def by blast
  have P7A: C1 B J CongA G H I
    using P7 by simp
  have P7B: ¬ B C1 OS C J
    using P7 by simp
  have P7C: ¬ C B TS C1 J
    using P7 by simp
  have P7D: Coplanar C B C1 J
    using P7 by simp
  obtain F1 where P8: F E F1 CongA G H I ∧ ¬ E F OS D F1 ∧ ¬ D E TS F F1 ∧ Coplanar D E F F1
    using SAMS-def assms(2) by auto
  have P8A: F E F1 CongA G H I
    using P8 by simp
```

```
have P8B: ¬ E F OS D F1
  using P8 by simp
have P8C: ¬ D E TS F F1
  using P8 by simp
have P8D: Coplanar D E F F1
  using P8 by simp
have C B J CongA D' E' F'
proof -
  have C B J CongA D E F1
  proof -
    have B C1 TS C J
    proof -
      have Coplanar B C1 C J
        using P7D ncoplanar-perm-12 by blast
      moreover have ¬ Col C B C1
        using ‹B C1 OS C A› not-col-permutation-2 one-side-not-col123 by blast
      moreover have ¬ Col J B C1
        using P7 ‹¬ Col G H I› col-conga-col not-col-permutation-3 by blast
      moreover have ¬ B C1 OS C J
        by (simp add: P7B)
      ultimately show ?thesis
        by (simp add: cop-nos--ts)
    qed
    moreover have E F TS D F1
    proof -
      have Coplanar E F D F1
        using P8D ncoplanar-perm-12 by blast
      moreover have ¬ Col F1 E F
        using P8 ‹¬ Col G H I› col-conga-col not-col-permutation-3 by blast
      ultimately show ?thesis
        using P8B ‹¬ Col D E F› cop-nos--ts by blast
    qed
    moreover have C B C1 CongA D E F
      using P6A by blast
    moreover have C1 B J CongA F E F1
      using P8 by (meson P7A not-conga not-conga-sym)
    ultimately show ?thesis
      using l11-22a by blast
  qed
  moreover have D E F1 CongA D' E' F'
  proof -
    have D E F G H I SumA D E F1
      using P8A P8B P8D SumA-def ‹¬ Col D E F› conga-distinct conga-refl not-col-distincts by auto
    moreover have D E F G H I SumA D' E' F'
      by (simp add: assms(4))
    ultimately show ?thesis
      using suma2--conga by auto
  qed
  ultimately show ?thesis
    using conga-trans by blast
qed
have ¬ Col C B C1
  using ‹B C1 OS C A› col123--nos col-permutation-1 by blast
show ?thesis
proof -
  have A B C C B J SumA A B J
  proof -
    have B C TS J A
    proof -
      have B C TS C1 A using cop-nos--ts
        using P6B P6D T1 ‹¬ Col C B C1› l9-2 ncoplanar-perm-12 not-col-permutation-3 by blast
      moreover have B C OS C1 J
      proof -
        have ¬ Col C1 C B
          using Col-perm ‹¬ Col C B C1› by blast
        moreover have ¬ Col J C B
```

```
                using ‹C B J CongA D' E' F'› ‹¬ Col D' E' F'› col-conga-col col-permutation-1 by blast
              ultimately show ?thesis
                using P7C P7D cop-nos--ts invert-one-side by blast
            qed
            ultimately show ?thesis
              using l9-8-2 by blast
          qed
          thus ?thesis
            by (simp add: l9-2 ts--suma-1)
        qed
        moreover have A B C CongA A B C
          using T1 conga-refl not-col-distincts by fastforce
        moreover have A B J CongA K L M
        proof -
          have A' B' C' G H I SumA A B J
          proof -
            have A B C1 C1 B J SumA A B J
            proof -
              have B C1 TS A J
              proof -
                have B C1 TS C J
                proof -
                  have Coplanar B C1 C J
                    using P7D ncoplanar-perm-12 by blast
                  moreover have ¬ Col J B C1
                    using P7 ‹¬ Col G H I› col-conga-col not-col-permutation-3 by blast
                  ultimately show ?thesis
                    by (simp add: ‹¬ Col C B C1› P7B cop-nos--ts)
                qed
                moreover have B C1 OS C A
                  by (simp add: ‹B C1 OS C A›)
                ultimately show ?thesis
                  using l9-8-2 by blast
              qed
              thus ?thesis
                by (simp add: ts--suma-1)
            qed
            moreover have A B C1 CongA A' B' C'
              using ‹A' B' C' CongA A B C1› not-conga-sym by blast
            moreover have C1 B J CongA G H I
              by (simp add: P7A)
            moreover have A B J CongA A B J
              using ‹A B C C B J SumA A B J› suma2--conga by auto
            ultimately show ?thesis
              using conga3-suma--suma by blast
          qed
          moreover have A' B' C' G H I SumA K L M
            using assms(5) by simp
          ultimately show ?thesis
            using suma2--conga by auto
        qed
        ultimately show ?thesis
          using ‹C B J CongA D' E' F'› conga3-suma--suma by blast
      qed
    qed
  qed
 qed
qed
qed
qed

lemma suma-assoc-2:
  assumes SAMS A B C D E F and
    SAMS D E F G H I and
    A B C D E F SumA A' B' C' and
    D E F G H I SumA D' E' F' and
```

$A\ B\ C\ D'\ E'\ F'$ *SumA* $K\ L\ M$
shows $A'\ B'\ C'\ G\ H\ I$ *SumA* $K\ L\ M$
by (*meson assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *sams-sym suma-assoc-1 suma-sym*)

lemma *suma-assoc*:
assumes *SAMS* $A\ B\ C\ D\ E\ F$ **and**
SAMS $D\ E\ F\ G\ H\ I$ **and**
$A\ B\ C\ D\ E\ F$ *SumA* $A'\ B'\ C'$ **and**
$D\ E\ F\ G\ H\ I$ *SumA* $D'\ E'\ F'$
shows
$A'\ B'\ C'\ G\ H\ I$ *SumA* $K\ L\ M \longleftrightarrow A\ B\ C\ D'\ E'\ F'$ *SumA* $K\ L\ M$
by (*meson assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *suma-assoc-1 suma-assoc-2*)

lemma *sams-assoc-1*:
assumes *SAMS* $A\ B\ C\ D\ E\ F$ **and**
SAMS $D\ E\ F\ G\ H\ I$ **and**
$A\ B\ C\ D\ E\ F$ *SumA* $A'\ B'\ C'$ **and**
$D\ E\ F\ G\ H\ I$ *SumA* $D'\ E'\ F'$ **and**
SAMS $A'\ B'\ C'\ G\ H\ I$
shows *SAMS* $A\ B\ C\ D'\ E'\ F'$
proof *cases*
assume *Col* $A\ B\ C$
{
assume E *Out* $D\ F$
have *SAMS* $A\ B\ C\ D'\ E'\ F'$
proof −
have $A'\ B'\ C'$ *CongA* $A\ B\ C$
using *assms*(*3*) ‹E *Out* $D\ F$› *conga-sym out546-suma--conga* **by** *blast*
moreover have $G\ H\ I$ *CongA* $D'\ E'\ F'$
using ‹E *Out* $D\ F$› *assms*(*4*) *out213-suma--conga* **by** *blast*
ultimately show *?thesis*
using *assms*(*5*) *conga2-sams--sams* **by** *blast*
qed
}
{
assume ¬ *Bet* $A\ B\ C$
then have *P1*: B *Out* $A\ C$
using ‹*Col* $A\ B\ C$› *or-bet-out* **by** *blast*
have *SAMS* $D'\ E'\ F'\ A\ B\ C$
proof −
have $D' \neq E'$
using *assms*(*4*) *suma-distincts* **by** *auto*
moreover have $F'\ E'\ F'$ *CongA* $A\ B\ C$
proof −
have $E' \neq F'$
using *assms*(*4*) *suma-distincts* **by** *blast*
then have E' *Out* $F'\ F'$
using *out-trivial* **by** *auto*
thus *?thesis*
using *P1 l11-21-b* **by** *blast*
qed
moreover have ¬ $E'\ F'$ *OS* $D'\ F'$
using *os-distincts* **by** *blast*
moreover have ¬ $D'\ E'$ *TS* $F'\ F'$
by (*simp add*: *not-two-sides-id*)
moreover have *Coplanar* $D'\ E'\ F'\ F'$
using *ncop-distincts* **by** *blast*
ultimately show *?thesis* **using** *SAMS-def P1* **by** *blast*
qed
then have *SAMS* $A\ B\ C\ D'\ E'\ F'$
using *sams-sym* **by** *blast*
}
thus *?thesis*
using *SAMS-def assms*(*1*) ‹E *Out* $D\ F \Longrightarrow$ *SAMS* $A\ B\ C\ D'\ E'\ F'$› **by** *blast*
next
assume ¬ *Col* $A\ B\ C$

show *?thesis*
proof *cases*
  assume *Col D E F*
  have *H Out G I* ∨ ¬ *Bet D E F*
    using *SAMS-def assms*(*2*) **by** *blast*
  {
    assume *H Out G I*
    have *SAMS A B C D′ E′ F′*
    proof −
      have *A B C CongA A B C*
        using ‹¬ *Col A B C*› *conga-refl not-col-distincts* **by** *fastforce*
      moreover have *D E F CongA D′ E′ F′*
        using ‹*H Out G I*› *assms*(*4*) *out546-suma--conga* **by** *blast*
      ultimately show *?thesis*
        using *assms*(*1*) *conga2-sams--sams* **by** *blast*
    qed
  }
  {
    assume ¬ *Bet D E F*
    then have *E Out D F*
      using ‹*Col D E F*› *l6-4-2* **by** *blast*
    have *SAMS A B C D′ E′ F′*
    proof −
      have *A′ B′ C′ CongA A B C*
        using *out546-suma--conga* ‹*E Out D F*› *assms*(*3*) *not-conga-sym* **by** *blast*
      moreover have *G H I CongA D′ E′ F′*
        using *out213-suma--conga* ‹*E Out D F*› *assms*(*4*) **by** *auto*
      ultimately show *?thesis*
        using *assms*(*5*) *conga2-sams--sams* **by** *auto*
    qed
  }
  thus *?thesis*
    using ‹*H Out G I* ⟹ *SAMS A B C D′ E′ F′*› ‹*H Out G I* ∨ ¬ *Bet D E F*› **by** *blast*
next
  assume ¬ *Col D E F*
  show *?thesis*
  proof *cases*
    assume *Col G H I*
    have *SAMS G H I D E F*
      by (*simp add*: *assms*(*2*) *sams-sym*)
    then have *E Out D F* ∨ ¬ *Bet G H I*
      using *SAMS-def* **by** *blast*
    {
      assume *E Out D F*
      then have *False*
        using *Col-cases* ‹¬ *Col D E F*› *out-col* **by** *blast*
      then have *SAMS A B C D′ E′ F′*
        by *simp*
    }
    {
      assume ¬ *Bet G H I*
      then have *H Out G I*
        by (*simp add*: ‹*Col G H I*› *l6-4-2*)
      have *SAMS A B C D′ E′ F′*
      proof −
        have *A B C CongA A B C*
          by (*metis* ‹¬ *Col A B C*› *col-trivial-1 col-trivial-2 conga-refl*)
        moreover have *D E F CongA D′ E′ F′*
          using *out546-suma--conga* ‹*H Out G I*› *assms*(*4*) **by** *blast*
        moreover have *SAMS A B C D E F*
          using *assms*(*1*) **by** *auto*
        ultimately show *?thesis*
          using *conga2-sams--sams* **by** *auto*
      qed
    }
    thus *?thesis*

using ‹E Out D F ∨ ¬ Bet G H I› ‹E Out D F ⟹ $SAMS$ A B C D' E' F'› by *blast*
next
assume ¬ *Col G H I*
show *?thesis*
proof −
have ¬ *Bet A B C*
using *Col-def* ‹¬ *Col A B C*› **by** *auto*
moreover have ∃ *J*. (*C B J CongA* D' E' F' ∧ ¬ *B C OS A J* ∧ ¬ *A B TS C J* ∧ *Coplanar A B C J*)
proof −
have ¬ *Col* A' B' C'
proof −
{
assume *Col* A' B' C'
have *H Out G I* ∨ ¬ *Bet* A' B' C'
using *SAMS-def assms*(*5*) **by** *blast*
{
assume *H Out G I*
then have *False*
using *Col-cases* ‹¬ *Col G H I*› *out-col* **by** *blast*
}
{
assume ¬ *Bet* A' B' C'
then have B' *Out* A' C'
using *not-bet-out* ‹*Col* A' B' C'› **by** *blast*
have *A B C LeA* A' B' C'
using *assms*(*1*) *assms*(*3*) *sams-suma--lea123789* **by** *auto*
then have *B Out A C*
using ‹B' *Out* A' C'› *out-lea--out* **by** *blast*
then have *Col A B C*
using *Col-perm out-col* **by** *blast*
then have *False*
using ‹¬ *Col A B C*› **by** *auto*
}
then have *False*
using ‹*H Out G I* ⟹ *False*› ‹*H Out G I* ∨ ¬ *Bet* A' B' C'› **by** *blast*
}
thus *?thesis* **by** *blast*
qed
obtain *C1* **where** *P1*: *C B C1 CongA D E F* ∧ ¬ *B C OS A C1* ∧ ¬ *A B TS C C1* ∧ *Coplanar A B C C1*
using *SAMS-def assms*(*1*) **by** *auto*
have *P1A*: *C B C1 CongA D E F*
using *P1* **by** *simp*
have *P1B*: ¬ *B C OS A C1*
using *P1* **by** *simp*
have *P1C*: ¬ *A B TS C C1*
using *P1* **by** *simp*
have *P1D*: *Coplanar A B C C1*
using *P1* **by** *simp*
have *A B C1 CongA* A' B' C'
proof −
have *A B C D E F SumA A B C1*
using *P1A P1B P1D SumA-def* ‹¬ *Col A B C*› *conga-distinct conga-refl not-col-distincts* **by** *auto*
thus *?thesis*
using *assms*(*3*) *suma2--conga* **by** *auto*
qed
have *SAMS C B C1 G H I*
proof −
have *D E F CongA C B C1*
using *P1A not-conga-sym* **by** *blast*
moreover have *G H I CongA G H I*
using ‹¬ *Col G H I*› *conga-refl not-col-distincts* **by** *fastforce*
ultimately show *?thesis* **using** *conga2-sams--sams*
using *assms*(*2*) **by** *blast*
qed
then obtain *J* **where** *T1*: *C1 B J CongA G H I* ∧ ¬ *B C1 OS C J* ∧ ¬ *C B TS C1 J* ∧ *Coplanar C B C1 J*
using *SAMS-def* **by** *auto*

```
have T1A: C1 B J CongA G H I
  using T1 by simp
have T1B: ¬ B C1 OS C J
  using T1 by simp
have T1C: ¬ C B TS C1 J
  using T1 by simp
have T1D: Coplanar C B C1 J
  using T1 by simp
have SAMS A B C1 C1 B J
proof -
  have A' B' C' CongA A B C1
    using ‹A B C1 CongA A' B' C'› not-conga-sym by blast
  moreover have G H I CongA C1 B J
    using T1A not-conga-sym by blast
  ultimately show ?thesis
    using assms(5) conga2-sams--sams by auto
qed
then obtain J' where T2: C1 B J' CongA C1 B J ∧ ¬ B C1 OS A J' ∧ ¬ A B TS C1 J' ∧ Coplanar A B C1 J'
  using SAMS-def by auto
have T2A: C1 B J' CongA C1 B J
  using T2 by simp
have T2B: ¬ B C1 OS A J'
  using T2 by simp
have T2C: ¬ A B TS C1 J'
  using T2 by simp
have T2D: Coplanar A B C1 J'
  using T2 by simp
have A' B' C' CongA A B C1
  using ‹A B C1 CongA A' B' C'› not-conga-sym by blast
then have ¬ Col A B C1
  using ncol-conga-ncol ‹¬ Col A' B' C'› by blast
have D E F CongA C B C1
  using P1A not-conga-sym by blast
then have ¬ Col C B C1
  using ncol-conga-ncol ‹¬ Col D E F› by blast
then have Coplanar C1 B A J
  using coplanar-trans-1 P1D T1D coplanar-perm-15 coplanar-perm-6 by blast
have Coplanar C1 B J' J
   using T2D ‹Coplanar C1 B A J› ‹¬ Col A B C1› coplanar-perm-14 coplanar-perm-6 coplanar-trans-1 by blast
have B Out J' J ∨ C1 B TS J' J
 by (meson T2 ‹Coplanar C1 B A J› ‹¬ Col A B C1› conga-cop--or-out-ts coplanar-trans-1 ncoplanar-perm-14 ncoplanar-perm-6)
{
  assume B Out J' J
  have ∃ J. (C B J CongA D' E' F' ∧ ¬ B C OS A J ∧ ¬ A B TS C J ∧ Coplanar A B C J)
  proof -
    have C B C1 C1 B J SumA C B J
    proof -
      have C1 B J CongA C1 B J
        using CongA-def T2A conga-refl by auto
      moreover have C B J CongA C B J
         using ‹¬ Col C B C1› calculation conga-diff56 conga-pseudo-refl conga-right-comm not-col-distincts by blast
      ultimately show ?thesis
        using T1D T1B SumA-def by blast
    qed
    then have D E F G H I SumA C B J
      using conga3-suma--suma by (meson P1A T1A suma2--conga)
    then have C B J CongA D' E' F'
      using assms(4) suma2--conga by auto
    moreover have ¬ B C OS A J
          by (metis (no-types, lifting) Col-perm P1B P1D T1C ‹¬ Col A B C› ‹¬ Col C B C1› cop-nos--ts coplanar-perm-8 invert-two-sides l9-2 l9-8-2)
    moreover have ¬ A B TS C J
```

proof *cases*
assume *Col A B J*
thus *?thesis*
using *TS-def invert-two-sides not-col-permutation-3* **by** *blast*
next
assume ¬ *Col A B J*
have *A B OS C J*
proof −
have *A B OS C C1*
by (*simp add*: *P1C P1D* ‹¬ *Col A B C1*› ‹¬ *Col A B C*› *cop-nts--os not-col-permutation-2*)
moreover have *A B OS C1 J*
proof −
have *A B OS C1 J′*
by (*metis T2C T2D* ‹*B Out J′ J*› ‹¬ *Col A B C1*› ‹¬ *Col A B J*› *col-trivial-2 colx cop-nts--os not-col-permutation-2 out-col out-distinct*)
thus *?thesis*
using ‹*B Out J′ J*› *invert-one-side out-out-one-side* **by** *blast*
qed
ultimately show *?thesis*
using *one-side-transitivity* **by** *blast*
qed
thus *?thesis*
using *l9-9* **by** *blast*
qed
moreover have *Coplanar A B C J*
by (*meson P1D* ‹*Coplanar C1 B A J*› ‹¬ *Col A B C1*› *coplanar-perm-18 coplanar-perm-2 coplanar-trans-1 not-col-permutation-2*)
ultimately show *?thesis*
by *blast*
qed
}
{
assume *C1 B TS J′ J*
have *B C1 OS C J*
proof −
have *B C1 TS C J′*
proof −
have *B C1 TS A J′*
by (*meson T2B T2D TS-def* ‹*C1 B TS J′ J*› ‹¬ *Col A B C1*› *cop-nts--os invert-two-sides ncoplanar-perm-12*)
moreover have *B C1 OS A C*
by (*meson P1B P1C P1D* ‹¬ *Col A B C1*› ‹¬ *Col A B C*› ‹¬ *Col C B C1*› *cop-nts--os invert-one-side ncoplanar-perm-12 not-col-permutation-2 not-col-permutation-3 os-ts1324--os*)
ultimately show *?thesis*
using *l9-8-2* **by** *blast*
qed
moreover have *B C1 TS J J′*
using ‹*C1 B TS J′ J*› *invert-two-sides l9-2* **by** *blast*
ultimately show *?thesis*
using *OS-def* **by** *blast*
qed
then have *False*
by (*simp add*: *T1B*)
then have ∃ *J*. (*C B J CongA D′ E′ F′* ∧ ¬ *B C OS A J* ∧ ¬ *A B TS C J* ∧ *Coplanar A B C J*)
by *auto*
}
thus *?thesis*
using ‹*B Out J′ J* ⟹ ∃ *J*. *C B J CongA D′ E′ F′* ∧ ¬ *B C OS A J* ∧ ¬ *A B TS C J* ∧ *Coplanar A B C J*› ‹*B Out J′ J* ∨ *C1 B TS J′ J*› **by** *blast*
qed
ultimately show *?thesis*
using *SAMS-def not-bet-distincts* **by** *auto*
qed
qed
qed
qed

lemma *sams-assoc-2*:
  assumes *SAMS A B C D E F* **and**
    SAMS D E F G H I **and**
    A B C D E F SumA A′ B′ C′ **and**
    D E F G H I SumA D′ E′ F′ **and**
    SAMS A B C D′ E′ F′
  shows *SAMS A′ B′ C′ G H I*
proof −
  have *SAMS G H I A′ B′ C′*
  proof −
    have *SAMS G H I D E F*
      by (*simp add*: *assms*(*2*) *sams-sym*)
    moreover have *SAMS D E F A B C*
      by (*simp add*: *assms*(*1*) *sams-sym*)
    moreover have *G H I D E F SumA D′ E′ F′*
      by (*simp add*: *assms*(*4*) *suma-sym*)
    moreover have *D E F A B C SumA A′ B′ C′*
      by (*simp add*: *assms*(*3*) *suma-sym*)
    moreover have *SAMS D′ E′ F′ A B C*
      by (*simp add*: *assms*(*5*) *sams-sym*)
    ultimately show *?thesis*
      using *sams-assoc-1* **by** *blast*
  qed
  thus *?thesis*
    using *sams-sym* **by** *blast*
qed

lemma *sams-assoc*:
  assumes *SAMS A B C D E F* **and**
    SAMS D E F G H I **and**
    A B C D E F SumA A′ B′ C′ **and**
    D E F G H I SumA D′ E′ F′
  shows (*SAMS A′ B′ C′ G H I*) ⟷ (*SAMS A B C D′ E′ F′*)
  using *sams-assoc-1 sams-assoc-2*
  by (*meson assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*))

lemma *sams-nos--nts*:
  assumes *SAMS A B C C B J* **and**
    ¬ *B C OS A J*
  shows ¬ *A B TS C J*
proof −
  obtain *J′* **where** *P1*: *C B J′ CongA C B J* ∧ ¬ *B C OS A J′* ∧ ¬ *A B TS C J′* ∧ *Coplanar A B C J′*
    using *SAMS-def assms*(*1*) **by** *blast*
  have *P1A*: *C B J′ CongA C B J*
    using *P1* **by** *simp*
  have *P1B*: ¬ *B C OS A J′*
    using *P1* **by** *simp*
  have *P1C*: ¬ *A B TS C J′*
    using *P1* **by** *simp*
  have *P1D*: *Coplanar A B C J′*
    using *P1* **by** *simp*
  have *P2*: *B Out C J* ∨ ¬ *Bet A B C*
    using *SAMS-def assms*(*1*) **by** *blast*
  {
    assume *A B TS C J*
    have *Coplanar C B J J′*
    proof −
      have ¬ *Col A C B*
        using *TS-def* ‹*A B TS C J*› *not-col-permutation-4* **by** *blast*
      moreover have *Coplanar A C B J*
        using ‹*A B TS C J*› *ncoplanar-perm-2 ts--coplanar* **by** *blast*
      moreover have *Coplanar A C B J′*
        using *P1D ncoplanar-perm-2* **by** *blast*
      ultimately show *?thesis*
        using *coplanar-trans-1* **by** *blast*
    qed

have *B Out J J'* ∨ *C B TS J J'*
by (*metis P1* ‹*A B TS C J*› ‹*Coplanar C B J J'*› *bet-conga--bet bet-out col-conga-col col-two-sides-bet conga-distinct conga-os--out conga-sym cop-nts--os invert-two-sides l5-2 l6-6 not-col-permutation-3 not-col-permutation-4*)
{
assume *B Out J J'*
have ¬ *Col B A J* ∨ ¬ *Col B A J'*
using *TS-def* ‹*A B TS C J*› *not-col-permutation-3* **by** *blast*
then have *A B OS C J'*
by (*metis* (*full-types*) ‹*B Out J J'*› *Col-cases P1C P1D TS-def* ‹*A B TS C J*› *col2--eq cop-nts--os l6-3-1 out-col*)
then have *A B TS C J'*
by (*meson* ‹*A B TS C J*› ‹*B Out J J'*› *l6-6 l9-2 not-col-distincts out-two-sides-two-sides*)
then have *False*
by (*simp add*: *P1C*)
}
{
assume *C B TS J J'*
then have ¬ *Col C A B* ∧ ¬ *Col J A B*
using *TS-def* ‹*A B TS C J*› **by** *blast*
then have *False*
by (*metis P1B P1D TS-def* ‹*C B TS J J'*› *assms*(*2*) *cop-nts--os invert-two-sides l9-8-1 ncoplanar-perm-12 not-col-permutation-1*)
}
then have *False*
using ‹*B Out J J'* ⟹ *False*› ‹*B Out J J'* ∨ *C B TS J J'*› **by** *blast*
}
thus *?thesis* **by** *auto*
qed

lemma *conga-sams-nos--nts*:
assumes *SAMS A B C D E F* **and**
C B J CongA D E F **and**
¬ *B C OS A J*
shows ¬ *A B TS C J*
proof −
have *SAMS A B C C B J*
proof −
have *A B C CongA A B C*
using *assms*(*1*) *conga-refl sams-distincts* **by** *fastforce*
moreover have *D E F CongA C B J*
using *assms*(*2*) *not-conga-sym* **by** *blast*
ultimately show *?thesis*
using *assms*(*1*) *conga2-sams--sams* **by** *auto*
qed
thus *?thesis*
by (*simp add*: *assms*(*3*) *sams-nos--nts*)
qed

lemma *sams-lea2-suma2--conga123*:
assumes *A B C LeA A' B' C'* **and**
D E F LeA D' E' F' **and**
SAMS A' B' C' D' E' F' **and**
A B C D E F SumA G H I **and**
A' B' C' D' E' F' SumA G H I
shows *A B C CongA A' B' C'*
proof −
have *SAMS A B C D E F*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *sams-lea2--sams* **by** *blast*
moreover have *SAMS A' B' C' D E F*
by (*metis assms*(*2*) *assms*(*3*) *lea-refl sams-distincts sams-lea2--sams*)
moreover have *A' B' C' D E F SumA G H I*
proof −
obtain *G' H' I'* **where** *P1*: *A' B' C' D E F SumA G' H' I'*
using *calculation*(*2*) *ex-suma sams-distincts* **by** *blast*
show *?thesis*
proof −
have $A' \neq B' \wedge B' \neq C'$

```
        using assms(1) lea-distincts by blast
      then have A′ B′ C′ CongA A′ B′ C′
        using conga-refl by auto
      moreover
      have D ≠ E ∧ E ≠ F
        using ‹SAMS A B C D E F› sams-distincts by blast
      then have D E F CongA D E F
        using conga-refl by auto
      moreover have G′ H′ I′ CongA G H I
      proof −
        have G′ H′ I′ LeA G H I
          using P1 assms(2) assms(3) assms(5) sams-lea456-suma2--lea by blast
        moreover have G H I LeA G′ H′ I′
        proof −
          have SAMS A′ B′ C′ D E F
            using ‹SAMS A′ B′ C′ D E F› by auto
          thus ?thesis
            using P1 assms(1) assms(4) sams-lea123-suma2--lea by blast
        qed
        ultimately show ?thesis
          by (simp add: lea-asym)
      qed
      ultimately show ?thesis
        using P1 conga3-suma--suma by blast
    qed
  qed
  ultimately show ?thesis
    using assms(4) sams2-suma2--conga123 by blast
qed

lemma sams-lea2-suma2--conga456:
  assumes A B C LeA A′ B′ C′ and
    D E F LeA D′ E′ F′ and
    SAMS A′ B′ C′ D′ E′ F′ and
    A B C D E F SumA G H I and
    A′ B′ C′ D′ E′ F′ SumA G H I
  shows D E F CongA D′ E′ F′
proof −
  have SAMS D′ E′ F′ A′ B′ C′
    by (simp add: assms(3) sams-sym)
  moreover have D E F A B C SumA G H I
    by (simp add: assms(4) suma-sym)
  moreover have D′ E′ F′ A′ B′ C′ SumA G H I
    by (simp add: assms(5) suma-sym)
  ultimately show ?thesis
    using assms(1) assms(2) sams-lea2-suma2--conga123 by auto
qed

lemma sams-suma--out213:
  assumes A B C D E F SumA D E F and
    SAMS A B C D E F
  shows B Out A C
proof −
  have E ≠ D
    using assms(2) sams-distincts by blast
  then have E Out D D
    using out-trivial by auto
  moreover have D E D CongA A B C
  proof −
    have D E D LeA A B C
      using assms(1) lea121345 suma-distincts by auto
    moreover
    have E ≠ D ∧ E ≠ F
      using assms(2) sams-distincts by blast
    then have D E F LeA D E F
      using lea-refl by auto
```

```
    moreover have D E D D E F SumA D E F
    proof -
      have ¬ E D OS D F
        using os-distincts by auto
      moreover have Coplanar D E D F
        using ncop-distincts by auto
      ultimately show ?thesis
        using SumA-def ‹D E F LeA D E F› lea-asym by blast
    qed
    ultimately show ?thesis
      using assms(1) assms(2) sams-lea2-suma2--conga123 by auto
  qed
  ultimately show ?thesis
    using eq-conga-out by blast
qed

lemma sams-suma--out546:
  assumes A B C D E F SumA A B C and
    SAMS A B C D E F
  shows E Out D F
proof -
  have D E F A B C SumA A B C
    using assms(1) suma-sym by blast
  moreover have SAMS D E F A B C
    using assms(2) sams-sym by blast
  ultimately show ?thesis
    using sams-suma--out213 by blast
qed

lemma sams-lea-lta123-suma2--lta:
  assumes A B C LtA A′ B′ C′ and
    D E F LeA D′ E′ F′ and
    SAMS A′ B′ C′ D′ E′ F′ and
    A B C D E F SumA G H I and
    A′ B′ C′ D′ E′ F′ SumA G′ H′ I′
  shows G H I LtA G′ H′ I′
proof -
  have G H I LeA G′ H′ I′
  proof -
    have A B C LeA A′ B′ C′
      by (simp add: assms(1) lta--lea)
    thus ?thesis
      using assms(2) assms(3) assms(4) assms(5) sams-lea2-suma2--lea by blast
  qed
  moreover have ¬ G H I CongA G′ H′ I′
  proof -
    {
      assume G H I CongA G′ H′ I′
      have A B C CongA A′ B′ C′
      proof -
        have A B C LeA A′ B′ C′
          by (simp add: assms(1) lta--lea)
        moreover have A′ B′ C′ D′ E′ F′ SumA G H I
          by (meson ‹G H I CongA G′ H′ I′› assms(3) assms(5) conga3-suma--suma conga-sym sams2-suma2--conga123
sams2-suma2--conga456)
        ultimately show ?thesis
          using assms(2) assms(3) assms(4) sams-lea2-suma2--conga123 by blast
      qed
      then have False
        using assms(1) lta-not-conga by auto
    }
    thus ?thesis
      by auto
  qed
  ultimately show ?thesis
    using LtA-def by blast
```

qed

lemma *sams-lea-lta456-suma2--lta*:
  assumes *A B C LeA A′ B′ C′* **and**
    D E F LtA D′ E′ F′ **and**
    SAMS A′ B′ C′ D′ E′ F′ **and**
    A B C D E F SumA G H I **and**
    A′ B′ C′ D′ E′ F′ SumA G′ H′ I′
  shows *G H I LtA G′ H′ I′*
  using *sams-lea-lta123-suma2--lta*
  by (*meson assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *sams-sym suma-sym*)

lemma *sams-lta2-suma2--lta*:
  assumes *A B C LtA A′ B′ C′* **and**
    D E F LtA D′ E′ F′ **and**
    SAMS A′ B′ C′ D′ E′ F′ **and**
    A B C D E F SumA G H I **and**
    A′ B′ C′ D′ E′ F′ SumA G′ H′ I′
  shows *G H I LtA G′ H′ I′*
  using *sams-lea-lta123-suma2--lta*
  by (*meson LtA-def assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*))

lemma *sams-lea2-suma2--lea123*:
  assumes *D′ E′ F′ LeA D E F* **and**
    G H I LeA G′ H′ I′ **and**
    SAMS A B C D E F **and**
    A B C D E F SumA G H I **and**
    A′ B′ C′ D′ E′ F′ SumA G′ H′ I′
  shows *A B C LeA A′ B′ C′*
proof *cases*
  assume *A′ B′ C′ LtA A B C*
  then have *G′ H′ I′ LtA G H I* **using** *sams-lea-lta123-suma2--lta*
    using *assms*(*1*) *assms*(*3*) *assms*(*4*) *assms*(*5*) **by** *blast*
  then have ¬ *G H I LeA G′ H′ I′*
    using *lea--nlta* **by** *blast*
  then have *False*
    using *assms*(*2*) **by** *auto*
  thus *?thesis* **by** *auto*
next
  assume ¬ *A′ B′ C′ LtA A B C*
  have $A' \neq B' \wedge B' \neq C' \wedge A \neq B \wedge B \neq C$
    using *assms*(*4*) *assms*(*5*) *suma-distincts* **by** *auto*
  thus *?thesis*
    by (*simp add*: ‹¬ *A′ B′ C′ LtA A B C*› *nlta--lea*)
qed

lemma *sams-lea2-suma2--lea456*:
  assumes *A′ B′ C′ LeA A B C* **and**
    G H I LeA G′ H′ I′ **and**
    SAMS A B C D E F **and**
    A B C D E F SumA G H I **and**
    A′ B′ C′ D′ E′ F′ SumA G′ H′ I′
  shows *D E F LeA D′ E′ F′*
proof −
  have *SAMS D E F A B C*
    by (*simp add*: *assms*(*3*) *sams-sym*)
  moreover have *D E F A B C SumA G H I*
    by (*simp add*: *assms*(*4*) *suma-sym*)
  moreover have *D′ E′ F′ A′ B′ C′ SumA G′ H′ I′*
    by (*simp add*: *assms*(*5*) *suma-sym*)
  ultimately show *?thesis*
    using *assms*(*1*) *assms*(*2*) *sams-lea2-suma2--lea123* **by** *blast*
qed

lemma *sams-lea-lta456-suma2--lta123*:
  assumes *D′ E′ F′ LtA D E F* **and**

G H I LeA G′ H′ I′ **and**
SAMS A B C D E F **and**
A B C D E F SumA G H I **and**
A′ B′ C′ D′ E′ F′ SumA G′ H′ I′
shows *A B C LtA A′ B′ C′*
proof *cases*
assume *A′ B′ C′ LeA A B C*
then have *G′ H′ I′ LtA G H I*
using *sams-lea-lta456-suma2--lta assms(1) assms(3) assms(4) assms(5)* **by** *blast*
then have ¬ *G H I LeA G′ H′ I′*
using *lea--nlta* **by** *blast*
then have *False*
using *assms(2)* **by** *blast*
thus *?thesis* **by** *blast*
next
assume ¬ *A′ B′ C′ LeA A B C*
have $A' \neq B' \wedge B' \neq C' \wedge A \neq B \wedge B \neq C$
using *assms(4) assms(5) suma-distincts* **by** *auto*
thus *?thesis* **using** *nlea--lta*
by (*simp add*: ‹¬ *A′ B′ C′ LeA A B C*›)
qed

lemma *sams-lea-lta123-suma2--lta456*:
assumes *A′ B′ C′ LtA A B C* **and**
G H I LeA G′ H′ I′ **and**
SAMS A B C D E F **and**
A B C D E F SumA G H I **and**
A′ B′ C′ D′ E′ F′ SumA G′ H′ I′
shows *D E F LtA D′ E′ F′*
proof −
have *SAMS D E F A B C*
by (*simp add*: *assms(3) sams-sym*)
moreover have *D E F A B C SumA G H I*
by (*simp add*: *assms(4) suma-sym*)
moreover have *D′ E′ F′ A′ B′ C′ SumA G′ H′ I′*
by (*simp add*: *assms(5) suma-sym*)
ultimately show *?thesis*
using *assms(1) assms(2) sams-lea-lta456-suma2--lta123* **by** *blast*
qed

lemma *sams-lea-lta789-suma2--lta123*:
assumes *D′ E′ F′ LeA D E F* **and**
G H I LtA G′ H′ I′ **and**
SAMS A B C D E F **and**
A B C D E F SumA G H I **and**
A′ B′ C′ D′ E′ F′ SumA G′ H′ I′
shows *A B C LtA A′ B′ C′*
proof *cases*
assume *A′ B′ C′ LeA A B C*
then have *G′ H′ I′ LeA G H I*
using *assms(1) assms(3) assms(4) assms(5) sams-lea2-suma2--lea* **by** *blast*
then have ¬ *G H I LtA G′ H′ I′*
by (*simp add*: *lea--nlta*)
then have *False*
using *assms(2)* **by** *blast*
thus *?thesis* **by** *auto*
next
assume ¬ *A′ B′ C′ LeA A B C*
have $A' \neq B' \wedge B' \neq C' \wedge A \neq B \wedge B \neq C$
using *assms(4) assms(5) suma-distincts* **by** *auto*
thus *?thesis*
using *nlea--lta* **by** (*simp add*: ‹¬ *A′ B′ C′ LeA A B C*›)
qed

lemma *sams-lea-lta789-suma2--lta456*:
assumes *A′ B′ C′ LeA A B C* **and**

G H I LtA G′ H′ I′ **and**
SAMS A B C D E F **and**
A B C D E F SumA G H I **and**
A′ B′ C′ D′ E′ F′ SumA G′ H′ I′
shows *D E F LtA D′ E′ F′*
proof −
have *SAMS D E F A B C*
by (*simp add*: *assms*(*3*) *sams-sym*)
moreover have *D E F A B C SumA G H I*
by (*simp add*: *assms*(*4*) *suma-sym*)
moreover have *D′ E′ F′ A′ B′ C′ SumA G′ H′ I′*
using *assms*(*5*) *suma-sym* **by** *blast*
ultimately show *?thesis*
using *assms*(*1*) *assms*(*2*) *sams-lea-lta789-suma2--lta123* **by** *blast*
qed

lemma *sams-lta2-suma2--lta123*:
assumes *D′ E′ F′ LtA D E F* **and**
G H I LtA G′ H′ I′ **and**
SAMS A B C D E F **and**
A B C D E F SumA G H I **and**
A′ B′ C′ D′ E′ F′ SumA G′ H′ I′
shows *A B C LtA A′ B′ C′*
proof −
have *D′ E′ F′ LeA D E F*
by (*simp add*: *assms*(*1*) *lta--lea*)
thus *?thesis*
using *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *sams-lea-lta789-suma2--lta123* **by** *blast*
qed

lemma *sams-lta2-suma2--lta456*:
assumes *A′ B′ C′ LtA A B C* **and**
G H I LtA G′ H′ I′ **and**
SAMS A B C D E F **and**
A B C D E F SumA G H I **and**
A′ B′ C′ D′ E′ F′ SumA G′ H′ I′
shows *D E F LtA D′ E′ F′*
proof −
have *A′ B′ C′ LeA A B C*
by (*simp add*: *assms*(*1*) *lta--lea*)
thus *?thesis*
using *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *sams-lea-lta789-suma2--lta456* **by** *blast*
qed

lemma *sams123231*:
assumes $A \neq B$ **and**
$A \neq C$ **and**
$B \neq C$
shows *SAMS A B C B C A*
proof −
obtain *A′* **where** *B Midpoint A A′*
using *symmetric-point-construction* **by** *auto*
then have $A' \neq B$
using *assms*(*1*) *midpoint-not-midpoint* **by** *blast*
moreover have *Bet A B A′*
by (*simp add*: ‹*B Midpoint A A′*› *midpoint-bet*)
moreover have *B C A LeA C B A′*
proof *cases*
assume *Col A B C*
show *?thesis*
proof *cases*
assume *Bet A C B*
thus *?thesis*
by (*metis assms*(*2*) *assms*(*3*) *between-exchange3 calculation*(*1*) *calculation*(*2*) *l11-31-2*)
next
assume ¬ *Bet A C B*

then have *C Out B A*
using *Col-cases* ‹*Col A B C*› *l6-6 or-bet-out* **by** *blast*
thus *?thesis*
using *assms*(*3*) *calculation*(*1*) *l11-31-1* **by** *auto*
qed
next
assume ¬ *Col A B C*
thus *?thesis*
using *l11-41-aux* ‹*B Midpoint A A'*› *calculation*(*1*) *lta--lea midpoint-bet not-col-permutation-4* **by** *blast*
qed
ultimately show *?thesis*
using *assms*(*1*) *sams-chara* **by** *blast*
qed

lemma *col-suma--col*:
assumes *Col D E F* **and**
A B C B C A SumA D E F
shows *Col A B C*
proof −
{
assume ¬ *Col A B C*
have *False*
proof *cases*
assume *Bet D E F*
obtain *P* **where** *P1*: *Bet A B P* ∧ *Cong A B B P*
using *Cong-perm segment-construction* **by** *blast*
have *D E F LtA A B P*
proof −
have *A B C LeA A B C*
using ‹¬ *Col A B C*› *lea-total not-col-distincts* **by** *blast*
moreover
have *B C TS A P*
by (*metis Cong-perm P1* ‹¬ *Col A B C*› *bet--ts bet-col between-trivial2 cong-reverse-identity not-col-permutation-1*)
then have *B C A LtA C B P*
using *Col-perm P1* ‹¬ *Col A B C*› *calculation l11-41-aux ts-distincts* **by** *blast*
moreover have *A B C C B P SumA A B P*
by (*simp add*: ‹*B C TS A P*› *ts--suma-1*)
ultimately show *?thesis*
by (*meson P1 Tarski-neutral-dimensionless.sams-lea-lta456-suma2--lta Tarski-neutral-dimensionless-axioms assms*(*2*) *bet-suma--sams*)
qed
thus *?thesis*
using *P1* ‹*Bet D E F*› *bet2-lta--lta lta-distincts* **by** *blast*
next
assume ¬ *Bet D E F*
have *C Out B A*
proof −
have *E Out D F*
by (*simp add*: ‹¬ *Bet D E F*› *assms*(*1*) *l6-4-2*)
moreover have *B C A LeA D E F*
using *sams-suma--lea456789*
by (*metis assms*(*2*) *sams123231 suma-distincts*)
ultimately show *?thesis*
using *out-lea--out* **by** *blast*
qed
thus *?thesis*
using *Col-cases* ‹¬ *Col A B C*› *out-col* **by** *blast*
qed
}
thus *?thesis* **by** *auto*
qed

lemma *ncol-suma--ncol*:
assumes ¬ *Col A B C* **and**
A B C B C A SumA D E F
shows ¬ *Col D E F*

using *col-suma--col assms*(*1*) *assms*(*2*) **by** *blast*

lemma *per2-suma--bet*:
assumes *Per A B C* **and**
Per D E F **and**
A B C D E F SumA G H I
shows *Bet G H I*
proof −
obtain *A1* **where** *P1*: *C B A1 CongA D E F* ∧ ¬ *B C OS A A1* ∧ *Coplanar A B C A1* ∧ *A B A1 CongA G H I*
using *SumA-def assms*(*3*) **by** *blast*
then have *D E F CongA A1 B C*
using *conga-right-comm conga-sym* **by** *blast*
then have *Per A1 B C*
using *assms*(*2*) *l11-17* **by** *blast*
have *Bet A B A1*
proof −
have *Col B A A1*
proof −
have *Coplanar C A A1 B*
using *P1 ncoplanar-perm-10* **by** *blast*
moreover have *C* ≠ *B*
using ‹*D E F CongA A1 B C*› *conga-distinct* **by** *auto*
ultimately show *?thesis*
using *assms*(*1*) ‹*Per A1 B C*› *col-permutation-2 cop-per2--col* **by** *blast*
qed
moreover have *B C TS A A1*
proof −
have *Coplanar B C A A1*
using *calculation ncop--ncols* **by** *blast*
moreover
have *A* ≠ *B* ∧ *B* ≠ *C*
using *CongA-def P1* **by** *blast*
then have ¬ *Col A B C*
by (*simp add*: *assms*(*1*) *per-not-col*)
moreover
have *A1* ≠ *B* ∧ *B* ≠ *C*
using ‹*D E F CongA A1 B C*› *conga-distinct* **by** *blast*
then have ¬ *Col A1 B C*
using ‹*Per A1 B C*› **by** (*simp add*: *per-not-col*)
ultimately show *?thesis*
by (*simp add*: *P1 cop-nos--ts*)
qed
ultimately show *?thesis*
using *col-two-sides-bet* **by** *blast*
qed
thus *?thesis*
using *P1 bet-conga--bet* **by** *blast*
qed

lemma *bet-per2--suma*:
assumes *A* ≠ *B* **and**
B ≠ *C* **and**
D ≠ *E* **and**
E ≠ *F* **and**
G ≠ *H* **and**
H ≠ *I* **and**
Per A B C **and**
Per D E F **and**
Bet G H I
shows *A B C D E F SumA G H I*
proof −
obtain *G′ H′ I′* **where** *A B C D E F SumA G′ H′ I′*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *ex-suma* **by** *blast*
moreover have *A B C CongA A B C*
using *assms*(*1*) *assms*(*2*) *conga-refl* **by** *auto*
moreover have *D E F CongA D E F*

using *assms*(*3*) *assms*(*4*) *conga-refl* **by** *auto*
moreover have *G′ H′ I′ CongA G H I*
proof −
have *G′* ≠ *H′*
using *calculation*(*1*) *suma-distincts* **by** *auto*
moreover have *H′* ≠ *I′*
using ‹*A B C D E F SumA G′ H′ I′*› *suma-distincts* **by** *blast*
moreover have *Bet G′ H′ I′*
using ‹*A B C D E F SumA G′ H′ I′*› *assms*(*7*) *assms*(*8*) *per2-suma--bet* **by** *auto*
ultimately show *?thesis*
using *conga-line* **by** (*simp add*: *assms*(*5*) *assms*(*6*) *assms*(*9*))
qed
ultimately show *?thesis*
using *conga3-suma--suma* **by** *blast*
qed

lemma *per2--sams*:
assumes *A* ≠ *B* **and**
B ≠ *C* **and**
D ≠ *E* **and**
E ≠ *F* **and**
Per A B C **and**
Per D E F
shows *SAMS A B C D E F*
proof −
obtain *G H I* **where** *A B C D E F SumA G H I*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *ex-suma* **by** *blast*
moreover then have *Bet G H I*
using *assms*(*5*) *assms*(*6*) *per2-suma--bet* **by** *auto*
ultimately show *?thesis*
using *bet-suma--sams* **by** *blast*
qed

lemma *bet-per-suma--per456*:
assumes *Per A B C* **and**
Bet G H I **and**
A B C D E F SumA G H I
shows *Per D E F*
proof −
obtain *A1* **where** *B Midpoint A A1*
using *symmetric-point-construction* **by** *auto*
have ¬ *Col A B C*
using *assms*(*1*) *assms*(*3*) *per-col-eq suma-distincts* **by** *blast*
have *A B C CongA D E F*
proof −
have *SAMS A B C A B C*
using ‹¬ *Col A B C*› *assms*(*1*) *not-col-distincts per2--sams* **by** *auto*
moreover have *SAMS A B C D E F*
using *assms*(*2*) *assms*(*3*) *bet-suma--sams* **by** *blast*
moreover have *A B C A B C SumA G H I*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *bet-per2--suma suma-distincts* **by** *blast*
ultimately show *?thesis*
using *assms*(*3*) *sams2-suma2--conga456* **by** *auto*
qed
thus *?thesis*
using *assms*(*1*) *l11-17* **by** *blast*
qed

lemma *bet-per-suma--per123*:
assumes *Per D E F* **and**
Bet G H I **and**
A B C D E F SumA G H I
shows *Per A B C*
using *bet-per-suma--per456*
by (*meson assms*(*1*) *assms*(*2*) *assms*(*3*) *suma-sym*)

```
lemma bet-suma--per:
  assumes Bet D E F and
    A B C A B C SumA D E F
  shows Per A B C
proof –
  obtain A' where C B A' CongA A B C ∧ A B A' CongA D E F
    using SumA-def assms(2) by blast
  have Per C B A
  proof –
    have Bet A B A'
    proof –
      have D E F CongA A B A'
        using ‹C B A' CongA A B C ∧ A B A' CongA D E F› not-conga-sym by blast
      thus ?thesis
        using assms(1) bet-conga--bet by blast
    qed
    moreover have C B A CongA C B A'
      using conga-left-comm not-conga-sym ‹C B A' CongA A B C ∧ A B A' CongA D E F› by blast
    ultimately show ?thesis
      using l11-18-2 by auto
  qed
  thus ?thesis
    using Per-cases by auto
qed

lemma acute--sams:
  assumes Acute A B C
  shows SAMS A B C A B C
proof –
  obtain A' where B Midpoint A A'
    using symmetric-point-construction by auto
  show ?thesis
  proof –
    have A ≠ B ∧ A' ≠ B
      using ‹B Midpoint A A'› acute-distincts assms is-midpoint-id-2 by blast
    moreover have Bet A B A'
      by (simp add: ‹B Midpoint A A'› midpoint-bet)
    moreover
    have Obtuse C B A'
      using acute-bet--obtuse assms calculation(1) calculation(2) obtuse-sym by blast
    then have A B C LeA C B A'
      by (metis acute--not-obtuse assms calculation(1) lea-obtuse-obtuse lea-total obtuse-distincts)
    ultimately show ?thesis
      using sams-chara by blast
  qed
qed

lemma acute-suma--nbet:
  assumes Acute A B C and
    A B C A B C SumA D E F
  shows ¬ Bet D E F
proof –
  {
    assume Bet D E F
    then have Per A B C
      using assms(2) bet-suma--per by auto
    then have A B C LtA A B C
      using acute-not-per assms(1) by auto
    then have False
      by (simp add: nlta)
  }
  thus ?thesis by blast
qed

lemma acute2--sams:
  assumes Acute A B C and
```

Acute D E F
shows *SAMS A B C D E F*
by (*metis acute--sams acute-distincts assms*(*1*) *assms*(*2*) *lea-total sams-lea2--sams*)

lemma *acute2-suma--nbet-a*:
assumes *Acute A B C* **and**
D E F LeA A B C **and**
A B C D E F SumA G H I
shows ¬ *Bet G H I*
proof −
{
assume *Bet G H I*
obtain *A′ B′ C′* **where** *A B C A B C SumA A′ B′ C′*
using *acute-distincts assms*(*1*) *ex-suma* **by** *moura*
have *G H I LeA A′ B′ C′*
proof −
have *A B C LeA A B C*
using *acute-distincts assms*(*1*) *lea-refl* **by** *blast*
moreover have *SAMS A B C A B C*
by (*simp add*: *acute--sams assms*(*1*))
ultimately show *?thesis*
using ‹*A B C A B C SumA A′ B′ C′*› *assms*(*1*) *assms*(*2*) *assms*(*3*) *sams-lea456-suma2--lea* **by** *blast*
qed
then have *Bet A′ B′ C′*
using ‹*Bet G H I*› *bet-lea--bet* **by** *blast*
then have *False*
using *acute-suma--nbet assms*(*1*) *assms*(*3*) ‹*A B C A B C SumA A′ B′ C′*› **by** *blast*
}
thus *?thesis* **by** *blast*
qed

lemma *acute2-suma--nbet*:
assumes *Acute A B C* **and**
Acute D E F **and**
A B C D E F SumA G H I
shows ¬ *Bet G H I*
proof −
have $A \neq B \wedge B \neq C \wedge D \neq E \wedge E \neq F$
using *assms*(*3*) *suma-distincts* **by** *auto*
then have *A B C LeA D E F* ∨ *D E F LeA A B C*
by (*simp add*: *lea-total*)
moreover
{
assume *P3*: *A B C LeA D E F*
have *D E F A B C SumA G H I*
by (*simp add*: *assms*(*3*) *suma-sym*)
then have ¬ *Bet G H I*
using *P3 assms*(*2*) *acute2-suma--nbet-a* **by** *auto*
}
{
assume *D E F LeA A B C*
then have ¬ *Bet G H I*
using *acute2-suma--nbet-a assms*(*1*) *assms*(*3*) **by** *auto*
}
thus *?thesis*
using ‹*A B C LeA D E F* ⟹ ¬ *Bet G H I*› *calculation* **by** *blast*
qed

lemma *acute-per--sams*:
assumes $A \neq B$ **and**
$B \neq C$ **and**
Per A B C **and**
Acute D E F
shows *SAMS A B C D E F*
proof −
have *SAMS A B C A B C*

by (*simp add*: *assms*(*1*) *assms*(*2*) *assms*(*3*) *per2--sams*)
moreover have *A B C LeA A B C*
using *assms*(*1*) *assms*(*2*) *lea-refl* **by** *auto*
moreover have *D E F LeA A B C*
by (*metis acute-distincts acute-lea-acute acute-not-per assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *lea-total*)
ultimately show *?thesis*
using *sams-lea2--sams* **by** *blast*
qed

lemma *acute-per-suma--nbet*:
assumes *A* ≠ *B* **and**
B ≠ *C* **and**
Per A B C **and**
Acute D E F **and**
A B C D E F SumA G H I
shows ¬ *Bet G H I*
proof −
{
assume *Bet G H I*
have *G H I LtA G H I*
proof −
have *A B C LeA A B C*
using *assms*(*1*) *assms*(*2*) *lea-refl* **by** *auto*
moreover have *D E F LtA A B C*
by (*simp add*: *acute-per--lta assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*))
moreover have *SAMS A B C A B C*
by (*simp add*: *assms*(*1*) *assms*(*2*) *assms*(*3*) *per2--sams*)
moreover have *A B C D E F SumA G H I*
by (*simp add*: *assms*(*5*))
moreover have *A B C A B C SumA G H I*
by (*meson Tarski-neutral-dimensionless.bet-per-suma--per456 Tarski-neutral-dimensionless-axioms* ‹*Bet G H I*› *acute-not-per assms*(*3*) *assms*(*4*) *calculation*(*4*))
ultimately show *?thesis*
using *sams-lea-lta456-suma2--lta* **by** *blast*
qed
then have *False*
by (*simp add*: *nlta*)
}
thus *?thesis* **by** *blast*
qed

lemma *obtuse--nsams*:
assumes *Obtuse A B C*
shows ¬ *SAMS A B C A B C*
proof −
{
assume *SAMS A B C A B C*
obtain *A'* **where** *B Midpoint A A'*
using *symmetric-point-construction* **by** *auto*
have *A B C LtA A B C*
proof −
have *A B C LeA A' B C*
by (*metis* ‹*B Midpoint A A'*› ‹*SAMS A B C A B C*› *lea-right-comm midpoint-bet midpoint-distinct-2 sams-chara sams-distincts*)
moreover have *A' B C LtA A B C*
using ‹*B Midpoint A A'*› *assms calculation lea-distincts midpoint-bet obtuse-chara* **by** *blast*
ultimately show *?thesis*
using *lea--nlta* **by** *blast*
qed
then have *False*
by (*simp add*: *nlta*)
}
thus *?thesis* **by** *blast*
qed

lemma *nbet-sams-suma--acute*:

assumes ¬ *Bet D E F* **and**
SAMS A B C A B C **and**
A B C A B C SumA D E F
shows *Acute A B C*
proof −
have *Acute A B C* ∨ *Per A B C* ∨ *Obtuse A B C*
by (*metis angle-partition l8-20-1-R1 l8-5*)
{
assume *Per A B C*
then have *Bet D E F*
using *assms*(*3*) *per2-suma--bet* **by** *auto*
then have *False*
using *assms*(*1*) **by** *auto*
}
{
assume *Obtuse A B C*
then have ¬ *SAMS A B C A B C*
by (*simp add*: *obtuse--nsams*)
then have *False*
using *assms*(*2*) **by** *auto*
}
thus *?thesis*
using ‹*Acute A B C* ∨ *Per A B C* ∨ *Obtuse A B C*› ‹*Per A B C* ⟹ *False*› **by** *auto*
qed

lemma *nsams--obtuse*:
assumes $A \neq B$ **and**
$B \neq C$ **and**
¬ *SAMS A B C A B C*
shows *Obtuse A B C*
proof −
{
assume *Per A B C*
obtain A' **where** *B Midpoint A* A'
using *symmetric-point-construction* **by** *blast*
have ¬ *Col A B C*
using ‹*Per A B C*› *assms*(*1*) *assms*(*2*) *per-col-eq* **by** *blast*
have *SAMS A B C A B C*
proof −
have *C B* A' *CongA A B C*
using ‹*Per A B C*› *assms*(*1*) *assms*(*2*) *assms*(*3*) *per2--sams* **by** *blast*
moreover have ¬ *B C OS A* A'
by (*meson Col-cases* ‹*B Midpoint A* A'› *col-one-side-out l6-4-1 midpoint-bet midpoint-col*)
moreover have ¬ *A B TS C* A'
using *Col-def Midpoint-def TS-def* ‹*B Midpoint A* A'› **by** *blast*
moreover have *Coplanar A B C* A'
using ‹*Per A B C*› *assms*(*1*) *assms*(*2*) *assms*(*3*) *per2--sams* **by** *blast*
ultimately show *?thesis*
using *SAMS-def* ‹¬ *Col A B C*› *assms*(*1*) *bet-col* **by** *auto*
qed
then have *False*
using *assms*(*3*) **by** *auto*
}
{
assume *Acute A B C*
then have *SAMS A B C A B C*
by (*simp add*: *acute--sams*)
then have *False*
using *assms*(*3*) **by** *auto*
}
thus *?thesis*
using ‹*Per A B C* ⟹ *False*› *angle-partition assms*(*1*) *assms*(*2*) **by** *auto*
qed

lemma *sams2-suma2--conga*:
assumes *SAMS A B C A B C* **and**

A B C A B C SumA D E F **and**
SAMS A′ B′ C′ A′ B′ C′ **and**
A′ B′ C′ A′ B′ C′ SumA D E F
shows *A B C CongA A′ B′ C′*
proof −
have *A B C LeA A′ B′ C′* ∨ *A′ B′ C′ LeA A B C*
using *assms*(*1*) *assms*(*3*) *lea-total sams-distincts* **by** *auto*
moreover
have *A B C LeA A′ B′ C′* ⟶ *A B C CongA A′ B′ C′*
using *assms*(*2*) *assms*(*3*) *assms*(*4*) *sams-lea2-suma2--conga123* **by** *auto*
ultimately show *?thesis*
by (*meson Tarski-neutral-dimensionless.conga-sym Tarski-neutral-dimensionless.sams-lea2-suma2--conga123 Tarski-neutral-dimensi*
assms(*1*) *assms*(*2*) *assms*(*4*))
qed

lemma *acute2-suma2--conga*:
assumes *Acute A B C* **and**
A B C A B C SumA D E F **and**
Acute A′ B′ C′ **and**
A′ B′ C′ A′ B′ C′ SumA D E F
shows *A B C CongA A′ B′ C′*
proof −
have *SAMS A B C A B C*
by (*simp add*: *acute--sams assms*(*1*))
moreover have *SAMS A′ B′ C′ A′ B′ C′*
by (*simp add*: *acute--sams assms*(*3*))
ultimately show *?thesis*
using *assms*(*2*) *assms*(*4*) *sams2-suma2--conga* **by** *auto*
qed

lemma *bet2-suma--out*:
assumes *Bet A B C* **and**
Bet D E F **and**
A B C D E F SumA G H I
shows *H Out G I*
proof −
have *A B C D E F SumA A B A*
proof −
have *C B A CongA D E F*
by (*metis Bet-cases assms*(*1*) *assms*(*2*) *assms*(*3*) *conga-line suma-distincts*)
moreover have ¬ *B C OS A A*
by (*simp add*: *Col-def assms*(*1*) *col124--nos*)
moreover have *Coplanar A B C A*
using *ncop-distincts* **by** *blast*
moreover have *A B A CongA A B A*
using *calculation*(*1*) *conga-diff2 conga-trivial-1* **by** *auto*
ultimately show *?thesis*
using *SumA-def* **by** *blast*
qed
then have *A B A CongA G H I*
using *assms*(*3*) *suma2--conga* **by** *auto*
thus *?thesis*
using *eq-conga-out* **by** *auto*
qed

lemma *col2-suma--col*:
assumes *Col A B C* **and**
Col D E F **and**
A B C D E F SumA G H I
shows *Col G H I*
proof *cases*
assume *Bet A B C*
show *?thesis*
proof *cases*
assume *Bet D E F*
thus *?thesis* **using** *bet2-suma--out*

```
      by (meson ‹Bet A B C› assms(3) not-col-permutation-4 out-col)
  next
    assume ¬ Bet D E F
    show ?thesis
    proof -
      have E Out D F
        using ‹¬ Bet D E F› assms(2) or-bet-out by auto
      then have A B C CongA G H I
        using assms(3) out546-suma--conga by auto
      thus ?thesis
        using assms(1) col-conga-col by blast
    qed
  qed
next
  assume ¬ Bet A B C
  have D E F CongA G H I
  proof -
    have B Out A C
      by (simp add: ‹¬ Bet A B C› assms(1) l6-4-2)
    thus ?thesis
      using assms(3) out213-suma--conga by auto
  qed
  thus ?thesis
    using assms(2) col-conga-col by blast
qed

lemma suma-suppa--bet:
  assumes A B C SuppA D E F and
    A B C D E F SumA G H I
  shows Bet G H I
proof -
  obtain A' where P1: Bet A B A' ∧ D E F CongA C B A'
    using SuppA-def assms(1) by auto
  obtain A'' where P2: C B A'' CongA D E F ∧ ¬ B C OS A A'' ∧ Coplanar A B C A'' ∧ A B A'' CongA G H I
    using SumA-def assms(2) by auto
  have B Out A' A'' ∨ C B TS A' A''
  proof -
    have Coplanar C B A' A''
    proof -
      have Coplanar C A'' B A
        using P2 coplanar-perm-17 by blast
      moreover have B ≠ A
        using SuppA-def assms(1) by auto
      moreover have Col B A A' using P1
        by (simp add: bet-col col-permutation-4)
      ultimately show ?thesis
        using col-cop--cop coplanar-perm-3 by blast
    qed
    moreover have C B A' CongA C B A''
    proof -
      have C B A' CongA D E F
        using P1 not-conga-sym by blast
      moreover have D E F CongA C B A''
        using P2 not-conga-sym by blast
      ultimately show ?thesis
        using not-conga by blast
    qed
    ultimately show ?thesis
      using conga-cop--or-out-ts by simp
  qed
  have Bet A B A''
  proof -
    have ¬ C B TS A' A''
    proof -
      {
        assume C B TS A' A''
```

have *B C TS A A′*
proof −
{
assume *Col A B C*
then have *Col A′ C B*
using *P1 assms*(*1*) *bet-col bet-col1 col3 suppa-distincts* **by** *blast*
then have *False*
using *TS-def* ‹*C B TS A′ A″*› **by** *auto*
}
then have ¬ *Col A B C* **by** *auto*
moreover have ¬ *Col A′ B C*
using *TS-def* ‹*C B TS A′ A″*› *not-col-permutation-5* **by** *blast*
moreover
have ∃ *T*. (*Col T B C* ∧ *Bet A T A′*)
using *P1 not-col-distincts* **by** *blast*
ultimately show *?thesis*
by (*simp add*: *TS-def*)
qed
then have *B C OS A A″*
using *OS-def* ‹*C B TS A′ A″*› *invert-two-sides l9-2* **by** *blast*
then have *False*
using *P2* **by** *simp*
}
thus *?thesis* **by** *blast*
qed
then have *B Out A′ A″*
using ‹*B Out A′ A″* ∨ *C B TS A′ A″*› **by** *auto*
moreover have *A′* ≠ *B* ∧ *A″* ≠ *B* ∧ *A* ≠ *B*
using *P2 calculation conga-diff1 out-diff1 out-diff2* **by** *blast*
moreover have *Bet A′ B A*
using *P1 Bet-perm* **by** *blast*
ultimately show *?thesis*
using *bet-out--bet between-symmetry* **by** *blast*
qed
moreover have *A B A″ CongA G H I*
using *P2* **by** *blast*
ultimately show *?thesis*
using *bet-conga--bet* **by** *blast*
qed

lemma *bet-suppa--suma*:
assumes *G* ≠ *H* **and**
H ≠ *I* **and**
A B C SuppA D E F **and**
Bet G H I
shows *A B C D E F SumA G H I*
proof −
obtain *G′ H′ I′* **where** *A B C D E F SumA G′ H′ I′*
using *assms*(*3*) *ex-suma suppa-distincts* **by** *blast*
moreover have *A B C CongA A B C*
using *calculation conga-refl suma-distincts* **by** *fastforce*
moreover
have *D* ≠ *E* ∧ *E* ≠ *F*
using *assms*(*3*) *suppa-distincts* **by** *auto*
then have *D E F CongA D E F*
using *conga-refl* **by** *auto*
moreover
have *G′ H′ I′ CongA G H I*
proof −
have *G′* ≠ *H′* ∧ *H′* ≠ *I′*
using *calculation*(*1*) *suma-distincts* **by** *auto*
moreover have *Bet G′ H′ I′*
using ‹*A B C D E F SumA G′ H′ I′*› *assms*(*3*) *suma-suppa--bet* **by** *blast*
ultimately show *?thesis*
using *assms*(*1*) *assms*(*2*) *assms*(*4*) *conga-line* **by** *auto*
qed

ultimately show *?thesis*
  using *conga3-suma--suma* **by** *blast*
qed

lemma *bet-suma--suppa*:
  assumes *A B C D E F SumA G H I* **and**
    Bet G H I
  shows *A B C SuppA D E F*
proof −
  obtain *A'* **where** *C B A' CongA D E F* ∧ *A B A' CongA G H I*
    using *SumA-def assms(1)* **by** *blast*
  moreover
  have *G H I CongA A B A'*
    using *calculation not-conga-sym* **by** *blast*
  then have *Bet A B A'*
    using *assms(2) bet-conga--bet* **by** *blast*
  moreover have *D E F CongA C B A'*
    using *calculation(1) not-conga-sym* **by** *blast*
  ultimately show *?thesis*
    by (*metis SuppA-def conga-diff1*)
qed

lemma *bet2-suma--suma*:
  assumes *A'* ≠ *B* **and**
    D' ≠ *E* **and**
    Bet A B A' **and**
    Bet D E D' **and**
    A B C D E F SumA G H I
  shows *A' B C D' E F SumA G H I*
proof −
  obtain *J* **where** *P1*: *C B J CongA D E F* ∧ ¬ *B C OS A J* ∧ *Coplanar A B C J* ∧ *A B J CongA G H I*
    using *SumA-def assms(5)* **by** *auto*
  moreover
  obtain *C'* **where** *P2*: *Bet C B C'* ∧ *Cong B C' B C*
    using *segment-construction* **by** *blast*
  moreover
  have *A B C' D' E F SumA G H I*
  proof −
    have *C' B J CongA D' E F*
      by (*metis assms(2) assms(4) calculation(1) calculation(2) cong-diff-3 conga-diff1 l11-13*)
    moreover have ¬ *B C' OS A J*
      by (*metis Col-cases P1 P2 bet-col col-one-side cong-diff*)
    moreover have *Coplanar A B C' J*
      by (*smt P1 P2 bet-col bet-col1 col2-cop--cop cong-diff ncoplanar-perm-5*)
    ultimately show *?thesis*
      using *P1 SumA-def* **by** *blast*
  qed
  moreover have *A B C' CongA A' B C*
    using *assms(1) assms(3) assms(5) between-symmetry calculation(2) calculation(3) l11-14 suma-distincts* **by** *auto*
  moreover
  have *D'* ≠ *E* ∧ *E* ≠ *F*
    using *assms(2) calculation(1) conga-distinct* **by** *blast*
  then have *D' E F CongA D' E F*
    using *conga-refl* **by** *auto*
  moreover
  have *G* ≠ *H* ∧ *H* ≠ *I*
    using *assms(5) suma-distincts* **by** *blast*
  then have *G H I CongA G H I*
    using *conga-refl* **by** *auto*
  ultimately show *?thesis*
    using *conga3-suma--suma* **by** *blast*
qed

lemma *suma-suppa2--suma*:
  assumes *A B C SuppA A' B' C'* **and**
    D E F SuppA D' E' F' **and**

A B C D E F SumA G H I
shows *A′ B′ C′ D′ E′ F′ SumA G H I*
proof −
obtain *A0* **where** *P1*: *Bet A B A0* ∧ *A′ B′ C′ CongA C B A0*
using *SuppA-def assms*(*1*) **by** *auto*
obtain *D0* **where** *P2*: *Bet D E D0* ∧ *D′ E′ F′ CongA F E D0*
using *SuppA-def assms*(*2*) **by** *auto*
show *?thesis*
proof −
have *A0 B C D0 E F SumA G H I*
proof −
have *A0* ≠ *B*
using *CongA-def P1* **by** *auto*
moreover have *D0* ≠ *E*
using *CongA-def P2* **by** *blast*
ultimately show *?thesis*
using *P1 P2 assms*(*3*) *bet2-suma--suma* **by** *auto*
qed
moreover have *A0 B C CongA A′ B′ C′*
using *P1 conga-left-comm not-conga-sym* **by** *blast*
moreover have *D0 E F CongA D′ E′ F′*
using *P2 conga-left-comm not-conga-sym* **by** *blast*
moreover
have *G* ≠ *H* ∧ *H* ≠ *I*
using *assms*(*3*) *suma-distincts* **by** *blast*
then have *G H I CongA G H I*
using *conga-refl* **by** *auto*
ultimately show *?thesis*
using *conga3-suma--suma* **by** *blast*
qed
qed

lemma *suma2-obtuse2--conga*:
assumes *Obtuse A B C* **and**
A B C A B C SumA D E F **and**
Obtuse A′ B′ C′ **and**
A′ B′ C′ A′ B′ C′ SumA D E F
shows *A B C CongA A′ B′ C′*
proof −
obtain *A0* **where** *P1*: *Bet A B A0* ∧ *Cong B A0 A B*
using *segment-construction* **by** *blast*
moreover
obtain *A0′* **where** *P2*: *Bet A′ B′ A0′* ∧ *Cong B′ A0′ A′ B′*
using *segment-construction* **by** *blast*
moreover
have *A0 B C CongA A0′ B′ C′*
proof −
have *Acute A0 B C*
using *assms*(*1*) *bet-obtuse--acute P1 cong-diff-3 obtuse-distincts* **by** *blast*
moreover have *A0 B C A0 B C SumA D E F*
using *P1 acute-distincts assms*(*2*) *bet2-suma--suma calculation* **by** *blast*
moreover have *Acute A0′ B′ C′*
using *P2 assms*(*3*) *bet-obtuse--acute cong-diff-3 obtuse-distincts* **by** *blast*
moreover have *A0′ B′ C′ A0′ B′ C′ SumA D E F*
by (*metis P2 assms*(*4*) *bet2-suma--suma cong-diff-3*)
ultimately show *?thesis*
using *acute2-suma2--conga* **by** *blast*
qed
moreover have *Bet A0 B A*
using *Bet-perm calculation*(*1*) **by** *blast*
moreover have *Bet A0′ B′ A′*
using *Bet-perm calculation*(*2*) **by** *blast*
moreover have *A* ≠ *B*
using *assms*(*1*) *obtuse-distincts* **by** *blast*
moreover have *A′* ≠ *B′*
using *assms*(*3*) *obtuse-distincts* **by** *blast*

ultimately show *?thesis*
using *l11-13* **by** *blast*
qed

lemma *bet-suma2--or-conga*:
assumes $A0 \neq B$ **and**
Bet A B A0 **and**
A B C A B C SumA D E F **and**
$A' B' C' A' B' C'$ *SumA D E F*
shows *A B C CongA* $A' B' C' \vee A0$ *B C CongA* $A' B' C'$
proof –
{
fix $A' B' C'$
assume*Acute* $A' B' C' \wedge A' B' C' A' B' C'$ *SumA D E F*
have *Per A B C* $\vee$ *Acute A B C* $\vee$ *Obtuse A B C*
by (*metis angle-partition l8-20-1-R1 l8-5*)
{
assume *Per A B C*
then have *Bet D E F*
using *per2-suma--bet assms*(*3*) **by** *auto*
then have *False*
using ‹*Acute* $A' B' C' \wedge A' B' C' A' B' C'$ *SumA D E F*› *acute-suma--nbet* **by** *blast*
then have *A B C CongA* $A' B' C' \vee A0$ *B C CongA* $A' B' C'$ **by** *blast*
}
{
assume *Acute A B C*
have *Acute* $A' B' C'$
by (*simp add*: ‹*Acute* $A' B' C' \wedge A' B' C' A' B' C'$ *SumA D E F*›)
moreover have $A' B' C' A' B' C'$ *SumA D E F*
by (*simp add*: ‹*Acute* $A' B' C' \wedge A' B' C' A' B' C'$ *SumA D E F*›)
ultimately
have *A B C CongA* $A' B' C' \vee A0$ *B C CongA* $A' B' C'$
using *assms*(*3*) ‹*Acute A B C*› *acute2-suma2--conga* **by** *auto*
}
{
assume *Obtuse A B C*
have *Acute A0 B C*
using ‹*Obtuse A B C*› *assms*(*1*) *assms*(*2*) *bet-obtuse--acute* **by** *auto*
moreover have *A0 B C A0 B C SumA D E F*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *bet2-suma--suma* **by** *auto*
ultimately have *A0 B C CongA* $A' B' C'$
using ‹*Acute* $A' B' C' \wedge A' B' C' A' B' C'$ *SumA D E F*› *acute2-suma2--conga* **by** *auto*
then have *A B C CongA* $A' B' C' \vee A0$ *B C CongA* $A' B' C'$ **by** *blast*
}
then have *A B C CongA* $A' B' C' \vee A0$ *B C CongA* $A' B' C'$
using ‹*Acute A B C* $\Longrightarrow$ *A B C CongA* $A' B' C' \vee A0$ *B C CongA* $A' B' C'$› ‹*Per A B C* $\Longrightarrow$ *A B C CongA* A' $B' C' \vee A0$ *B C CongA* $A' B' C'$› ‹*Per A B C* $\vee$ *Acute A B C* $\vee$ *Obtuse A B C*› **by** *blast*
}
then have *P1*: $\forall\ A' B' C'$. (*Acute* $A' B' C' \wedge A' B' C' A' B' C'$ *SumA D E F*) $\longrightarrow$ (*A B C CongA* $A' B' C' \vee A0$ *B C CongA* $A' B' C'$) **by** *blast*
have *Per* $A' B' C' \vee$ *Acute* $A' B' C' \vee$ *Obtuse* $A' B' C'$
by (*metis angle-partition l8-20-1-R1 l8-5*)
{
assume *P2*: *Per* $A' B' C'$
have *A B C CongA* $A' B' C'$
proof –
have $A \neq B \wedge B \neq C$
using *assms*(*3*) *suma-distincts* **by** *blast*
moreover have $A' \neq B' \wedge B' \neq C'$
using *assms*(*4*) *suma-distincts* **by** *auto*
moreover have *Per A B C*
proof –
have *Bet D E F*
using *P2 assms*(*4*) *per2-suma--bet* **by** *blast*
moreover have *A B C A B C SumA D E F*
by (*simp add*: *assms*(*3*))

ultimately show *?thesis*
using *assms*(*3*) *bet-suma--per* **by** *blast*
qed
ultimately show *?thesis*
using *P2 l11-16* **by** *blast*
qed
then have *A B C CongA A′ B′ C′* ∨ *A0 B C CongA A′ B′ C′* **by** *blast*
}
{
assume *Acute A′ B′ C′*
then have *A B C CongA A′ B′ C′* ∨ *A0 B C CongA A′ B′ C′*
using *P1 assms*(*4*) **by** *blast*
}
{
assume *Obtuse A′ B′ C′*
obtain *A0′* **where** *Bet A′ B′ A0′* ∧ *Cong B′ A0′ A′ B′*
using *segment-construction* **by** *blast*
moreover
have *Acute A0′ B′ C′*
using ‹*Obtuse A′ B′ C′*› *bet-obtuse--acute calculation cong-diff-3 obtuse-distincts* **by** *blast*
moreover have *A0′ B′ C′ A0′ B′ C′ SumA D E F*
using *acute-distincts assms*(*4*) *bet2-suma--suma calculation*(*1*) *calculation*(*2*) **by** *blast*
ultimately
have *A B C CongA A′ B′ C′* ∨ *A0 B C CongA A′ B′ C′*
using *P1* **by** (*metis assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *between-symmetry l11-13 suma-distincts*)
}
thus *?thesis*
using ‹*Acute A′ B′ C′* ⟹ *A B C CongA A′ B′ C′* ∨ *A0 B C CongA A′ B′ C′*› ‹*Per A′ B′ C′* ⟹ *A B C CongA A′ B′ C′* ∨ *A0 B C CongA A′ B′ C′*› ‹*Per A′ B′ C′* ∨ *Acute A′ B′ C′* ∨ *Obtuse A′ B′ C′*› **by** *blast*
qed

lemma *suma2--or-conga-suppa*:
assumes *A B C A B C SumA D E F* **and**
A′ B′ C′ A′ B′ C′ SumA D E F
shows *A B C CongA A′ B′ C′* ∨ *A B C SuppA A′ B′ C′*
proof −
obtain *A0* **where** *P1*: *Bet A B A0* ∧ *Cong B A0 A B*
using *segment-construction* **by** *blast*
then have *A0* ≠ *B*
using *assms*(*1*) *bet-cong-eq suma-distincts* **by** *blast*
then have *A B C CongA A′ B′ C′* ∨ *A0 B C CongA A′ B′ C′*
using *assms*(*1*) *assms*(*2*) *P1 bet-suma2--or-conga* **by** *blast*
thus *?thesis*
by (*metis P1 SuppA-def cong-diff conga-right-comm conga-sym*)
qed

lemma *ex-trisuma*:
assumes *A* ≠ *B* **and**
B ≠ *C* **and**
A ≠ *C*
shows ∃ *D E F*. *A B C TriSumA D E F*
proof −
obtain *G H I* **where** *A B C B C A SumA G H I*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *ex-suma* **by** *presburger*
moreover
then obtain *D E F* **where** *G H I C A B SumA D E F*
using *ex-suma suma-distincts* **by** *presburger*
ultimately show *?thesis*
using *TriSumA-def* **by** *blast*
qed

lemma *trisuma-perm-231*:
assumes *A B C TriSumA D E F*
shows *B C A TriSumA D E F*
proof −
obtain *A1 B1 C1* **where** *P1*: *A B C B C A SumA A1 B1 C1* ∧ *A1 B1 C1 C A B SumA D E F*

using *TriSumA-def assms*(*1*) **by** *auto*
obtain *A2 B2 C2* **where** *P2*: *B C A C A B SumA B2 C2 A2*
using *P1 ex-suma suma-distincts* **by** *fastforce*
have *A B C B2 C2 A2 SumA D E F*
proof −
have *SAMS A B C B C A*
using *assms sams123231 trisuma-distincts* **by** *auto*
moreover have *SAMS B C A C A B*
using *P2 sams123231 suma-distincts* **by** *fastforce*
ultimately show *?thesis*
using *P1 P2 suma-assoc* **by** *blast*
qed
thus *?thesis*
using *P2 TriSumA-def suma-sym* **by** *blast*
qed

lemma *trisuma-perm-312*:
assumes *A B C TriSumA D E F*
shows *C A B TriSumA D E F*
by (*simp add*: *assms trisuma-perm-231*)

lemma *trisuma-perm-321*:
assumes *A B C TriSumA D E F*
shows *C B A TriSumA D E F*
proof −
obtain *G H I* **where** *A B C B C A SumA G H I* ∧ *G H I C A B SumA D E F*
using *TriSumA-def assms*(*1*) **by** *auto*
thus *?thesis*
by (*meson TriSumA-def suma-comm suma-right-comm suma-sym trisuma-perm-231*)
qed

lemma *trisuma-perm-213*:
assumes *A B C TriSumA D E F*
shows *B A C TriSumA D E F*
using *assms trisuma-perm-231 trisuma-perm-321* **by** *blast*

lemma *trisuma-perm-132*:
assumes *A B C TriSumA D E F*
shows *A C B TriSumA D E F*
using *assms trisuma-perm-312 trisuma-perm-321* **by** *blast*

lemma *conga-trisuma--trisuma*:
assumes *A B C TriSumA D E F* **and**
D E F CongA D′ E′ F′
shows *A B C TriSumA D′ E′ F′*
proof −
obtain *G H I* **where** *P1*: *A B C B C A SumA G H I* ∧ *G H I C A B SumA D E F*
using *TriSumA-def assms*(*1*) **by** *auto*
have *G H I C A B SumA D′ E′ F′*
proof −
have *f1*: $B \neq A$
by (*metis P1 suma-distincts*)
have *f2*: $C \neq A$
using *P1 suma-distincts* **by** *blast*
have *G H I CongA G H I*
by (*metis* (*full-types*) *P1 conga-refl suma-distincts*)
then show *?thesis*
using *f2 f1* **by** (*meson P1 assms*(*2*) *conga3-suma--suma conga-refl*)
qed
thus *?thesis* **using** *P1 TriSumA-def* **by** *blast*
qed

lemma *trisuma2--conga*:
assumes *A B C TriSumA D E F* **and**
A B C TriSumA D′ E′ F′
shows *D E F CongA D′ E′ F′*

proof −
  obtain *G H I* **where** *P1*: *A B C B C A SumA G H I* ∧ *G H I C A B SumA D E F*
    using *TriSumA-def assms*(*1*) **by** *auto*
  then have *P1A*: *G H I C A B SumA D E F* **by** *simp*
  obtain *G′ H′ I′* **where** *P2*: *A B C B C A SumA G′ H′ I′* ∧ *G′ H′ I′ C A B SumA D′ E′ F′*
    using *TriSumA-def assms*(*2*) **by** *auto*
  have *G′ H′ I′ C A B SumA D E F*
  proof −
    have *G H I CongA G′ H′ I′* **using** *P1 P2 suma2--conga* **by** *blast*
    moreover have *D E F CongA D E F* ∧ *C A B CongA C A B*
      by (*metis assms*(*1*) *conga-refl trisuma-distincts*)
    ultimately show *?thesis*
      by (*meson P1 conga3-suma--suma*)
  qed
  thus *?thesis*
    using *P2 suma2--conga* **by** *auto*
qed

lemma *conga3-trisuma--trisuma*:
  assumes *A B C TriSumA D E F* **and**
    A B C CongA A′ B′ C′ **and**
    B C A CongA B′ C′ A′ **and**
    C A B CongA C′ A′ B′
  shows *A′ B′ C′ TriSumA D E F*
proof −
  obtain *G H I* **where** *P1*: *A B C B C A SumA G H I* ∧ *G H I C A B SumA D E F*
    using *TriSumA-def assms*(*1*) **by** *auto*
  thus *?thesis*
  proof −
    have *A′ B′ C′ B′ C′ A′ SumA G H I*
      using *conga3-suma--suma P1* **by** (*meson assms*(*2*) *assms*(*3*) *suma2--conga*)
    moreover have *G H I C′ A′ B′ SumA D E F*
      using *conga3-suma--suma P1* **by** (*meson P1 assms*(*4*) *suma2--conga*)
    ultimately show *?thesis*
      using *TriSumA-def* **by** *blast*
  qed
qed

lemma *col-trisuma--bet*:
  assumes *Col A B C* **and**
    A B C TriSumA P Q R
  shows *Bet P Q R*
proof −
  obtain *D E F* **where** *P1*: *A B C B C A SumA D E F* ∧ *D E F C A B SumA P Q R*
    using *TriSumA-def assms*(*2*) **by** *auto*
  {
    assume *Bet A B C*
    have *A B C CongA P Q R*
    proof −
      have *A B C CongA D E F*
      proof −
        have *C* ≠ *A* ∧ *C* ≠ *B*
          using *assms*(*2*) *trisuma-distincts* **by** *blast*
        then have *C Out B A*
          using ‹ *Bet A B C*› *bet-out-1* **by** *fastforce*
        thus *?thesis*
          using *P1 out546-suma--conga* **by** *auto*
      qed
      moreover have *D E F CongA P Q R*
      proof −
        have *A* ≠ *C* ∧ *A* ≠ *B*
          using *assms*(*2*) *trisuma-distincts* **by** *blast*
        then have *A Out C B*
          using *Out-def* ‹*Bet A B C*› **by** *auto*
        thus *?thesis*
          using *P1 out546-suma--conga* **by** *auto*

```
      qed
      ultimately show ?thesis
        using conga-trans by blast
    qed
    then have Bet P Q R
      using ‹Bet A B C› bet-conga--bet by blast
  }
  {
    assume Bet B C A
    have B C A CongA P Q R
    proof –
      have B C A CongA D E F
      proof –
        have B ≠ A ∧ B ≠ C
          using assms(2) trisuma-distincts by blast
        then have B Out A C
          using Out-def ‹Bet B C A› by auto
        thus ?thesis
          using P1 out213-suma--conga by blast
      qed
      moreover have D E F CongA P Q R
      proof –
        have A ≠ C ∧ A ≠ B
          using assms(2) trisuma-distincts by auto
        then have A Out C B
          using ‹Bet B C A› bet-out-1 by auto
        thus ?thesis
          using P1 out546-suma--conga by blast
      qed
      ultimately show ?thesis
        using not-conga by blast
    qed
    then have Bet P Q R
      using ‹Bet B C A› bet-conga--bet by blast
  }
  {
    assume Bet C A B
    have E Out D F
    proof –
      have C Out B A
        using ‹Bet C A B› assms(2) bet-out l6-6 trisuma-distincts by blast
      moreover have B C A CongA D E F
      proof –
        have B ≠ A ∧ B ≠ C
          using P1 suma-distincts by blast
        then have B Out A C
          using ‹Bet C A B› bet-out-1 by auto
        thus ?thesis using out213-suma--conga P1 by blast
      qed
      ultimately show ?thesis
        using l11-21-a by blast
    qed

    then have C A B CongA P Q R
      using P1 out213-suma--conga by blast
    then have Bet P Q R
      using ‹Bet C A B› bet-conga--bet by blast
  }
  thus ?thesis
    using Col-def ‹Bet A B C ⟹ Bet P Q R› ‹Bet B C A ⟹ Bet P Q R› assms(1) by blast
qed

lemma suma-dec:
  A B C D E F SumA G H I ∨ ¬ A B C D E F SumA G H I by simp

lemma sams-dec:
```

SAMS A B C D E F ∨ ¬ *SAMS A B C D E F* **by** *simp*

lemma *trisuma-dec*:
A B C TriSumA P Q R ∨ ¬ *A B C TriSumA P Q R*
by *simp*

3.11 Parallelism

lemma *par-reflexivity*:
assumes $A \neq B$
shows *A B Par A B*
using *Par-def assms not-col-distincts* **by** *blast*

lemma *par-strict-irreflexivity*:
¬ *A B ParStrict A B*
using *ParStrict-def col-trivial-3* **by** *blast*

lemma *not-par-strict-id*:
¬ *A B ParStrict A C*
using *ParStrict-def col-trivial-1* **by** *blast*

lemma *par-id*:
assumes *A B Par A C*
shows *Col A B C*
using *Col-cases Par-def assms not-par-strict-id* **by** *auto*

lemma *par-strict-not-col-1*:
assumes *A B ParStrict C D*
shows ¬ *Col A B C*
using *Col-perm ParStrict-def assms col-trivial-1* **by** *blast*

lemma *par-strict-not-col-2*:
assumes *A B ParStrict C D*
shows ¬ *Col B C D*
using *ParStrict-def assms col-trivial-3* **by** *auto*

lemma *par-strict-not-col-3*:
assumes *A B ParStrict C D*
shows ¬ *Col C D A*
using *Col-perm ParStrict-def assms col-trivial-1* **by** *blast*

lemma *par-strict-not-col-4*:
assumes *A B ParStrict C D*
shows ¬ *Col A B D*
using *Col-perm ParStrict-def assms col-trivial-3* **by** *blast*

lemma *par-id-1*:
assumes *A B Par A C*
shows *Col B A C*
using *Par-def assms not-par-strict-id* **by** *auto*

lemma *par-id-2*:
assumes *A B Par A C*
shows *Col B C A*
using *Col-perm assms par-id-1* **by** *blast*

lemma *par-id-3*:
assumes *A B Par A C*
shows *Col A C B*
using *Col-perm assms par-id-2* **by** *blast*

lemma *par-id-4*:
assumes *A B Par A C*
shows *Col C B A*
using *Col-perm assms par-id-2* **by** *blast*

lemma *par-id-5*:
assumes *A B Par A C*
shows *Col C A B*
using *assms col-permutation-2 par-id* **by** *blast*

lemma *par-strict-symmetry*:
assumes *A B ParStrict C D*
shows *C D ParStrict A B*
using *ParStrict-def assms coplanar-perm-16* **by** *blast*

lemma *par-symmetry*:
assumes *A B Par C D*
shows *C D Par A B*
by (*smt NCol-perm Par-def assms l6-16-1 par-strict-symmetry*)

lemma *par-left-comm*:
assumes *A B Par C D*
shows *B A Par C D*
by (*metis* (*mono-tags*, *lifting*) *ParStrict-def Par-def assms ncoplanar-perm-6 not-col-permutation-5*)

lemma *par-right-comm*:
assumes *A B Par C D*
shows *A B Par D C*
using *assms par-left-comm par-symmetry* **by** *blast*

lemma *par-comm*:
assumes *A B Par C D*
shows *B A Par D C*
using *assms par-left-comm par-right-comm* **by** *blast*

lemma *par-strict-left-comm*:
assumes *A B ParStrict C D*
shows *B A ParStrict C D*
using *ParStrict-def assms ncoplanar-perm-6 not-col-permutation-5* **by** *blast*

lemma *par-strict-right-comm*:
assumes *A B ParStrict C D*
shows *A B ParStrict D C*
using *assms par-strict-left-comm par-strict-symmetry* **by** *blast*

lemma *par-strict-comm*:
assumes *A B ParStrict C D*
shows *B A ParStrict D C*
by (*simp add*: *assms par-strict-left-comm par-strict-right-comm*)

lemma *par-strict-neq1*:
assumes *A B ParStrict C D*
shows $A \neq B$
using *assms col-trivial-1 par-strict-not-col-4* **by** *blast*

lemma *par-strict-neq2*:
assumes *A B ParStrict C D*
shows $C \neq D$
using *assms col-trivial-2 par-strict-not-col-2* **by** *blast*

lemma *par-neq1*:
assumes *A B Par C D*
shows $A \neq B$
using *Par-def assms par-strict-neq1* **by** *blast*

lemma *par-neq2*:
assumes *A B Par C D*
shows $C \neq D$
using *assms par-neq1 par-symmetry* **by** *blast*

lemma *Par-cases*:

assumes *A B Par C D* ∨ *B A Par C D* ∨ *A B Par D C* ∨ *B A Par D C* ∨ *C D Par A B* ∨ *C D Par B A* ∨ *D C Par A B* ∨ *D C Par B A*
shows *A B Par C D*
using *assms par-right-comm par-symmetry* **by** *blast*

lemma *Par-perm*:
assumes *A B Par C D*
shows *A B Par C D* ∧ *B A Par C D* ∧ *A B Par D C* ∧ *B A Par D C* ∧ *C D Par A B* ∧ *C D Par B A* ∧ *D C Par A B* ∧ *D C Par B A*
using *Par-cases assms* **by** *blast*

lemma *Par-strict-cases*:
assumes *A B ParStrict C D* ∨ *B A ParStrict C D* ∨ *A B ParStrict D C* ∨ *B A ParStrict D C* ∨ *C D ParStrict A B* ∨ *C D ParStrict B A* ∨ *D C ParStrict A B* ∨ *D C ParStrict B A*
shows *A B ParStrict C D*
using *assms par-strict-right-comm par-strict-symmetry* **by** *blast*

lemma *Par-strict-perm*:
assumes *A B ParStrict C D*
shows *A B ParStrict C D* ∧ *B A ParStrict C D* ∧ *A B ParStrict D C* ∧ *B A ParStrict D C* ∧ *C D ParStrict A B* ∧ *C D ParStrict B A* ∧ *D C ParStrict A B* ∧ *D C ParStrict B A*
using *Par-strict-cases assms* **by** *blast*

lemma *l12-6*:
assumes *A B ParStrict C D*
shows *A B OS C D*
by (*metis Col-def ParStrict-def Par-strict-perm TS-def assms cop-nts--os par-strict-not-col-2*)

lemma *pars--os3412*:
assumes *A B ParStrict C D*
shows *C D OS A B*
by (*simp add*: *assms l12-6 par-strict-symmetry*)

lemma *perp-dec*:
A B Perp C D ∨ ¬ *A B Perp C D*
by *simp*

lemma *col-cop2-perp2--col*:
assumes *X1 X2 Perp A B* **and**
Y1 Y2 Perp A B **and**
Col X1 Y1 Y2 **and**
Coplanar A B X2 Y1 **and**
Coplanar A B X2 Y2
shows *Col X2 Y1 Y2*
proof *cases*
assume *X1* = *Y2*
thus *?thesis*
using *assms*(*1*) *assms*(*2*) *assms*(*4*) *cop-perp2--col not-col-permutation-2 perp-left-comm* **by** *blast*
next
assume *X1* ≠ *Y2*
then have *Y2 X1 Perp A B*
by (*metis Col-cases assms*(*2*) *assms*(*3*) *perp-col perp-comm perp-right-comm*)
then have *P1*: *X1 Y2 Perp A B*
using *Perp-perm* **by** *blast*
thus *?thesis*
proof *cases*
assume *X1* = *Y1*
thus *?thesis*
using *assms*(*1*) *assms*(*2*) *assms*(*5*) *cop-perp2--col not-col-permutation-4* **by** *blast*
next
assume *X1* ≠ *Y1*
then have *X1 Y1 Perp A B*
using *Col-cases P1 assms*(*3*) *perp-col* **by** *blast*
thus *?thesis*
using *P1 assms*(*1*) *assms*(*4*) *assms*(*5*) *col-transitivity-2 cop-perp2--col perp-not-eq-1* **by** *blast*
qed

qed

lemma *col-perp2-ncol-col*:
assumes *X1 X2 Perp A B* **and**
Y1 Y2 Perp A B **and**
Col X1 Y1 Y2 **and**
¬ *Col X1 A B*
shows *Col X2 Y1 Y2*
proof −
have *Coplanar A B X2 Y1*
proof *cases*
assume *X1* = *Y1*
thus *?thesis*
using *assms*(*1*) *ncoplanar-perm-22 perp--coplanar* **by** *blast*
next
assume *X1* ≠ *Y1*
then have *Y1 X1 Perp A B*
by (*metis Col-cases assms*(*2*) *assms*(*3*) *perp-col*)
thus *?thesis*
by (*meson assms*(*1*) *assms*(*4*) *coplanar-trans-1 ncoplanar-perm-18 ncoplanar-perm-4 perp--coplanar*)
qed
then moreover have *Coplanar A B X2 Y2*
by (*smt assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *col-cop2--cop coplanar-perm-17 coplanar-perm-18 coplanar-trans-1 perp--coplanar*)
ultimately show *?thesis*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *col-cop2-perp2--col* **by** *blast*
qed

lemma *l12-9*:
assumes
Coplanar C1 C2 A1 B1 **and**
Coplanar C1 C2 A1 B2 **and**
Coplanar C1 C2 A2 B1 **and**
Coplanar C1 C2 A2 B2 **and**
A1 A2 Perp C1 C2 **and**
B1 B2 Perp C1 C2
shows *A1 A2 Par B1 B2*
proof −
have *P1*: *A1* ≠ *A2* ∧ *C1* ≠ *C2*
using *assms*(*5*) *perp-distinct* **by** *auto*
have *P2*: *B1* ≠ *B2*
using *assms*(*6*) *perp-distinct* **by** *auto*
show *?thesis*
proof *cases*
assume *Col A1 B1 B2*
then show *?thesis*
using *P1 P2 Par-def assms*(*3*) *assms*(*4*) *assms*(*5*) *assms*(*6*) *col-cop2-perp2--col* **by** *blast*
next
assume *P3*: ¬ *Col A1 B1 B2*
{
assume ¬ *Col C1 C2 A1*
then have *Coplanar A1 A2 B1 B2*
by (*smt assms*(*1*) *assms*(*2*) *assms*(*5*) *coplanar-perm-22 coplanar-perm-8 coplanar-pseudo-trans ncop-distincts perp--coplanar*)
}
have *C1 C2 Perp A1 A2*
using *Perp-cases assms*(*5*) **by** *blast*
then have *Coplanar A1 A2 B1 B2*
by (*smt* ‹¬ *Col C1 C2 A1* ⟹ *Coplanar A1 A2 B1 B2*› *assms*(*3*) *assms*(*4*) *coplanar-perm-1 coplanar-pseudo-trans ncop-distincts perp--coplanar perp-not-col2*)
{
assume ∃ *X*. *Col X A1 A2* ∧ *Col X B1 B2*
then obtain *AB* **where** *P4*: *Col AB A1 A2* ∧ *Col AB B1 B2* **by** *auto*
then have *False*
proof *cases*
assume *AB* = *A1*

```
      thus ?thesis
        using P3 P4 by blast
    next
      assume AB ≠ A1
      then have A1 AB Perp C1 C2
        by (metis P4 assms(5) not-col-permutation-2 perp-col)
      then have AB A1 Perp C1 C2
        by (simp add: perp-left-comm)
      thus ?thesis
        using P3 P4 assms(1) assms(2) assms(6) col-cop2-perp2--col by blast
    qed
  }
  then show ?thesis
    using ParStrict-def Par-def ‹Coplanar A1 A2 B1 B2› by blast
 qed
qed

lemma parallel-existence:
 assumes A ≠ B
 shows ∃ C D. C ≠ D ∧ A B Par C D ∧ Col P C D
proof cases
 assume Col A B P
 then show ?thesis
  using Col-perm assms par-reflexivity by blast
next
 assume P1: ¬ Col A B P
 then obtain P' where P2: Col A B P' ∧ A B Perp P P'
  using l8-18-existence by blast
 then have P3: P ≠ P'
  using P1 by blast
 show ?thesis
 proof cases
  assume P4: P' = A
  have ∃ Q. Per Q P A ∧ Cong Q P A B ∧ A P OS Q B
  proof −
   have Col A P P
    using not-col-distincts by auto
   moreover have ¬ Col A P B
    by (simp add: P1 not-col-permutation-5)
   ultimately show ?thesis
    using P3 P4 assms ex-per-cong by simp
  qed
  then obtain Q where T1: Per Q P A ∧ Cong Q P A B ∧ A P OS Q B by auto
  then have T2: P ≠ Q
   using os-distincts by auto
  have T3: A B Par P Q
  proof −
   have P Q Perp P A
   proof −
    have P ≠ A
     using P3 P4 by auto
    moreover have Col P P Q
     by (simp add: col-trivial-1)
    ultimately show ?thesis
     by (metis T1 T2 Tarski-neutral-dimensionless.Perp-perm Tarski-neutral-dimensionless-axioms per-perp)
   qed
   moreover have Coplanar P A A P
    using ncop-distincts by auto
   moreover have Coplanar P A B P
    using ncop-distincts by auto
   moreover have Coplanar P A B Q
    by (metis (no-types) T1 ncoplanar-perm-7 os--coplanar)
   moreover have A B Perp P A
    using P2 P4 by auto
   ultimately show ?thesis using l12-9 ncop-distincts by blast
  qed
```

thus *?thesis*
using *T2 col-trivial-1* **by** *auto*
next
assume *T4*: $P' \neq A$
have $\exists$ *Q*. *Per Q P P'* $\wedge$ *Cong Q P A B* $\wedge$ *P' P OS Q A*
proof −
have $P' \neq P$
using *P3* **by** *auto*
moreover have $A \neq B$
by (*simp add*: *assms*)
moreover have *Col P' P P*
using *not-col-distincts* **by** *blast*
moreover have $\neg$ *Col P' P A*
by (*metis P1 P2 T4 col2--eq col-permutation-1*)
ultimately show *?thesis*
using *ex-per-cong* **by** *blast*
qed
then obtain *Q* **where** *T5*: *Per Q P P'* $\wedge$ *Cong Q P A B* $\wedge$ *P' P OS Q A* **by** *blast*
then have *T6*: $P \neq Q$
using *os-distincts* **by** *blast*
moreover have *A B Par P Q*
proof −
have *Coplanar P P' A P*
using *ncop-distincts* **by** *auto*
moreover have *Coplanar P P' A Q*
by (*meson T5 ncoplanar-perm-7 os--coplanar*)
then moreover have *Coplanar P P' B Q*
by (*smt P2 T4 col2-cop--cop col-permutation-5 col-transitivity-1 coplanar-perm-5*)
moreover have *Coplanar P P' B P*
using *ncop-distincts* **by** *auto*
moreover have *A B Perp P P'*
by (*simp add*: *P2*)
moreover have *P Q Perp P P'*
by (*metis P3 T5 T6 Tarski-neutral-dimensionless.Perp-perm Tarski-neutral-dimensionless-axioms per-perp*)
ultimately show *?thesis*
using *l12-9* **by** *blast*
qed
moreover have *Col P P Q*
by (*simp add*: *col-trivial-1*)
ultimately show *?thesis*
by *blast*
qed
qed

lemma *par-col-par*:
assumes $C \neq D'$ **and**
A B Par C D **and**
Col C D D'
shows *A B Par C D'*
proof −
{
assume *P1*: *A B ParStrict C D*
have *Coplanar A B C D'*
using *assms*(*2*) *assms*(*3*) *col2--eq col2-cop--cop par--coplanar par-neq2* **by** *blast*
then have *A B Par C D'*
by (*smt ParStrict-def Par-def P1 assms*(*1*) *assms*(*3*) *colx not-col-distincts not-col-permutation-5*)
}
{
assume $A \neq B \wedge C \neq D \wedge$ *Col A C D* $\wedge$ *Col B C D*
then have *A B Par C D'*
using *Par-def assms*(*1*) *assms*(*3*) *col2--eq col-permutation-2* **by** *blast*
}
thus *?thesis*
using *Par-def* ‹*A B ParStrict C D* $\Longrightarrow$ *A B Par C D'*› *assms*(*2*) **by** *auto*
qed

lemma *parallel-existence1*:
  assumes $A \neq B$
  shows $\exists$ *Q*. *A B Par P Q*
proof −
  obtain *C D* **where** $C \neq D \wedge$ *A B Par C D* $\wedge$ *Col P C D*
    using *assms parallel-existence* **by** *blast*
  then show *?thesis*
    by (*metis Col-cases Par-cases par-col-par*)
qed

lemma *par-not-col*:
  assumes *A B ParStrict C D* **and**
    Col X A B
  shows ¬ *Col X C D*
  using *ParStrict-def assms*(*1*) *assms*(*2*) **by** *blast*

lemma *not-strict-par1*:
  assumes *A B Par C D* **and**
    Col A B X **and**
    Col C D X
  shows *Col A B C*
  by (*smt Par-def assms*(*1*) *assms*(*2*) *assms*(*3*) *col2--eq col-permutation-2 par-not-col*)

lemma *not-strict-par2*:
  assumes *A B Par C D* **and**
    Col A B X **and**
    Col C D X
  shows *Col A B D*
  using *Par-cases assms*(*1*) *assms*(*2*) *assms*(*3*) *not-col-permutation-4 not-strict-par1* **by** *blast*

lemma *not-strict-par*:
  assumes *A B Par C D* **and**
    Col A B X **and**
    Col C D X
  shows *Col A B C* $\wedge$ *Col A B D*
  using *assms*(*1*) *assms*(*2*) *assms*(*3*) *not-strict-par1 not-strict-par2* **by** *blast*

lemma *not-par-not-col*:
  assumes $A \neq B$ **and**
    $A \neq C$ **and**
    ¬ *A B Par A C*
  shows ¬ *Col A B C*
  using *Par-def assms*(*1*) *assms*(*2*) *assms*(*3*) *not-col-distincts not-col-permutation-4* **by** *blast*

lemma *not-par-inter-uniqueness*:
  assumes $A \neq B$ **and**
    $C \neq D$ **and**
    ¬ *A B Par C D* **and**
    Col A B X **and**
    Col C D X **and**
    Col A B Y **and**
    Col C D Y
  shows $X = Y$
proof *cases*
  assume *P1*: $C = Y$
  thus *?thesis*
  proof *cases*
    assume *P2*: $C = X$
    thus *?thesis*
      using *P1* **by** *auto*
  next
    assume $C \neq X$
    thus *?thesis*
     by (*smt Par-def assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *assms*(*6*) *assms*(*7*) *col3 col-permutation-5 l6-21*)
  qed
next

assume *C* ≠ *Y*
thus *?thesis*
by (*smt Par-def assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *assms*(*6*) *assms*(*7*) *col-permutation-2 col-permutation-4 l6-21*)
qed

lemma *inter-uniqueness-not-par*:
assumes ¬ *Col A B C* **and**
Col A B P **and**
Col C D P
shows ¬ *A B Par C D*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *not-strict-par1* **by** *blast*

lemma *col-not-col-not-par*:
assumes ∃ *P*. *Col A B P* ∧ *Col C D P* **and**
∃ *Q*. *Col C D Q* ∧ ¬*Col A B Q*
shows ¬ *A B Par C D*
using *assms*(*1*) *assms*(*2*) *colx not-strict-par par-neq2* **by** *blast*

lemma *par-distincts*:
assumes *A B Par C D*
shows *A B Par C D* ∧ *A* ≠ *B* ∧ *C* ≠ *D*
using *assms par-neq1 par-neq2* **by** *blast*

lemma *par-not-col-strict*:
assumes *A B Par C D* **and**
Col C D P **and**
¬ *Col A B P*
shows *A B ParStrict C D*
using *Col-cases Par-def assms*(*1*) *assms*(*2*) *assms*(*3*) *col3* **by** *blast*

lemma *col-cop-perp2-pars*:
assumes ¬ *Col A B P* **and**
Col C D P **and**
Coplanar A B C D **and**
A B Perp P Q **and**
C D Perp P Q
shows *A B ParStrict C D*
proof −
have *P1*: *C* ≠ *D*
using *assms*(*5*) *perp-not-eq-1* **by** *auto*
then have *P2*: *Coplanar A B C P*
using *col-cop--cop assms*(*2*) *assms*(*3*) **by** *blast*
moreover have *P3*: *Coplanar A B D P* **using** *col-cop--cop*
using *P1 assms*(*2*) *assms*(*3*) *col2-cop--cop col-trivial-2* **by** *blast*
have *A B Par C D*
proof −
have *Coplanar P A Q C*
proof −
have ¬ *Col B P A*
by (*simp add*: *assms*(*1*) *not-col-permutation-1*)
moreover have *Coplanar B P A Q*
by (*meson assms*(*4*) *ncoplanar-perm-12 perp--coplanar*)
moreover have *Coplanar B P A C*
using *P2 ncoplanar-perm-13* **by** *blast*
ultimately show *?thesis*
using *coplanar-trans-1* **by** *auto*
qed
then have *P4*: *Coplanar P Q A C*
using *ncoplanar-perm-2* **by** *blast*
have *Coplanar P A Q D*
proof −
have ¬ *Col B P A*
by (*simp add*: *assms*(*1*) *not-col-permutation-1*)
moreover have *Coplanar B P A Q*
by (*meson assms*(*4*) *ncoplanar-perm-12 perp--coplanar*)

moreover have *Coplanar B P A D*
using *P3 ncoplanar-perm-13* **by** *blast*
ultimately show *?thesis*
using *coplanar-trans-1* **by** *blast*
qed
then moreover have *Coplanar P Q A D*
using *ncoplanar-perm-2* **by** *blast*
moreover have *Coplanar P Q B C*
using *P2 assms*(*1*) *assms*(*4*) *coplanar-perm-1 coplanar-perm-10 coplanar-trans-1 perp--coplanar* **by** *blast*
moreover have *Coplanar P Q B D*
by (*meson P3 assms*(*1*) *assms*(*4*) *coplanar-trans-1 ncoplanar-perm-1 ncoplanar-perm-13 perp--coplanar*)
ultimately show *?thesis*
using *assms*(*4*) *assms*(*5*) *l12-9 P4* **by** *auto*
qed
thus *?thesis*
using *assms*(*1*) *assms*(*2*) *par-not-col-strict* **by** *auto*
qed

lemma *all-one-side-par-strict*:
assumes $C \neq D$ **and**
$\forall$ *P*. *Col C D P* $\longrightarrow$ *A B OS C P*
shows *A B ParStrict C D*
proof −
have *P1*: *Coplanar A B C D*
by (*simp add*: *assms*(*2*) *col-trivial-2 os--coplanar*)
{
assume $\exists$ *X*. *Col X A B* $\wedge$ *Col X C D*
then obtain *X* **where** *P2*: *Col X A B* $\wedge$ *Col X C D* **by** *blast*
have *A B OS C X*
by (*simp add*: *P2 Col-perm assms*(*2*))
then obtain *M* **where** *A B TS C M* $\wedge$ *A B TS X M*
by (*meson Col-cases P2 col124--nos*)
then have *False*
using *P2 TS-def* **by** *blast*
}
thus *?thesis*
using *P1 ParStrict-def* **by** *auto*
qed

lemma *par-col-par-2*:
assumes $A \neq P$ **and**
Col A B P **and**
A B Par C D
shows *A P Par C D*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *par-col-par par-symmetry* **by** *blast*

lemma *par-col2-par*:
assumes $E \neq F$ **and**
A B Par C D **and**
Col C D E **and**
Col C D F
shows *A B Par E F*
by (*metis assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *col-transitivity-2 not-col-permutation-4 par-col-par par-distincts par-right-comm*)

lemma *par-col2-par-bis*:
assumes $E \neq F$ **and**
A B Par C D **and**
Col E F C **and**
Col E F D
shows *A B Par E F*
by (*metis assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *col-transitivity-1 not-col-permutation-2 par-col2-par*)

lemma *par-strict-col-par-strict*:
assumes $C \neq E$ **and**
A B ParStrict C D **and**

Col C D E
shows *A B ParStrict C E*
proof −
have *P1*: *C E Par A B*
using *Par-def Par-perm assms*(*1*) *assms*(*2*) *assms*(*3*) *par-col-par-2* **by** *blast*
{
assume *C E ParStrict A B*
then have *A B ParStrict C E*
by (*metis par-strict-symmetry*)
}
thus *?thesis*
using *Col-cases Par-def P1 assms*(*2*) *par-strict-not-col-1* **by** *blast*
qed

lemma *par-strict-col2-par-strict*:
assumes *E* ≠ *F* **and**
A B ParStrict C D **and**
Col C D E **and**
Col C D F
shows *A B ParStrict E F*
by (*smt ParStrict-def assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *col2-cop--cop colx not-col-permutation-1 par-strict-neq1 par-strict-symmetry*)

lemma *line-dec*:
(*Col C1 B1 B2* ∧ *Col C2 B1 B2*) ∨ ¬ (*Col C1 B1 B2* ∧ *Col C2 B1 B2*)
by *simp*

lemma *par-distinct*:
assumes *A B Par C D*
shows *A* ≠ *B* ∧ *C* ≠ *D*
using *assms par-neq1 par-neq2* **by** *auto*

lemma *par-col4--par*:
assumes *E* ≠ *F* **and**
G ≠ *H* **and**
A B Par C D **and**
Col A B E **and**
Col A B F **and**
Col C D G **and**
Col C D H
shows *E F Par G H*
proof −
have *C D Par E F*
using *Par-cases assms*(*1*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *par-col2-par* **by** *blast*
then have *E F Par C D*
by (*simp add*: ‹*C D Par E F*› *par-symmetry*)
thus *?thesis*
using *assms*(*2*) *assms*(*6*) *assms*(*7*) *par-col2-par* **by** *blast*
qed

lemma *par-strict-col4--par-strict*:
assumes *E* ≠ *F* **and**
G ≠ *H* **and**
A B ParStrict C D **and**
Col A B E **and**
Col A B F **and**
Col C D G **and**
Col C D H
shows *E F ParStrict G H*
proof −
have *C D ParStrict E F*
using *Par-strict-cases assms*(*1*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *par-strict-col2-par-strict* **by** *blast*
then have *E F ParStrict C D*
by (*simp add*: ‹*C D ParStrict E F*› *par-strict-symmetry*)
thus *?thesis*
using *assms*(*2*) *assms*(*6*) *assms*(*7*) *par-strict-col2-par-strict* **by** *blast*

qed

lemma *par-strict-one-side*:
  assumes *A B ParStrict C D* **and**
    Col C D P
  shows *A B OS C P*
proof *cases*
  assume $C = P$
  thus *?thesis*
    using *assms*(*1*) *assms*(*2*) *not-col-permutation-5 one-side-reflexivity par-not-col* **by** *blast*
next
  assume $C \neq P$
  thus *?thesis*
    using *assms*(*1*) *assms*(*2*) *l12-6 par-strict-col-par-strict* **by** *blast*
qed

lemma *par-strict-all-one-side*:
  assumes *A B ParStrict C D*
  shows ∀ *P*. *Col C D P* ⟶ *A B OS C P*
  using *assms par-strict-one-side* **by** *blast*

lemma *inter-trivial*:
  assumes ¬ *Col A B X*
  shows *X Inter A X B X*
  by (*metis Col-perm Inter-def assms col-trivial-1*)

lemma *inter-sym*:
  assumes *X Inter A B C D*
  shows *X Inter C D A B*
proof −
  obtain *P* **where** *P1*: *Col P C D* ∧ ¬ *Col P A B*
    using *Inter-def assms* **by** *auto*
  have *P2*: $A \neq B$
    using *P1 col-trivial-2* **by** *blast*
  then show *?thesis*
  proof *cases*
    assume $A = X$
    have *Col B A B*
      by (*simp add*: *col-trivial-3*)
    {
      assume *P3*: *Col B C D*
      have *Col P A B*
      proof −
        have $C \neq D$
          using *Inter-def assms* **by** *blast*
        moreover have *Col C D P*
          using *P1 not-col-permutation-2* **by** *blast*
        moreover have *Col C D A*
          using *Inter-def* ‹$A = X$› *assms* **by** *auto*
        moreover have *Col C D B*
          using *P3 not-col-permutation-2* **by** *blast*
        ultimately show *?thesis*
          using *col3* **by** *blast*
      qed
      then have *False*
        by (*simp add*: *P1*)
    }
    then have ¬ *Col B C D* **by** *auto*
    then show *?thesis*
      using *Inter-def P2 assms* **by** (*meson col-trivial-3*)
  next
    assume *P5*: $A \neq X$
    have *P6*: *Col A A B*
      using *not-col-distincts* **by** *blast*
    {
      assume *P7*: *Col A C D*

have *Col A P X*
proof −
have *C* ≠ *D*
using *Inter-def assms* **by** *auto*
moreover have *Col C D A*
using *Col-cases P7* **by** *blast*
moreover have *Col C D P*
using *Col-cases P1* **by** *auto*
moreover have *Col C D X*
using *Inter-def assms* **by** *auto*
ultimately show *?thesis*
using *col3* **by** *blast*
qed
then have *Col P A B*
by (*metis* (*full-types*) *Col-perm Inter-def P5 assms col-transitivity-2*)
then have *False*
by (*simp add*: *P1*)
}
then have ¬ *Col A C D* **by** *auto*
then show *?thesis*
by (*meson Inter-def P2 assms col-trivial-1*)
qed
qed

lemma *inter-left-comm*:
assumes *X Inter A B C D*
shows *X Inter B A C D*
using *Col-cases Inter-def assms* **by** *auto*

lemma *inter-right-comm*:
assumes *X Inter A B C D*
shows *X Inter A B D C*
by (*metis assms inter-left-comm inter-sym*)

lemma *inter-comm*:
assumes *X Inter A B C D*
shows *X Inter B A D C*
using *assms inter-left-comm inter-right-comm* **by** *blast*

lemma *l12-17*:
assumes *A* ≠ *B* **and**
P Midpoint A C **and**
P Midpoint B D
shows *A B Par C D*
proof *cases*
assume *P1*: *Col A B P*
thus *?thesis*
proof *cases*
assume *A* = *P*
thus *?thesis*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *cong-diff-2 is-midpoint-id midpoint-col midpoint-cong not-par-not-col* **by** *blast*
next
assume *P2*: *A* ≠ *P*
thus *?thesis*
proof *cases*
assume *B* = *P*
thus *?thesis*
by (*metis assms*(*1*) *assms*(*2*) *assms*(*3*) *midpoint-col midpoint-distinct-2 midpoint-distinct-3 not-par-not-col par-comm*)
next
assume *P3*: *B* ≠ *P*
have *P4*: *Col B P D*
using *assms*(*3*) *midpoint-col not-col-permutation-4* **by** *blast*
have *P5*: *Col A P C*
using *assms*(*2*) *midpoint-col not-col-permutation-4* **by** *blast*
then have *P6*: *Col B C P*

using *P1 P2 col-transitivity-2 not-col-permutation-3 not-col-permutation-5* **by** *blast*
have $C \neq D$
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *l7-9* **by** *blast*
moreover have *Col A C D*
using *P1 P3 P4 P6 col3 not-col-permutation-3 not-col-permutation-5* **by** *blast*
moreover have *Col B C D*
using *P3 P4 P6 col-trivial-3 colx* **by** *blast*
ultimately show *?thesis*
by (*simp add*: *Par-def assms*(*1*))
qed
qed
next
assume *T1*: ¬ *Col A B P*
then obtain *E* **where** *T2*: *Col A B E* ∧ *A B Perp P E*
using *l8-18-existence* **by** *blast*
have *T3*: $A \neq P$
using *T1 col-trivial-3* **by** *blast*
then show *?thesis*
proof *cases*
assume *T4*: $A = E$
then have *T5*: *Per P A B*
using *T2 l8-2 perp-per-1* **by** *blast*
obtain B' **where** *T6*: *Bet B A* B' ∧ *Cong A* B' *B A*
using *segment-construction* **by** *blast*
obtain D' **where** *T7*: *Bet* B' *P* D' ∧ *Cong P* D' B' *P*
using *segment-construction* **by** *blast*
have *T8*: *C Midpoint D* D'
using *T6 T7 assms*(*2*) *assms*(*3*) *midpoint-def not-cong-3412 symmetry-preserves-midpoint* **by** *blast*
have *Col A B* B'
using *Col-cases Col-def T6* **by** *blast*
then have *T9*: *Per P A* B'
using *per-col T5 assms*(*1*) **by** *blast*
obtain B'' **where** *T10*: *A Midpoint B* B'' ∧ *Cong P B P* B''
using *Per-def T5* **by** *auto*
then have $B' = B''$
using *T6 cong-symmetry midpoint-def symmetric-point-uniqueness* **by** *blast*
then have *Cong P D P* D'
by (*metis Cong-perm Midpoint-def T10 T7 assms*(*3*) *cong-inner-transitivity*)
then have *T12*: *Per P C D*
using *Per-def T8* **by** *auto*
then have *T13*: *C PerpAt P C C D*
by (*metis T3 assms*(*1*) *assms*(*2*) *assms*(*3*) *l7-3-2 per-perp-in sym-preserve-diff*)
have *T14*: $P \neq C$
using *T3 assms*(*2*) *is-midpoint-id-2* **by** *auto*
have *T15*: $C \neq D$
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *l7-9* **by** *auto*
have *T15A*: *C C Perp C D* ∨ *P C Perp C D*
using *T12 T14 T15 per-perp* **by** *auto*
{
assume *C C Perp C D*
then have *A B Par C D*
using *perp-distinct* **by** *auto*
}
{
assume *P C Perp C D*
have *A B Par C D*
proof −
have *Coplanar P A A C*
using *ncop-distincts* **by** *blast*
moreover have *Coplanar P A A D*
using *ncop-distincts* **by** *blast*
moreover have *Coplanar P A B C*
by (*simp add*: *assms*(*2*) *coplanar-perm-1 midpoint--coplanar*)
moreover have *Coplanar P A B D*
using *assms*(*3*) *midpoint-col ncop--ncols* **by** *blast*
moreover have *A B Perp P A*

using *T2 T4* **by** *auto*
moreover have *C D Perp P A*
proof −
have *P A Perp C D*
proof −
have $P \neq A$
using *T3* **by** *auto*
moreover have *P C Perp C D*
using *T14 T15 T12 per-perp* **by** *blast*
moreover have *Col P C A*
by (*simp add*: *assms*(*2*) *l7-2 midpoint-col*)
ultimately show *?thesis*
using *perp-col* **by** *blast*
qed
then show *?thesis*
using *Perp-perm* **by** *blast*
qed
ultimately show *?thesis* **using** *l12-9* **by** *blast*
qed
}
then show *?thesis* **using** *T15A*
using ‹$C\ C\ Perp\ C\ D \Longrightarrow A\ B\ Par\ C\ D$› **by** *blast*
next
assume *S1B*: $A \neq E$
obtain *F* **where** *S2*: $Bet\ E\ P\ F \wedge Cong\ P\ F\ E\ P$
using *segment-construction* **by** *blast*
then have *S2A*: *P Midpoint E F*
using *midpoint-def not-cong-3412* **by** *blast*
then have *S3*: *Col C D F*
using *T2 assms*(*2*) *assms*(*3*) *mid-preserves-col* **by** *blast*
obtain A' **where** *S4*: $Bet\ A\ E\ A' \wedge Cong\ E\ A'\ A\ E$
using *segment-construction* **by** *blast*
obtain C' **where** *S5*: $Bet\ A'\ P\ C' \wedge Cong\ P\ C'\ A'\ P$
using *segment-construction* **by** *blast*
have *S6*: $F\ Midpoint\ C\ C'$
using *S4 S5 S2A assms*(*2*) *midpoint-def not-cong-3412 symmetry-preserves-midpoint* **by** *blast*
have *S7*: *Per P E A*
using *T2 col-trivial-3 l8-16-1* **by** *blast*
have *S8*: $Cong\ P\ C\ P\ C'$
proof −
have *Cong P C P A*
using *Cong-perm Midpoint-def assms*(*2*) **by** *blast*
moreover have $Cong\ P\ A\ P\ C'$
proof −
obtain A'' **where** *S9*: $E\ Midpoint\ A\ A'' \wedge Cong\ P\ A\ P\ A''$
using *Per-def S7* **by** *blast*
have *S10*: $A' = A''$
using *Cong-perm Midpoint-def S4 S9 symmetric-point-uniqueness* **by** *blast*
then have $Cong\ P\ A\ P\ A'$ **using** *S9* **by** *auto*
moreover have $Cong\ P\ A'\ P\ C'$
using *Cong-perm S5* **by** *blast*
ultimately show *?thesis*
using *cong-transitivity* **by** *blast*
qed
ultimately show *?thesis*
using *cong-transitivity* **by** *blast*
qed
then have *S9*: *Per P F C*
using *S6 Per-def* **by** *blast*
then have *F PerpAt P F F C*
by (*metis S2 S2A T1 T2 S1B assms*(*2*) *cong-diff-3 l7-9 per-perp-in*)
then have *F PerpAt F P C F*
using *Perp-in-perm* **by** *blast*
then have *S10*: $F\ P\ Perp\ C\ F \vee F\ F\ Perp\ C\ F$
using *l8-15-2 perp-in-col* **by** *blast*
{

assume *S11*: *F P Perp C F*
have *Coplanar P E A C*
proof −
have *Col P E P* ∧ *Col A C P*
using *assms*(*2*) *col-trivial-3 midpoint-col not-col-permutation-2* **by** *blast*
then show *?thesis*
using *Coplanar-def* **by** *blast*
qed
moreover have *Coplanar P E A D*
proof −
have *Col P D B* ∧ *Col E A B*
using *Mid-cases T2 assms*(*3*) *midpoint-col not-col-permutation-1* **by** *blast*
then show *?thesis*
using *Coplanar-def* **by** *blast*
qed
moreover have *Coplanar P E B C*
by (*metis S1B T2 calculation*(*1*) *col2-cop--cop col-transitivity-1 ncoplanar-perm-5 not-col-permutation-5*)
moreover have *Coplanar P E B D*
by (*metis S1B T2 calculation*(*2*) *col2-cop--cop col-transitivity-1 ncoplanar-perm-5 not-col-permutation-5*)
moreover have *C D Perp P E*
proof −
have *C* ≠ *D*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *sym-preserve-diff* **by** *blast*
moreover have *P F Perp C F*
using *Perp-perm S11* **by** *blast*
moreover have *Col P F E*
by (*simp add*: *Col-def S2*)
moreover have *Col C F D*
using *Col-perm S3* **by** *blast*
ultimately show *?thesis* **using** *per-col*
by (*smt Perp-cases S2 col-trivial-3 cong-diff perp-col4 perp-not-eq-1*)
qed
ultimately have *A B Par C D*
using *T2 l12-9* **by** *blast*
}
{
assume *F F Perp C F*
then have *A B Par C D*
using *perp-distinct* **by** *blast*
}
thus *?thesis*
using *S10* ‹*F P Perp C F* ⟹ *A B Par C D*› **by** *blast*
qed
qed

lemma *l12-18-a*:
assumes *Cong A B C D* **and**
Cong B C D A **and**
¬ *Col A B C* **and**
B ≠ *D* **and**
Col A P C **and**
Col B P D
shows *A B Par C D*
proof −
have *P Midpoint A C* ∧ *P Midpoint B D*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *assms*(*6*) *l7-21* **by** *blast*
then show *?thesis*
using *assms*(*3*) *l12-17 not-col-distincts* **by** *blast*
qed

lemma *l12-18-b*:
assumes *Cong A B C D* **and**
Cong B C D A **and**
¬ *Col A B C* **and**
B ≠ *D* **and**
Col A P C **and**

Col B P D
shows *B C Par D A*
by (*smt assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *assms*(*6*) *cong-symmetry inter-uniqueness-not-par l12-18-a l6-21 not-col-distincts*)

lemma *l12-18-c*:
assumes *Cong A B C D* **and**
Cong B C D A **and**
¬ *Col A B C* **and**
B ≠ *D* **and**
Col A P C **and**
Col B P D
shows *B D TS A C*
proof −
have *P Midpoint A C* ∧ *P Midpoint B D*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *assms*(*6*) *l7-21* **by** *blast*
then show *?thesis*
proof −
have *A C TS B D*
by (*metis Col-cases Tarski-neutral-dimensionless.mid-two-sides Tarski-neutral-dimensionless-axioms* ‹*P Midpoint A C* ∧ *P Midpoint B D*› *assms*(*3*))
then have ¬ *Col B D A*
by (*meson Col-cases Tarski-neutral-dimensionless.mid-preserves-col Tarski-neutral-dimensionless.ts--ncol Tarski-neutral-dimensiona*
‹*P Midpoint A C* ∧ *P Midpoint B D*› *l7-2*)
then show *?thesis*
by (*meson Tarski-neutral-dimensionless.mid-two-sides Tarski-neutral-dimensionless-axioms* ‹*P Midpoint A C* ∧ *P Midpoint B D*›)
qed
qed

lemma *l12-18-d*:
assumes *Cong A B C D* **and**
Cong B C D A **and**
¬ *Col A B C* **and**
B ≠ *D* **and**
Col A P C **and**
Col B P D
shows *A C TS B D*
by (*metis* (*no-types*, *lifting*) *Col-cases TS-def assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *assms*(*6*) *l12-18-c not-col-distincts not-cong-2143 not-cong-4321*)

lemma *l12-18*:
assumes *Cong A B C D* **and**
Cong B C D A **and**
¬ *Col A B C* **and**
B ≠ *D* **and**
Col A P C **and**
Col B P D
shows *A B Par C D* ∧ *B C Par D A* ∧ *B D TS A C* ∧ *A C TS B D*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *assms*(*6*) *l12-18-a l12-18-b l12-18-c l12-18-d* **by** *auto*

lemma *par-two-sides-two-sides*:
assumes *A B Par C D* **and**
B D TS A C
shows *A C TS B D*
by (*metis Par-def TS-def assms*(*1*) *assms*(*2*) *invert-one-side invert-two-sides l12-6 l9-31 not-col-permutation-4 one-side-symmetry os-ts1324--os pars--os3412*)

lemma *par-one-or-two-sides*:
assumes *A B ParStrict C D*
shows (*A C TS B D* ∧ *B D TS A C*) ∨ (*A C OS B D* ∧ *B D OS A C*)
by (*smt Par-def assms invert-one-side l12-6 l9-31 not-col-permutation-3 os-ts1324--os par-strict-not-col-1 par-strict-not-col-2 par-two-sides-two-sides pars--os3412 two-sides-cases*)

lemma *l12-21-b*:
assumes *A C TS B D* **and**

B A C CongA D C A
shows *A B Par C D*
proof −
have *P1*: ¬ *Col A B C*
using *TS-def assms*(*1*) *not-col-permutation-4* **by** *blast*
then have *P2*: $A \neq B$
using *col-trivial-1* **by** *auto*
have *P3*: $C \neq D$
using *assms*(*1*) *ts-distincts* **by** *blast*
then obtain D' **where** *P4*: *C Out D D'* ∧ *Cong C D' A B*
using *P2 segment-construction-3* **by** *blast*
have *P5*: *B A C CongA D' C A*
proof −
have *A Out B B*
using *P2 out-trivial* **by** *auto*
moreover have *A Out C C*
using *P1 col-trivial-3 out-trivial* **by** *force*
moreover have *C Out D' D*
by (*simp add*: *P4 l6-6*)
moreover have *C Out A A*
using *P1 not-col-distincts out-trivial* **by** *auto*
ultimately show *?thesis*
using *assms*(*2*) *l11-10* **by** *blast*
qed
then have *P6*: *Cong D' A B C*
using *Cong-perm P4 cong-pseudo-reflexivity l11-49* **by** *blast*
have *P7*: *A C TS D' B*
proof −
have *A C TS D B*
by (*simp add*: *assms*(*1*) *l9-2*)
moreover have *Col C A C*
using *col-trivial-3* **by** *auto*
ultimately show *?thesis*
using *P4 l9-5* **by** *blast*
qed
then obtain *M* **where** *P8*: *Col M A C* ∧ *Bet D' M B*
using *TS-def* **by** *blast*
have $B \neq D'$
using *P7 not-two-sides-id* **by** *blast*
then have *M Midpoint A C* ∧ *M Midpoint B D'*
by (*metis Col-cases P1 P4 P6 P8 bet-col l7-21 not-cong-3412*)
then have *A B Par C D'*
using *P2 l12-17* **by** *blast*
thus *?thesis*
by (*meson P3 P4 Tarski-neutral-dimensionless.par-col-par Tarski-neutral-dimensionless-axioms l6-6 out-col*)
qed

lemma *l12-22-aux*:
assumes $P \neq A$ **and**
$A \neq C$ **and**
Bet P A C **and**
P A OS B D **and**
B A P CongA D C P
shows *A B Par C D*
proof −
have *P1*: $P \neq C$
using *CongA-def assms*(*5*) **by** *blast*
obtain B' **where** *P2*: *Bet B A B'* ∧ *Cong A B' B A*
using *segment-construction* **by** *blast*
have *P3*: *P A B CongA C A B'*
by (*metis CongA-def P2 assms*(*2*) *assms*(*3*) *assms*(*5*) *cong-reverse-identity l11-14*)
have *P4*: *D C A CongA D C P*
by (*metis Col-def assms*(*2*) *assms*(*3*) *assms*(*4*) *bet-out-1 col124--nos l6-6 out2--conga out-trivial*)
have *P5*: *A B' Par C D*
proof −
have ¬ *Col B P A*

using *assms*(*4*) *col123--nos not-col-permutation-2* **by** *blast*
then have *P A TS B B′*
by (*metis P2 assms*(*4*) *bet--ts cong-reverse-identity invert-two-sides not-col-permutation-3 os-distincts*)
then have *A C TS B′ D*
by (*meson assms*(*2*) *assms*(*3*) *assms*(*4*) *bet-col bet-col1 col-preserves-two-sides l9-2 l9-8-2*)
moreover have *B′ A C CongA D C A*
proof −
have *B′ A C CongA B A P*
by (*simp add*: *P3 conga-comm conga-sym*)
moreover have *B A P CongA D C A*
using *P4 assms*(*5*) *not-conga not-conga-sym* **by** *blast*
ultimately show *?thesis*
using *not-conga* **by** *blast*
qed
ultimately show *?thesis*
using *l12-21-b* **by** *blast*
qed
have *C D Par A B*
proof −
have *A ≠ B*
using *assms*(*4*) *os-distincts* **by** *blast*
moreover have *C D Par A B′*
using *P5 par-symmetry* **by** *blast*
moreover have *Col A B′ B*
by (*simp add*: *Col-def P2*)
ultimately show *?thesis*
using *par-col-par* **by** *blast*
qed
thus *?thesis*
using *Par-cases* **by** *blast*
qed

lemma *l12-22-b*:
assumes *P Out A C* **and**
P A OS B D **and**
B A P CongA D C P
shows *A B Par C D*
proof *cases*
assume *A = C*
then show *?thesis*
using *assms*(*2*) *assms*(*3*) *conga-comm conga-os--out not-par-not-col os-distincts out-col* **by** *blast*
next
assume *P1*: *A ≠ C*
{
assume *Bet P A C*
then have *A B Par C D*
using *P1 assms*(*2*) *assms*(*3*) *conga-diff2 l12-22-aux* **by** *blast*
}
{
assume *P2*: *Bet P C A*
have *C D Par A B*
proof −
have *P C OS D B*
using *assms*(*1*) *assms*(*2*) *col-one-side one-side-symmetry out-col out-diff2* **by** *blast*
moreover have *D C P CongA B A P*
using *assms*(*3*) *not-conga-sym* **by** *blast*
then show *?thesis*
by (*metis P1 P2 assms*(*1*) *calculation l12-22-aux out-distinct*)
qed
then have *A B Par C D*
using *Par-cases* **by** *auto*
}
then show *?thesis*
using *Out-def ‹Bet P A C ⟹ A B Par C D› assms*(*1*) **by** *blast*
qed

lemma *par-strict-par*:
  assumes *A B ParStrict C D*
  shows *A B Par C D*
  using *Par-def assms* **by** *auto*

lemma *par-strict-distinct*:
  assumes *A B ParStrict C D*
  shows $A \neq B \wedge C \neq D$
  using *assms par-strict-neq1 par-strict-neq2* **by** *auto*

lemma *col-par*:
  assumes $A \neq B$ **and**
    $B \neq C$ **and**
    Col A B C
  shows *A B Par B C*
  by (*simp add*: *Par-def assms*(*1*) *assms*(*2*) *assms*(*3*) *col-trivial-1*)

lemma *acute-col-perp--out*:
  assumes *Acute A B C* **and**
    Col B C A' **and**
    B C Perp A A'
  shows *B Out A' C*
proof −
  {
    assume *P1*: ¬ *Col B C A*
    then obtain *B'* **where** *P2*: *B C Perp B' B* ∧ *B C OS A B'*
      using *assms*(*2*) *l10-15 os-distincts* **by** *blast*
    have *P3*: ¬ *Col B' B C*
      using *P2 col124--nos col-permutation-1* **by** *blast*
    {
      assume *Col B B' A*
      then have *A B C LtA A B C*
        using *P2 acute-one-side-aux acute-sym assms*(*1*) *one-side-not-col124* **by** *blast*
      then have *False*
        by (*simp add*: *nlta*)
    }
    then have *P4*: ¬ *Col B B' A* **by** *auto*
    have *P5*: *B B' ParStrict A A'*
    proof −
      have *B B' Par A A'*
      proof −
        have *Coplanar B C B A*
          using *ncop-distincts* **by** *blast*
        moreover have *Coplanar B C B A'*
          using *ncop-distincts* **by** *blast*
        moreover have *Coplanar B C B' A*
          using *P2 coplanar-perm-1 os--coplanar* **by** *blast*
        moreover have *Coplanar B C B' A'*
          using *assms*(*2*) *ncop--ncols* **by** *auto*
        moreover have *B B' Perp B C*
          using *P2 Perp-perm* **by** *blast*
        moreover have *A A' Perp B C*
          using *Perp-perm assms*(*3*) **by** *blast*
        ultimately show *?thesis*
          using *l12-9* **by** *auto*
      qed
      moreover have *Col A A' A*
        by (*simp add*: *col-trivial-3*)
      moreover have ¬ *Col B B' A*
        by (*simp add*: *P4*)
      ultimately show *?thesis*
        using *par-not-col-strict* **by** *auto*
    qed
    then have *P6*: ¬ *Col B B' A'*
      using *P5 par-strict-not-col-4* **by** *auto*
    then have *B B' OS A' C*

```
    proof -
      have B B' OS A' A
        using P5 l12-6 one-side-symmetry by blast
      moreover have B B' OS A C
        using P2 acute-one-side-aux acute-sym assms(1) one-side-symmetry by blast
      ultimately show ?thesis
        using one-side-transitivity by blast
    qed
    then have B Out A' C
      using Col-cases assms(2) col-one-side-out by blast
  }
  then show ?thesis
    using assms(2) assms(3) perp-not-col2 by blast
qed

lemma acute-col-perp--out-1:
  assumes Acute A B C and
    Col B C A' and
    B A Perp A A'
  shows B Out A' C
proof -
  obtain A0 where P1: Bet A B A0 ∧ Cong B A0 A B
    using segment-construction by blast
  obtain C0 where P2: Bet C B C0 ∧ Cong B C0 C B
    using segment-construction by blast
  have P3: ¬ Col B A A'
    using assms(3) col-trivial-2 perp-not-col2 by blast
  have Bet A' B C0
  proof -
    have P4: Col A' B C0
      using P2 acute-distincts assms(1) assms(2) bet-col col-transitivity-2 not-col-permutation-4 by blast
    {
      assume P5: B Out A' C0
      have B Out A A0
      proof -
        have Bet C B A'
          by (smt Bet-perm Col-def P2 P5 assms(2) between-exchange3 not-bet-and-out outer-transitivity-between2)
        then have A B C CongA A0 B A'
          using P1 P3 acute-distincts assms(1) cong-diff-4 l11-14 not-col-distincts by blast
        then have Acute A' B A0
          using acute-conga--acute acute-sym assms(1) by blast
        moreover have B A0 Perp A' A
        proof -
          have B ≠ A0
            using P1 P3 col-trivial-1 cong-reverse-identity by blast
          moreover have B A Perp A' A
            using Perp-perm assms(3) by blast
          moreover have Col B A A0
            using P1 bet-col not-col-permutation-4 by blast
          ultimately show ?thesis
            using perp-col by blast
        qed
        ultimately show ?thesis
          using Col-cases P1 acute-col-perp--out bet-col by blast
      qed
      then have False
        using P1 not-bet-and-out by blast
    }
    moreover then have ¬ B Out A' C0 by auto
    ultimately show ?thesis
      using l6-4-2 P4 by blast
  qed
  then show ?thesis
    by (metis P2 P3 acute-distincts assms(1) cong-diff-3 l6-2 not-col-distincts)
qed
```

lemma *conga-inangle-per2--inangle*:
assumes *Per A B C* **and**
T InAngle A B C **and**
P B A CongA P B C **and**
Per B P T **and**
Coplanar A B C P
shows *P InAngle A B C*
proof *cases*
assume *P = T*
then show *?thesis*
by (*simp add*: *assms*(*2*))
next
assume *P1*: *P ≠ T*
obtain *P′* **where** *P2*: *P′ InAngle A B C ∧ P′ B A CongA P′ B C*
using *CongA-def angle-bisector assms*(*3*) **by** *presburger*
have *P3*: *Acute P′ B A*
using *P2 acute-sym assms*(*1*) *conga-inangle-per--acute* **by** *blast*
have *P4*: ¬ *Col A B C*
using *assms*(*1*) *assms*(*3*) *conga-diff2 conga-diff56 l8-9* **by** *blast*
have *P5*: *Col B P P′*
proof −
have ¬ *B Out A C*
using *Col-cases P4 out-col* **by** *blast*
moreover have *Coplanar A B P P′*
proof −
have *T1*: ¬ *Col C A B*
using *Col-perm P4* **by** *blast*
moreover have *Coplanar C A B P*
using *assms*(*5*) *ncoplanar-perm-8* **by** *blast*
moreover have *Coplanar C A B P′*
using *P2 inangle--coplanar ncoplanar-perm-21* **by** *blast*
ultimately show *?thesis*
using *coplanar-trans-1* **by** *blast*
qed
moreover have *Coplanar B C P P′*
proof −
have *Coplanar A B C P*
by (*meson P2 bet--coplanar calculation*(*1*) *calculation*(*2*) *col-in-angle-out coplanar-perm-18 coplanar-trans-1 inangle--coplanar l11-21-a l6-6 l6-7 not-col-permutation-4 not-col-permutation-5*)
have *Coplanar A B C P′*
using *P2 inangle--coplanar ncoplanar-perm-18* **by** *blast*
then show *?thesis*
using *P4* ‹*Coplanar A B C P*› *coplanar-trans-1* **by** *blast*
qed
ultimately show *?thesis* **using** *conga2-cop2--col P2 assms*(*3*) **by** *blast*
qed
have *B Out P P′*
proof −
have *Acute T B P′*
using *P2 acute-sym assms*(*1*) *assms*(*2*) *conga-inangle2-per--acute* **by** *blast*
moreover have *B P′ Perp T P*
by (*metis P1 P5 acute-distincts assms*(*3*) *assms*(*4*) *calculation col-per-perp conga-distinct l8-2 not-col-permutation-4*)
ultimately show *?thesis*
using *Col-cases P5 acute-col-perp--out* **by** *blast*
qed
then show *?thesis*
using *Out-cases P2 in-angle-trans inangle-distincts out341--inangle* **by** *blast*
qed

lemma *perp-not-par*:
assumes *A B Perp X Y*
shows ¬ *A B Par X Y*
proof −
obtain *P* **where** *P1*: *P PerpAt A B X Y*
using *Perp-def assms* **by** *blast*
{

assume *P2*: *A B Par X Y*
{
assume *P3*: *A B ParStrict X Y*
then have *False*
proof −
have *Col P A B*
using *Col-perm P1 perp-in-col* **by** *blast*
moreover have *Col P X Y*
using *P1 col-permutation-2 perp-in-col* **by** *blast*
ultimately show *?thesis*
using *P3 par-not-col* **by** *blast*
qed
}
{
assume *P4*: $A \neq B \wedge X \neq Y \wedge Col\ A\ X\ Y \wedge Col\ B\ X\ Y$
then have *False*
proof *cases*
assume $A = Y$
thus *?thesis*
using *P4 assms not-col-permutation-1 perp-not-col* **by** *blast*
next
assume $A \neq Y$
thus *?thesis*
using *Col-perm P4 Perp-perm assms perp-not-col2* **by** *blast*
qed
}
then have *False*
using *Par-def P2* ‹*A B ParStrict X Y* ⟹ *False*› **by** *auto*
}
thus *?thesis* **by** *auto*
qed

lemma *cong-conga-perp*:
assumes *B P TS A C* **and**
Cong A B C B **and**
A B P CongA C B P
shows *A C Perp B P*
proof −
have *P1*: ¬ *Col A B P*
using *TS-def assms*(*1*) **by** *blast*
then have *P2*: $B \neq P$
using *col-trivial-2* **by** *blast*
have *P3*: $A \neq B$
using *assms*(*1*) *ts-distincts* **by** *blast*
have *P4*: $C \neq B$
using *assms*(*1*) *ts-distincts* **by** *auto*
have *P5*: $A \neq C$
using *assms*(*1*) *not-two-sides-id* **by** *auto*
show *?thesis*
proof *cases*
assume *P6*: *Bet A B C*
then have *Per P B A*
by (*meson Tarski-neutral-dimensionless.conga-comm Tarski-neutral-dimensionless-axioms assms*(*3*) *l11-18-2*)
then show *?thesis*
using *P2 P3 P5 Per-perm P6 bet-col per-perp perp-col* **by** *blast*
next
assume *P7*: ¬ *Bet A B C*
obtain *T* **where** *P7A*: *Col T B P* ∧ *Bet A T C*
using *TS-def assms*(*1*) **by** *auto*
then have *P8*: $B \neq T$
using *P7* **by** *blast*
then have *P9*: *T B A CongA T B C*
by (*meson Col-cases P7A Tarski-neutral-dimensionless.col-conga--conga Tarski-neutral-dimensionless.conga-comm Tarski-neutral-dimensionless-axioms assms*(*3*))
then have *P10*: *Cong T A T C*
using *assms*(*2*) *cong2-conga-cong cong-reflexivity not-cong-2143* **by** *blast*

then have *P11*: *T Midpoint A C*
using *P7A midpoint-def not-cong-2134* **by** *blast*
have *P12*: *Per B T A*
using *P11 Per-def assms*(*2*) *not-cong-2143* **by** *blast*
then show *?thesis*
proof −
have *A C Perp B T*
by (*metis P11 P12 P5 P8 col-per-perp midpoint-col midpoint-distinct-1*)
moreover have $B \neq T$
by (*simp add*: *P8*)
moreover have $T \neq A$
using *P1 P7A* **by** *blast*
moreover have $C \neq T$
using *P10 P5 cong-identity* **by** *blast*
moreover have $C \neq A$
using *P5* **by** *auto*
moreover have *Col T A C*
by (*meson P7A bet-col not-col-permutation-4*)
ultimately show *?thesis*
using *P2 P7A not-col-permutation-4 perp-col1* **by** *blast*
qed
qed
qed

lemma *perp-inter-exists*:
assumes *A B Perp C D*
shows $\exists$ *P*. *Col A B P* $\wedge$ *Col C D P*
proof −
obtain *P* **where** *P PerpAt A B C D*
using *Perp-def assms* **by** *auto*
then show *?thesis*
using *perp-in-col* **by** *blast*
qed

lemma *perp-inter-perp-in*:
assumes *A B Perp C D*
shows $\exists$ *P*. *Col A B P* $\wedge$ *Col C D P* $\wedge$ *P PerpAt A B C D*
by (*meson Perp-def Tarski-neutral-dimensionless.perp-in-col Tarski-neutral-dimensionless-axioms assms*)

end

context *Tarski-2D*

begin

lemma *l12-9-2D*:
assumes *A1 A2 Perp C1 C2* **and**
B1 B2 Perp C1 C2
shows *A1 A2 Par B1 B2*
using *l12-9 all-coplanar assms*(*1*) *assms*(*2*) **by** *auto*

end

context *Tarski-neutral-dimensionless*

begin

3.12 Tarski: Chapter 13

3.12.1 Introduction

lemma *per2-col-eq*:
assumes $A \neq P$ **and**
$A \neq P'$ **and**
Per A P B **and**
Per A P' B **and**
Col P A P'

shows $P = P'$
by (*metis assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *col-per2-cases l6-16-1 l8-2 not-col-permutation-3*)

lemma *per2-preserves-diff*:
assumes $PO \neq A'$ **and**
$PO \neq B'$ **and**
Col PO A' B' **and**
Per PO A' A **and**
Per PO B' B **and**
$A' \neq B'$
shows $A \neq B$
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *assms*(*6*) *not-col-permutation-4 per2-col-eq* **by** *blast*

lemma *per23-preserves-bet*:
assumes *Bet A B C* **and**
$A \neq B'$ **and** $A \neq C'$ **and**
Col A B' C' **and**
Per A B' B **and**
Per A C' C
shows *Bet A B' C'*
proof −
have *P1*: *Col A B C*
by (*simp add*: *assms*(*1*) *bet-col*)
show *?thesis*
proof *cases*
assume *P2*: $B = B'$
then have *Col A C' C*
using *P1 assms*(*2*) *assms*(*4*) *col-transitivity-1* **by** *blast*
then have *P4*: $A = C' \vee C = C'$
by (*simp add*: *assms*(*6*) *l8-9*)
{
assume $A = C'$
then have *Bet A B' C'*
using *assms*(*3*) **by** *auto*
}
{
assume $C = C'$
then have *Bet A B' C'*
using *P2 assms*(*1*) **by** *auto*
}
then show *?thesis*
using *P4 assms*(*3*) **by** *auto*
next
assume *T1*: $B \neq B'$
have *T2*: $A \neq C$
using *assms*(*3*) *assms*(*6*) *l8-8* **by** *auto*
have *T3*: $C \neq C'$
using *P1 T1 assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *col-trivial-3 colx l8-9 not-col-permutation-5* **by** *blast*
have *T3A*: *A B' Perp B' B*
using *T1 assms*(*2*) *assms*(*5*) *per-perp* **by** *auto*
have *T3B*: *A C' Perp C' C*
using *T3 assms*(*3*) *assms*(*6*) *per-perp* **by** *auto*
have *T4*: *B B' Par C C'*
proof −
have *Coplanar A B' B C*
using *P1 ncop--ncols* **by** *blast*
moreover have *Coplanar A B' B C'*
using *assms*(*4*) *ncop--ncols* **by** *blast*
moreover have *Coplanar A B' B' C*
using *ncop-distincts* **by** *blast*
moreover have *B B' Perp A B'*
using *Perp-perm* ‹*A B' Perp B' B*› **by** *blast*
moreover have *C C' Perp A B'*
using *Col-cases Perp-cases T3B assms*(*2*) *assms*(*4*) *perp-col1* **by** *blast*
ultimately show *?thesis*
using *l12-9 bet--coplanar between-trivial* **by** *auto*

qed
moreover have *Bet A B′ C′*
proof *cases*
assume *B = C*
then show *?thesis*
by (*metis T1 Tarski-neutral-dimensionless.per-col-eq Tarski-neutral-dimensionless-axioms assms*(*4*) *assms*(*5*) *calculation l6-16-1 l6-6 or-bet-out out-diff1 par-id*)
next
assume *T6*: *B ≠ C*
have *T7*: ¬ *Col A B′ B*
using *T1 assms*(*2*) *assms*(*5*) *l8-9* **by** *blast*
have *T8*: ¬ *Col A C′ C*
using *T3 assms*(*3*) *assms*(*6*) *l8-9* **by** *blast*
have *T9*: *B′ ≠ C′*
using *P1 T6 assms*(*2*) *assms*(*5*) *assms*(*6*) *col-per2--per col-permutation-1 l8-2 l8-8* **by** *blast*
have *T10*: *B B′ ParStrict C C′* ∨ (*B ≠ B′* ∧ *C ≠ C′* ∧ *Col B C C′* ∧ *Col B′ C C′*)
using *Par-def calculation* **by** *blast*
{
assume *T11*: *B B′ ParStrict C C′*
then have *T12*: *B B′ OS C′ C*
using *l12-6 one-side-symmetry* **by** *blast*
have *B B′ TS A C*
using *Col-cases T6 T7 assms*(*1*) *bet--ts* **by** *blast*
then have *Bet A B′ C′*
using *T12 assms*(*4*) *l9-5 l9-9 not-col-distincts or-bet-out* **by** *blast*
}
{
assume *B ≠ B′* ∧ *C ≠ C′* ∧ *Col B C C′* ∧ *Col B′ C C′*
then have *Bet A B′ C′*
using *Col-def T6 T8 assms*(*1*) *col-transitivity-2* **by** *blast*
}
then show *?thesis*
using *T10* ‹*B B′ ParStrict C C′* ⟹ *Bet A B′ C′*› **by** *blast*
qed
ultimately show *?thesis*
by (*smt P1 Par-def T1 T2 assms*(*4*) *col-transitivity-2 not-col-permutation-1 par-strict-not-col-2*)
qed
qed

lemma *per23-preserves-bet-inv*:
assumes *Bet A B′ C′* **and**
A ≠ B′ **and**
Col A B C **and**
Per A B′ B **and**
Per A C′ C
shows *Bet A B C*
proof *cases*
assume *T1*: *B = B′*
then have *Col A C′ C*
using *Col-def assms*(*1*) *assms*(*2*) *assms*(*3*) *col-transitivity-1* **by** *blast*
then have *T2*: *A = C′* ∨ *C = C′*
by (*simp add*: *assms*(*5*) *l8-9*)
{
assume *A = C′*
then have *Bet A B C*
using *assms*(*1*) *assms*(*2*) *between-identity* **by** *blast*
}
{
assume *C = C′*
then have *Bet A B C*
by (*simp add*: *T1 assms*(*1*))
}
then show *?thesis*
using *T2* ‹*A = C′* ⟹ *Bet A B C*› **by** *auto*
next
assume *P1*: *B ≠ B′*

then have *P2*: *A B′ Perp B′ B*
using *assms*(*2*) *assms*(*4*) *per-perp* **by** *auto*
have *Per A C′ C*
by (*simp add*: *assms*(*5*))
then have *P2*: *C′ PerpAt A C′ C′ C*
by (*metis* (*mono-tags*, *lifting*) *Col-cases P1 assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *bet-col bet-neq12--neq col-transitivity-1 l8-9 per-perp-in*)
then have *P3*: *A C′ Perp C′ C*
using *perp-in-perp* **by** *auto*
then have *C′* ≠ *C*
using ‹*A C′ Perp C′ C*› *perp-not-eq-2* **by** *auto*
have *C′ PerpAt C′ A C C′*
by (*simp add*: *Perp-in-perm P2*)
then have (*C′ A Perp C C′*) ∨ (*C′ C′ Perp C C′*)
using *Perp-def* **by** *blast*
have *A* ≠ *C′*
using *assms*(*1*) *assms*(*2*) *between-identity* **by** *blast*
{
assume *C′ A Perp C C′*
have *Col A B′ C′* **using** *assms*(*1*)
by (*simp add*: *Col-def*)
have *A B′ Perp C′ C*
using *Col-cases* ‹*A C′ Perp C′ C*› ‹*Col A B′ C′*› *assms*(*2*) *perp-col* **by** *blast*
have *P7*: *B′ B Par C′ C*
proof −
have *Coplanar A B′ B′ C′*
using *ncop-distincts* **by** *blast*
moreover have *Coplanar A B′ B′ C*
using *ncop-distincts* **by** *auto*
moreover have *Coplanar A B′ B C′*
using *Bet-perm assms*(*1*) *bet--coplanar ncoplanar-perm-20* **by** *blast*
moreover have *Coplanar A B′ B C*
using *assms*(*3*) *ncop--ncols* **by** *auto*
moreover have *B′ B Perp A B′*
by (*metis P1 Perp-perm assms*(*2*) *assms*(*4*) *per-perp*)
moreover have *C′ C Perp A B′*
using *Perp-cases* ‹*A B′ Perp C′ C*› **by** *auto*
ultimately show *?thesis* **using** *l12-9* **by** *blast*
qed
have *Bet A B C*
proof *cases*
assume *B* = *C*
then show *?thesis*
by (*simp add*: *between-trivial*)
next
assume *T1*: *B* ≠ *C*
have *T2*: *B′ B ParStrict C′ C* ∨ (*B′* ≠ *B* ∧ *C′* ≠ *C* ∧ *Col B′ C′ C* ∧ *Col B C′ C*)
using *P7 Par-def* **by** *auto*
{
assume *T3*: *B′ B ParStrict C′ C*
then have *B′* ≠ *C′*
using *not-par-strict-id* **by** *auto*
have ∃ *X*. *Col X B′ B* ∧ *Col X B′ C*
using *col-trivial-1* **by** *blast*
have *B′ B OS C′ C*
by (*simp add*: *T3 l12-6*)
have *B′ B TS A C′*
by (*metis Bet-cases T3 assms*(*1*) *assms*(*2*) *bet--ts l9-2 par-strict-not-col-1*)
then have *T8*: *B′ B TS C A*
using ‹*B′ B OS C′ C*› *l9-2 l9-8-2* **by** *blast*
then obtain *T* **where** *T9*: *Col T B′ B* ∧ *Bet C T A*
using *TS-def* **by** *auto*
have ¬ *Col A C B′*
using *T8 assms*(*3*) *not-col-permutation-2 not-col-permutation-3 ts--ncol* **by** *blast*
then have *T* = *B*
by (*metis Col-def Col-perm T9 assms*(*3*) *colx*)

then have *Bet A B C*
using *Bet-cases T9* by *auto*
}
{
assume $B' \neq B \wedge C' \neq C \wedge Col\ B'\ C'\ C \wedge Col\ B\ C'\ C$
then have *Col A B′ B*
by (*metis Col-perm T1 assms(3) l6-16-1*)
then have $A = B' \vee B = B'$
using *assms(4) l8-9* by *auto*
then have *Bet A B C*
by (*simp add: P1 assms(2)*)
}
then show *?thesis*
using *T2* ‹*B′ B ParStrict C′ C* ⟹ *Bet A B C*› by *auto*
qed
}
then show *?thesis*
by (*simp add: P3 perp-comm*)
qed

lemma *per13-preserves-bet*:
assumes *Bet A B C* **and**
$B \neq A'$ **and**
$B \neq C'$ **and**
Col A′ B C′ **and**
Per B A′ A **and**
Per B C′ C
shows *Bet A′ B C′*
by (*smt Col-cases Tarski-neutral-dimensionless.per23-preserves-bet-inv Tarski-neutral-dimensionless-axioms assms(1) assms(4) assms(5) assms(6) bet-col between-equality between-symmetry per-distinct third-point*)

lemma *per13-preserves-bet-inv*:
assumes *Bet A′ B C′* **and**
$B \neq A'$ **and**
$B \neq C'$ **and**
Col A B C **and**
Per B A′ A **and**
Per B C′ C
shows *Bet A B C*
proof −
have *P1*: *Col A′ B C′*
by (*simp add: Col-def assms(1)*)
show *?thesis*
proof *cases*
assume $A = A'$
then show *?thesis*
using *P1 assms(1) assms(3) assms(4) assms(6) col-transitivity-2 l8-9 not-bet-distincts* **by** *blast*
next
assume $A \neq A'$
show *?thesis*
by (*metis Col-cases P1 Tarski-neutral-dimensionless.per23-preserves-bet Tarski-neutral-dimensionless-axioms assms(1) assms(2) assms(3) assms(4) assms(5) assms(6) between-equality between-symmetry third-point*)
qed
qed

lemma *per3-preserves-bet1*:
assumes *Col PO A B* **and**
Bet A B C **and**
$PO \neq A'$ **and**
$PO \neq B'$ **and**
$PO \neq C'$ **and**
Per PO A′ A **and**
Per PO B′ B **and**
Per PO C′ C **and**
Col A′ B′ C′ **and**
Col PO A′ B′

shows *Bet* A' B' C'
proof *cases*
assume $A = B$
then show *?thesis*
using *assms*(*10*) *assms*(*3*) *assms*(*4*) *assms*(*6*) *assms*(*7*) *between-trivial2 per2-preserves-diff* **by** *blast*
next
assume *P1*: $A \neq B$
show *?thesis*
proof *cases*
assume *P2*: $A = A'$
show *?thesis*
proof *cases*
assume *P3*: $B = B'$
then have *Col PO C* C'
by (*metis* (*no-types*, *opaque-lifting*) *Col-def P1 P2 assms*(*1*) *assms*(*2*) *assms*(*9*) *col-transitivity-1*)
then have $C = C'$
using *assms*(*5*) *assms*(*8*) *l8-9 not-col-permutation-5* **by** *blast*
then show *?thesis*
using *P2 P3 assms*(*2*) **by** *blast*
next
assume *P4*: $B \neq B'$
show *?thesis*
proof *cases*
assume $A = B'$
then show *?thesis*
using *P2 between-trivial2* **by** *auto*
next
assume $A \neq B'$
have $A \neq C$
using *P1 assms*(*2*) *between-identity* **by** *blast*
have *P7*: ¬ *Col PO* B' *B*
using *P4 assms*(*4*) *assms*(*7*) *l8-9* **by** *blast*
show *?thesis*
using *P2 P7 assms*(*1*) *assms*(*10*) *assms*(*3*) *col-transitivity-1* **by** *blast*
qed
qed
next
assume *R1*: $A \neq A'$
show *?thesis*
proof *cases*
assume *R2*: $A' = B'$
then show *?thesis*
by (*simp add*: *between-trivial2*)
next
assume *R3*: $A' \neq B'$
show *?thesis*
proof *cases*
assume $B = C$
have $B' = C'$
by (*metis Tarski-neutral-dimensionless.per2-col-eq Tarski-neutral-dimensionless-axioms* ‹$A' \neq B'$› ‹$B = C$› *assms*(*10*) *assms*(*4*) *assms*(*5*) *assms*(*7*) *assms*(*8*) *assms*(*9*) *col-transitivity-2 not-col-permutation-2*)
then show *?thesis*
by (*simp add*: *between-trivial*)
next
assume *R4*: $B \neq C$
show *?thesis*
proof *cases*
assume $B = B'$
then show *?thesis*
by (*metis R1 assms*(*1*) *assms*(*10*) *assms*(*3*) *assms*(*4*) *assms*(*6*) *l6-16-1 l8-9 not-col-permutation-2*)
next
assume *R5*: $B \neq B'$
show *?thesis*
proof *cases*
assume $A' = B$
then show *?thesis*

using *R5 assms*(*10*) *assms*(*4*) *assms*(*7*) *col-permutation-5 l8-9* **by** *blast*
next
assume *R5A*: $A' \neq B$
have *R6*: $C \neq C'$
by (*metis P1 R1 R3 assms*(*1*) *assms*(*10*) *assms*(*2*) *assms*(*3*) *assms*(*5*) *assms*(*6*) *assms*(*9*) *bet-col col-permutation-1 col-trivial-2 l6-21 l8-9*)
have *R7*: *A A′ Perp PO A′*
by (*metis Perp-cases R1 assms*(*3*) *assms*(*6*) *per-perp*)
have *R8*: *C C′ Perp PO A′*
by (*smt Perp-cases R3 R6 assms*(*10*) *assms*(*3*) *assms*(*5*) *assms*(*8*) *assms*(*9*) *col2--eq col3 col-per-perp col-trivial-2 l8-2 per-perp*)
have *A A′ Par C C′*
proof −
have *Coplanar PO A′ A C*
using *P1 assms*(*1*) *assms*(*2*) *bet-col col-trivial-2 colx ncop--ncols* **by** *blast*
moreover have *Coplanar PO A′ A C′*
using *R3 assms*(*10*) *assms*(*9*) *col-trivial-2 colx ncop--ncols* **by** *blast*
moreover have *Coplanar PO A′ A′ C*
using *ncop-distincts* **by** *blast*
moreover have *Coplanar PO A′ A′ C′*
using *ncop-distincts* **by** *blast*
ultimately show *?thesis* **using** *l12-9 R7 R8* **by** *blast*
qed
have *S1*: *B B′ Perp PO A′*
by (*metis Col-cases Per-cases Perp-perm R5 assms*(*10*) *assms*(*3*) *assms*(*4*) *assms*(*7*) *col-per-perp*)
have *A A′ Par B B′*
proof −
have *Coplanar PO A′ A B*
using *assms*(*1*) *ncop--ncols* **by** *auto*
moreover have *Coplanar PO A′ A B′*
using *assms*(*10*) *ncop--ncols* **by** *auto*
moreover have *Coplanar PO A′ A′ B*
using *ncop-distincts* **by** *auto*
moreover have *Coplanar PO A′ A′ B′*
using *ncop-distincts* **by** *auto*
moreover have *A A′ Perp PO A′*
by (*simp add*: *R7*)
moreover have *B B′ Perp PO A′*
by (*simp add*: *S1*)
ultimately show *?thesis*
using *l12-9* **by** *blast*
qed
{
assume *A A′ ParStrict B B′*
then have *A A′ OS B B′*
by (*simp add*: *l12-6*)
have *B B′ TS A C*
using *R4* ‹*A A′ ParStrict B B′*› *assms*(*2*) *bet--ts par-strict-not-col-3* **by** *auto*
have *B B′ OS A A′*
using ‹*A A′ ParStrict B B′*› *pars--os3412* **by** *auto*
have *B B′ TS A′ C*
using ‹*B B′ OS A A′*› ‹*B B′ TS A C*› *l9-8-2* **by** *blast*
have *Bet A′ B′ C′*
proof *cases*
assume $C = C'$
then show *?thesis*
using *R6* **by** *auto*
next
assume $C \neq C'$
have *C C′ Perp PO A′*
by (*simp add*: *R8*)
have *Q2*: *B B′ Par C C′*
proof −
have *Coplanar PO A′ B C*
by (*metis P1 assms*(*1*) *assms*(*2*) *bet-col col-transitivity-1 colx ncop--ncols not-col-permutation-5*)
moreover have *Coplanar PO A′ B C′*

using *R3 assms*(*10*) *assms*(*9*) *col-trivial-2 colx ncop--ncols* **by** *blast*
moreover have *Coplanar PO A′ B′ C*
by (*simp add*: *assms*(*10*) *col--coplanar*)
moreover have *Coplanar PO A′ B′ C′*
using *assms*(*10*) *col--coplanar* **by** *auto*
moreover have *B B′ Perp PO A′*
by (*simp add*: *S1*)
moreover have *C C′ Perp PO A′*
by (*simp add*: *R8*)
ultimately show *?thesis*
using *l12-9* **by** *auto*
qed
then have *Q3*: (*B B′ ParStrict C C′*) $\vee$ ($B \neq B' \wedge C \neq C'$ $\wedge$ *Col B C C′* $\wedge$ *Col B′ C C′*)
by (*simp add*: *Par-def*)
{
assume *B B′ ParStrict C C′*
then have *B B′ OS C C′*
using *l12-6* **by** *auto*
then have *B B′ TS C′ A′*
using ‹*B B′ TS A′ C*› *l9-2 l9-8-2* **by** *blast*
then obtain *T* **where** *Q4*: *Col T B B′* $\wedge$ *Bet C′ T A′*
using *TS-def* **by** *blast*
have $T = B'$
proof −
have ¬ *Col B B′ A′*
using ‹*B B′ OS A A′*› *col124--nos* **by** *auto*
moreover have $A' \neq C'$
using ‹*B B′ TS C′ A′*› *not-two-sides-id* **by** *auto*
moreover have *Col B B′ T*
using *Col-cases Q4* **by** *auto*
moreover have *Col B B′ B′*
using *not-col-distincts* **by** *blast*
moreover have *Col A′ C′ T*
by (*simp add*: *Col-def Q4*)
ultimately show *?thesis*
by (*meson assms*(*9*) *col-permutation-5 l6-21*)
qed
then have *Bet A′ B′ C′*
using *Q4 between-symmetry* **by** *blast*
}
{
assume $B \neq B' \wedge C \neq C'$ $\wedge$ *Col B C C′* $\wedge$ *Col B′ C C′*
then have *Bet A′ B′ C′*
using *TS-def* ‹*B B′ TS A C*› *l6-16-1 not-col-permutation-2* **by** *blast*
}
then show *?thesis*
using *Q3* ‹*B B′ ParStrict C C′* $\Longrightarrow$ *Bet A′ B′ C′*› **by** *blast*
qed
}
{
assume *R8*: $A \neq A' \wedge B \neq B'$ $\wedge$ *Col A B B′* $\wedge$ *Col A′ B B′*
have *A′ A Perp PO A′*
by (*simp add*: *R7 perp-left-comm*)
have ¬ *Col A′ A PO*
using *Col-cases R8 assms*(*3*) *assms*(*6*) *l8-9* **by** *blast*
then have *Bet A′ B′ C′*
using *Col-perm P1 R8 assms*(*1*) *l6-16-1* **by** *blast*
}
then show *?thesis*
using *Par-def* ‹*A A′ Par B B′*› ‹*A A′ ParStrict B B′* $\Longrightarrow$ *Bet A′ B′ C′*› **by** *auto*
qed
qed
qed
qed
qed
qed

lemma *per3-preserves-bet2-aux*:
assumes *Col PO A C* **and**
$A \neq C'$ **and**
Bet A B' C' **and**
$PO \neq A$ **and**
$PO \neq B'$ **and**
$PO \neq C'$ **and**
Per PO B' B **and**
Per PO C' C **and**
Col A B C **and**
Col PO A C'
shows *Bet A B C*
proof *cases*
assume $A = B$
then show *?thesis*
by (*simp add*: *between-trivial2*)
next
assume *P1*: $A \neq B$
show *?thesis*
proof *cases*
assume $B = C$
then show *?thesis*
by (*simp add*: *between-trivial*)
next
assume *P2*: $B \neq C$
have *P3*: *Col PO A B'*
by (*metis Col-def assms*(*10*) *assms*(*2*) *assms*(*3*) *l6-16-1*)
then have *P4*: *Col PO B' C'*
using *assms*(*10*) *assms*(*4*) *col-transitivity-1* **by** *blast*
show *?thesis*
proof *cases*
assume $B = B'$
thus *?thesis*
by (*metis Tarski-neutral-dimensionless.per-col-eq Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*10*) *assms*(*3*) *assms*(*4*) *assms*(*6*) *assms*(*8*) *col-transitivity-1*)
next
assume *P5*: $B \neq B'$
have *P6*: $C = C'$
using *assms*(*1*) *assms*(*10*) *assms*(*4*) *assms*(*6*) *assms*(*8*) *col-transitivity-1 l8-9* **by** *blast*
then have *False*
by (*metis P3 P5 P6 Tarski-neutral-dimensionless.per-col-eq Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*2*) *assms*(*4*) *assms*(*5*) *assms*(*7*) *assms*(*9*) *col-transitivity-1 l6-16-1 not-col-permutation-4*)
then show *?thesis* **by** *blast*
qed
qed
qed

lemma *per3-preserves-bet2*:
assumes *Col PO A C* **and**
$A' \neq C'$ **and**
Bet A' B' C' **and**
$PO \neq A'$ **and**
$PO \neq B'$ **and**
$PO \neq C'$ **and**
Per PO A' A **and**
Per PO B' B **and**
Per PO C' C **and**
Col A B C **and**
Col PO A' C'
shows *Bet A B C*
proof *cases*
assume $A = A'$
then show *?thesis*
using *assms*(*1*) *assms*(*10*) *assms*(*11*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *assms*(*6*) *assms*(*8*) *assms*(*9*) *per3-preserves-bet2-aux* **by** *blast*

next
  assume *P1*: $A \neq A'$
  show *?thesis*
  proof *cases*
    assume $C = C'$
    thus *?thesis*
      by (*metis P1 assms*(*1*) *assms*(*11*) *assms*(*4*) *assms*(*6*) *assms*(*7*) *col-trivial-3 l6-21 l8-9 not-col-permutation-2*)
  next
    assume *P2*: $C \neq C'$
    then have *P3*: *PO* A' *Perp C* C'
      by (*metis assms*(*11*) *assms*(*4*) *assms*(*6*) *assms*(*9*) *col-per-perp l8-2 not-col-permutation-1*)
    have *P4*: *PO* A' *Perp A* A'
      using *P1 assms*(*4*) *assms*(*7*) *per-perp perp-right-comm* **by** *auto*
    have *A* A' *Par C* C'
    proof −
      have *Coplanar PO* A' *A C*
        using *assms*(*1*) *ncop--ncols* **by** *blast*
      moreover have *Coplanar PO* A' *A* C'
        by (*meson assms*(*11*) *ncop--ncols*)
      moreover have *Coplanar PO* A' A' *C*
        using *ncop-distincts* **by** *blast*
      moreover have *Coplanar PO* A' A' C'
        using *ncop-distincts* **by** *blast*
      moreover have *A* A' *Perp PO* A'
        using *P4 Perp-cases* **by** *blast*
      moreover have *C* C' *Perp PO* A'
        using *P3 Perp-cases* **by** *auto*
      ultimately show *?thesis*
        using *l12-9* **by** *blast*
    qed
    {
      assume *P5*: *A* A' *ParStrict C* C'
      then have *P6*: *A* A' *OS C* C'
        by (*simp add*: *l12-6*)
      have *P7*: *C* C' *OS A* A'
        by (*simp add*: *P5 pars--os3412*)

      have *Bet A B C*
      proof *cases*
        assume *P8*: $B = B'$
        then have A' *A OS B* C'
          by (*metis P6 assms*(*10*) *assms*(*3*) *bet-out col123--nos col124--nos invert-one-side out-one-side*)
        then have *A* A' *OS B* C'
          by (*simp add*: *invert-one-side*)
        then have *A* A' *OS B C*
          using *P6 one-side-symmetry one-side-transitivity* **by** *blast*
        then have *P12*: *A Out B C*
          using *assms*(*10*) *col-one-side-out* **by** *blast*
        have C' *C OS B* A'
          by (*metis Col-perm P5 P7 P8 assms*(*10*) *assms*(*3*) *bet-out-1 col123--nos out-one-side par-strict-not-col-2*)
        then have *C* C' *OS B A*
          by (*meson P7 invert-one-side one-side-symmetry one-side-transitivity*)
        then have *C* C' *OS A B*
          using *one-side-symmetry* **by** *blast*
        then have *C Out A B*
          using *assms*(*10*) *col-one-side-out col-permutation-2* **by** *blast*
        then show *?thesis*
          by (*simp add*: *P12 out2--bet*)
      next
        assume *T1*: $B \neq B'$
        have *T2*: *PO* A' *Perp B* B'
        proof −
          have *Per PO* B' *B*
            by (*simp add*: *assms*(*8*))
          then have B' *PerpAt PO* B' B' *B*
            using *T1 assms*(*5*) *per-perp-in* **by** *auto*

```
      then have B′ PerpAt B′ PO B B′
        by (simp add: perp-in-comm)
      then have T4: B′ PO Perp B B′ ∨ B′ B′ Perp B B′
        using Perp-def by auto
      {
        assume T5: B′ PO Perp B B′
        have Col A′ B′ C′
          by (simp add: assms(3) bet-col)
        then have Col PO B′ A′
          using assms(11) assms(2) col2--eq col-permutation-4 col-permutation-5 by blast
        then have PO A′ Perp B B′
          by (metis T5 assms(4) col-trivial-3 perp-col2 perp-comm)
      }
      {
        assume B′ B′ Perp B B′
        then have PO A′ Perp B B′
          using perp-distinct by auto
      }
      then show ?thesis
        using T4 ‹B′ PO Perp B B′ ⟹ PO A′ Perp B B′› by linarith
    qed
    have T6: B B′ Par A A′
    proof −
      have Coplanar PO A′ B A
        by (metis Col-cases P7 assms(1) assms(10) col-transitivity-2 ncop--ncols os-distincts)
      moreover have Coplanar PO A′ B A′
        using ncop-distincts by blast
      moreover have Coplanar PO A′ B′ A
      proof −
        have (Bet PO A′ C′ ∨ Bet PO C′ A′) ∨ Bet C′ PO A′
          by (meson assms(11) third-point)
        then show ?thesis
          by (meson Bet-perm assms(3) bet--coplanar between-exchange2 l5-3 ncoplanar-perm-8)
      qed
      moreover have Coplanar PO A′ B′ A′
        using ncop-distincts by auto
      moreover have B B′ Perp PO A′
        using Perp-cases T2 by blast
      moreover have A A′ Perp PO A′
        using P4 Perp-cases by blast
      ultimately show ?thesis
        using l12-9 by blast
    qed
    {
      assume B B′ ParStrict A A′
      then have B B′ OS A A′
        by (simp add: l12-6)
      have B B′ Par C C′
      proof −
        have Coplanar PO A′ B C
          by (metis Col-cases P7 assms(1) assms(10) col2--eq ncop--ncols os-distincts)
        moreover have Coplanar PO A′ B C′
          using assms(11) ncop--ncols by auto
        moreover have Coplanar PO A′ B′ C
            by (metis Out-def assms(11) assms(2) assms(3) col-trivial-2 l6-16-1 ncop--ncols not-col-permutation-1
out-col)
        moreover have Coplanar PO A′ B′ C′
          using assms(11) ncop--ncols by blast
        moreover have B B′ Perp PO A′
          using Perp-cases T2 by blast
        moreover have C C′ Perp PO A′
          using P3 Perp-cases by auto
        ultimately show ?thesis
          using l12-9 by blast
      qed
      {
```

assume *T9*: *B B′ ParStrict C C′*
then have *T10*: *B B′ OS C C′*
by (*simp add*: *l12-6*)
have *T11*: *B B′ TS A′ C′*
by (*metis Col-cases T10* ‹*B B′ ParStrict A A′*› *assms*(*3*) *bet--ts invert-two-sides os-distincts par-strict-not-col-4*)
have *T12*: *B B′ TS A C′*
using ‹*B B′ OS A A′*› ‹*B B′ TS A′ C′*› *l9-8-2 one-side-symmetry* by *blast*
then have *T12A*: *B B′ TS C A*
using *T10 l9-2 l9-8-2 one-side-symmetry* by *blast*
then obtain *T* where *T13*: *Col T B B′* ∧ *Bet C T A*
using *TS-def* by *auto*
then have *B* = *T*
by (*metis Col-perm TS-def T12A assms*(*10*) *bet-col1 col-transitivity-2 col-two-sides-bet*)
then have *Bet A B C*
using *Bet-perm T13* by *blast*
}
{
assume *B* ≠ *B′* ∧ *C* ≠ *C′* ∧ *Col B C C′* ∧ *Col B′ C C′*
then have *Bet A B C*
by (*metis Col-cases P5 assms*(*10*) *col3 col-trivial-2 not-bet-distincts par-strict-not-col-3*)
}
then have *Bet A B C*
using *Par-def* ‹*B B′ Par C C′*› ‹*B B′ ParStrict C C′* ⟹ *Bet A B C*› by *auto*
}
{
assume *B* ≠ *B′* ∧ *A* ≠ *A′* ∧ *Col B A A′* ∧ *Col B′ A A′*
then have *Bet A B C*
by (*smt P6 assms*(*10*) *col123--nos l6-16-1 not-bet-distincts not-col-permutation-1*)
}
then show *?thesis*
using *Par-def T6* ‹*B B′ ParStrict A A′* ⟹ *Bet A B C*› by *auto*
qed
}
{
assume *A* ≠ *A′* ∧ *C* ≠ *C′* ∧ *Col A C C′* ∧ *Col A′ C C′*
then have *Bet A B C*
by (*metis Col-perm P3 Par-def assms*(*11*) *assms*(*2*) *assms*(*4*) *col-transitivity-1 perp-not-par*)
}
thus *?thesis*
using *Par-def* ‹*A A′ Par C C′*› ‹*A A′ ParStrict C C′* ⟹ *Bet A B C*› by *auto*
qed
qed

lemma *symmetry-preserves-per*:
assumes *Per B P A* **and**
B Midpoint A A′ **and**
B Midpoint P P′
shows *Per B P′ A′*
proof −
obtain *C* **where** *P1*: *P Midpoint A C*
using *symmetric-point-construction* **by** *blast*
obtain *C′* **where** *P2*: *B Midpoint C C′*
using *symmetric-point-construction* **by** *blast*
have *P3*: *P′ Midpoint A′ C′*
using *P1 P2 assms*(*2*) *assms*(*3*) *symmetry-preserves-midpoint* **by** *blast*
have *Cong B A′ B C′*
by (*meson P1 P2 assms*(*1*) *assms*(*2*) *l7-16 l7-3-2 per-double-cong*)
then show *?thesis*
using *P3 Per-def* **by** *blast*
qed

lemma *l13-1-aux*:
assumes ¬ *Col A B C* **and**
P Midpoint B C **and**
Q Midpoint A C **and**
R Midpoint A B

shows
$\exists$ *X Y*. (*R PerpAt X Y A B* $\wedge$ *X Y Perp P Q* $\wedge$ *Coplanar A B C X* $\wedge$ *Coplanar A B C Y*)
proof −
have *P1*: $Q \neq C$
using *assms*(*1*) *assms*(*3*) *midpoint-not-midpoint not-col-distincts* **by** *blast*
have *P2*: $P \neq C$
using *assms*(*1*) *assms*(*2*) *is-midpoint-id-2 not-col-distincts* **by** *blast*
then have $Q \neq R$
using *assms*(*2*) *assms*(*3*) *assms*(*4*) *l7-3 symmetric-point-uniqueness* **by** *blast*
have $R \neq B$
using *assms*(*1*) *assms*(*4*) *midpoint-not-midpoint not-col-distincts* **by** *blast*
{
assume *V1*: *Col P Q C*
have *V2*: *Col B P C*
by (*simp add*: *assms*(*2*) *bet-col midpoint-bet*)
have *V3*: *Col A Q C*
by (*simp add*: *assms*(*3*) *bet-col midpoint-bet*)
have *Col A R B*
using *assms*(*4*) *midpoint-col not-col-permutation-4* **by** *blast*
then have *Col A B C* **using** *V1 V2 V3*
by (*metis P1 P2 col2--eq col-permutation-5*)
then have *False*
using *assms*(*1*) **by** *auto*
}
then have *P2A*: ¬ *Col P Q C* **by** *auto*
then obtain *C′* **where** *P3*: *Col P Q C′* $\wedge$ *P Q Perp C C′*
using *l8-18-existence* **by** *blast*
obtain *A′* **where** *P4*: *Q Midpoint C′ A′*
using *symmetric-point-construction* **by** *auto*
obtain *B′* **where** *P5*: *P Midpoint C′ B′*
using *symmetric-point-construction* **by** *auto*
have *P6*: *Cong C C′ B B′*
using *Mid-cases P5 assms*(*2*) *l7-13* **by** *blast*
have *P7*: *Cong C C′ A A′*
using *P4 assms*(*3*) *l7-13 l7-2* **by** *blast*
have *P8*: *Per P B′ B*
proof *cases*
assume $P = C'$
then show *?thesis*
using *P5 Per-cases is-midpoint-id l8-5* **by** *blast*
next
assume $P \neq C'$
then have *P C′ Perp C C′*
using *P3 perp-col* **by** *blast*
then have *Per P C′ C*
using *Perp-perm perp-per-2* **by** *blast*
then show *?thesis*
using *symmetry-preserves-per Mid-perm P5 assms*(*2*) **by** *blast*
qed
have *P8A*: *Per Q A′ A*
proof *cases*
assume $Q = C'$
then show *?thesis*
using *P4 Per-cases is-midpoint-id l8-5* **by** *blast*
next
assume $Q \neq C'$
then have *C′ Q Perp C C′*
using *P3 col-trivial-2 perp-col2* **by** *auto*
then have *Per Q C′ C*
by (*simp add*: *perp-per-1*)
then show *?thesis*
by (*meson Mid-cases P4 assms*(*3*) *l7-3-2 midpoint-preserves-per*)
qed
have *P9*: *Col A′ C′ Q*
using *P4 midpoint-col not-col-permutation-3* **by** *blast*
have *P10*: *Col B′ C′ P*

using *P5 midpoint-col not-col-permutation-3* **by** *blast*
have *P11*: $P \neq Q$
using *P2A col-trivial-1* **by** *auto*
then have *P12*: $A' \neq B'$
using *P4 P5 l7-17* **by** *blast*
have *P13*: *Col* A' B' *P*
by (*metis P10 P3 P4 P5 P9 col2--eq col-permutation-5 midpoint-distinct-1 not-col-distincts*)
have *P14*: *Col* A' B' *Q*
by (*smt P10 P3 P4 P5 P9 col3 col-permutation-1 midpoint-distinct-1 not-col-distincts*)
have *P15*: *Col* A' B' C'
using *P11 P13 P14 P3 colx* **by** *blast*
have *P16*: $C \neq C'$
using *P2A P3* **by** *blast*
then have *P17*: $A \neq A'$
using *P7 cong-diff* **by** *blast*
have *P18*: $B \neq B'$
using *P16 P6 cong-diff* **by** *blast*
have *P19*: *Per P* A' *A*
proof *cases*
assume *P20*: $A' = Q$
then have A' *P Perp C* A'
by (*metis P3 P4 Perp-cases midpoint-not-midpoint*)
then have *Per P* A' *C*
by (*simp add*: *perp-per-1*)
then show *?thesis*
using *P20 assms*(*3*) *l7-2 l8-4* **by** *blast*
next
assume $A' \neq Q$
then show *?thesis*
by (*meson P12 P13 P14 P8A col-transitivity-1 l8-2 per-col*)
qed
have *Per Q* B' *B*
proof *cases*
assume *P21*: $P = B'$
then have *P22*: $C' = B'$
using *P5 is-midpoint-id-2* **by** *auto*
then have *Per Q* B' *C*
using *P3 P21 perp-per-1* **by** *auto*
thus *?thesis*
by (*metis Col-perm P16 P21 P22 assms*(*2*) *midpoint-col per-col*)
next
assume *P23*: $P \neq B'$
have *Col* B' *P Q*
using *P12 P13 P14 col-transitivity-2* **by** *blast*
then have *Per B* B' *Q*
using *P8 P23 l8-2 l8-3* **by** *blast*
thus *?thesis*
using *Per-perm* **by** *blast*
qed
then have *P24*: *Per* A' B' *B*
using *P11 P13 P14 P8 l8-3 not-col-permutation-2* **by** *blast*
have *P25*: *Per A* A' B'
using *P11 P13 P14 P19 P8A l8-2 l8-3 not-col-permutation-5* **by** *blast*
then have *Per* B' A' *A*
using *Per-perm* **by** *blast*
then have $\neg$ *Col* B' A' *A*
using *P12 P17 P25 per-not-col* **by** *auto*
then have *P26*: $\neg$ *Col* A' B' *A*
using *Col-cases* **by** *auto*
have $\neg$ *Col* A' B' *B*
using *P12 P18 P24 l8-9* **by** *auto*
obtain *X* **where** *P28*: *X Midpoint* A' B'
using *midpoint-existence* **by** *blast*
then have *P28A*: *Col* A' B' *X*
using *midpoint-col not-col-permutation-2* **by** *blast*
then have $\exists$ *Q*. A' B' *Perp Q X* $\wedge$ A' B' *OS A Q*

```
  by (simp add: P26 l10-15)
then obtain y where P29: A′ B′ Perp y X ∧ A′ B′ OS A y by blast
then obtain B′′ where P30: (X y Perp A B′′ ∨ A = B′′) ∧ (∃ M. (Col X y M ∧ M Midpoint A B′′))
  using ex-sym by blast
then have P31: B′′ A ReflectL X y
  using P30 ReflectL-def by blast
have P32: X ≠ y
  using P29 P28A col124--nos by blast
then have X ≠ y ∧ B′′ A ReflectL X y ∨ X = y ∧ X Midpoint A B′′
  using P31 by auto
then have P33: B′′ A Reflect X y
  by (simp add: Reflect-def)
have P33A: X ≠ y ∧ A′ B′ ReflectL X y
  using P28 P29 Perp-cases ReflectL-def P32 col-trivial-3 l10-4-spec by blast
then have P34: A′ B′ Reflect X y
  using Reflect-def by blast
have P34A: A B′′ Reflect X y
  using P33 l10-4 by blast
then have P35: Cong B′′ B′ A A′
  using P34 l10-10 by auto
have Per A′ B′ B′′
proof −
  have R1: X ≠ y ∧ A B′′ ReflectL X y ∨ X = y ∧ X Midpoint B′′ A
    by (simp add: P31 P32 l10-4-spec)
  have R2: X ≠ y ∧ A′ B′ ReflectL X y ∨ X = y ∧ X Midpoint B′ A′
    using P33A by linarith
  {
    assume X ≠ y ∧ A B′′ ReflectL X y ∧ X ≠ y ∧ A′ B′ ReflectL X y
    then have Per A′ B′ B′′
      using ‹Per B′ A′ A› image-spec-preserves-per l10-4-spec by blast
  }
  {
    assume X ≠ y ∧ A B′′ ReflectL X y ∧ X = y ∧ X Midpoint B′ A′
    then have Per A′ B′ B′′ by blast
  }
  {
    assume X = y ∧ X Midpoint B′′ A ∧ X ≠ y ∧ A′ B′ ReflectL X y
    then have Per A′ B′ B′′ by blast
  }
  {
    assume X = y ∧ X Midpoint B′′ A ∧ X = y ∧ X Midpoint B′ A′
    then have Per A′ B′ B′′
      using P32 by blast
  }
  then show ?thesis using R1 R2
    using ‹X ≠ y ∧ A B′′ ReflectL X y ∧ X ≠ y ∧ A′ B′ ReflectL X y ⟹ Per A′ B′ B′′› by auto
qed
have A′ B′ OS A B′′
proof −
  {
    assume S1: X y Perp A B′′
    have Coplanar A y A′ X
      by (metis P28A P29 col-one-side coplanar-perm-16 ncop-distincts os--coplanar)
    have Coplanar A y B′ X
      by (smt P12 P28A P29 col2-cop--cop col-transitivity-1 ncoplanar-perm-22 not-col-permutation-5 os--coplanar)
    have S2: ¬ Col A X y
      using Col-perm P34A S1 local.image-id perp-distinct by blast

    have A′ B′ Par A B′′
    proof −
      have Coplanar X y A′ A
        using ‹Coplanar A y A′ X› ncoplanar-perm-21 by blast
      moreover have Coplanar X y A′ B′′
      proof −
        have Coplanar A X y A′
          using ‹Coplanar X y A′ A› ncoplanar-perm-9 by blast
```

moreover have *Coplanar A X y B''*
using *Coplanar-def S1 perp-inter-exists* **by** *blast*
ultimately show *?thesis*
using *S2 coplanar-trans-1* **by** *auto*
qed
moreover have *Coplanar X y B' A*
proof −
have ¬ *Col A X y*
by (*simp add*: *S2*)
moreover have *Coplanar A X y B'*
using ‹*Coplanar A y B' X*› *ncoplanar-perm-3* **by** *blast*
moreover have *Coplanar A X y B''*
using *Coplanar-def S1 perp-inter-exists* **by** *blast*
ultimately show *?thesis*
using *ncoplanar-perm-18* **by** *blast*
qed
moreover have *Coplanar X y B' B''*
proof −
have ¬ *Col A X y*
by (*simp add*: *S2*)
moreover have *Coplanar A X y B'*
using ‹*Coplanar X y B' A*› *ncoplanar-perm-9* **by** *blast*
moreover have *Coplanar A X y B''*
using *Coplanar-def S1 perp-inter-exists* **by** *blast*
ultimately show *?thesis*
using *coplanar-trans-1* **by** *blast*
qed
ultimately show *?thesis* **using** *l12-9*
using *P29 Perp-cases S1* **by** *blast*
qed
have *A' B' OS A B''*
proof −
{
assume *A' B' ParStrict A B''*
have *A' B' OS A B''* **using** *l12-6*
using ‹*A' B' ParStrict A B''*› **by** *blast*
}
{
assume *A'* ≠ *B'* ∧ *A* ≠ *B''* ∧ *Col A' A B''* ∧ *Col B' A B''*
have *A' B' OS A B''*
using *P26* ‹*A' B' Par A B''*› ‹*A' B' ParStrict A B''* ⟹ *A' B' OS A B''*› *col-trivial-3 par-not-col-strict* **by** *blast*
}
then show *?thesis*
using *Par-def* ‹*A' B' Par A B''*› ‹*A' B' ParStrict A B''* ⟹ *A' B' OS A B''*› **by** *auto*
qed
}
{
assume *A* = *B''*
then have *A' B' OS A B''*
using *P12 P25* ‹*Per A' B' B''*› *l8-2 l8-7* **by** *blast*
}
then show *?thesis*
using *P30* ‹*X y Perp A B''* ⟹ *A' B' OS A B''*› **by** *blast*
qed
have *A' B' OS A B*
proof −
have *A' B' TS A C*
proof −
have ¬ *Col A A' B'*
using *Col-perm* ‹¬ *Col B' A' A*› **by** *blast*
moreover have ¬ *Col C A' B'*
by (*metis P13 P14 P2A* ‹¬ *Col B' A' A*› *col3 not-col-distincts not-col-permutation-3 not-col-permutation-4*)
moreover have ∃ *T*. *Col T A' B'* ∧ *Bet A T C*
using *P14 assms*(*3*) *midpoint-bet not-col-permutation-1* **by** *blast*
ultimately show *?thesis*

```
      by (simp add: TS-def)
  qed
  moreover have A′ B′ TS B C
    by (metis Col-cases P13 TS-def ‹¬ Col A′ B′ B› assms(2) calculation midpoint-bet)
  ultimately show ?thesis
    using OS-def by blast
qed
have Col B B′′ B′
proof −
  have Coplanar A′ B B′′ B′
  proof −
    have Coplanar A′ B′ B B′′
    proof −
      have ¬ Col A A′ B′
        using Col-perm ‹¬ Col B′ A′ A› by blast
      moreover have Coplanar A A′ B′ B
        using ‹A′ B′ OS A B› ncoplanar-perm-8 os--coplanar by blast
      moreover have Coplanar A A′ B′ B′′
        using ‹A′ B′ OS A B′′› ncoplanar-perm-8 os--coplanar by blast
      ultimately show ?thesis
        using coplanar-trans-1 by blast
    qed
    then show ?thesis
      using ncoplanar-perm-4 by blast
  qed
  moreover have A′ ≠ B′
    by (simp add: P12)
  moreover have Per B B′ A′
    by (simp add: P24 l8-2)
  moreover have Per B′′ B′ A′
    using Per-cases ‹Per A′ B′ B′′› by auto
  ultimately show ?thesis
    using cop-per2--col by blast
qed
have Cong B B′ A A′
  using P6 P7 cong-inner-transitivity by blast
have B = B′′ ∨ B′ Midpoint B B′′
proof −
  have Col B B′ B′′
    using ‹Col B B′′ B′› not-col-permutation-5 by blast
  moreover have Cong B′ B B′ B′′
    by (metis Cong-perm P35 P6 P7 cong-inner-transitivity)
  ultimately show ?thesis
    using l7-20 by simp
qed
{
  assume B = B′′
  then obtain M where S1: Col X y M ∧ M Midpoint A B
    using P30 by blast
  then have R = M
    using assms(4) l7-17 by auto
  have A ≠ B
    using assms(1) col-trivial-1 by auto
  have Col R A B
    by (simp add: assms(4) midpoint-col)
  have X ≠ R
    using Midpoint-def P28 ‹A′ B′ OS A B′′› ‹B = B′′› assms(4) midpoint-col one-side-chara by auto
  then have ∃ X Y. (R PerpAt X Y A B ∧ X Y Perp P Q ∧ Coplanar A B C X ∧ Coplanar A B C Y)
  proof −
    have R PerpAt R X A B
    proof −
      have R X Perp A B
        using P30 S1 ‹A ≠ B› ‹B = B′′› ‹R = M› ‹X ≠ R› perp-col perp-left-comm by blast
      then show ?thesis
        using ‹Col R A B› l8-14-2-1b-bis not-col-distincts by blast
    qed
```

```
      moreover have R X Perp P Q
      proof –
        have X R Perp P Q
        proof –
          have X y Perp P Q
          proof –
            have P Q Perp X y
              using P11 P13 P14 P29 P33A col-trivial-2 col-trivial-3 perp-col4 by blast
            then show ?thesis
              using Perp-perm by blast
          qed
          moreover have Col X y R
            by (simp add: S1 ‹R = M›)
          ultimately show ?thesis
            using ‹X ≠ R› perp-col by blast
        qed
        then show ?thesis
          using Perp-perm by blast
      qed
      moreover have Coplanar A B C R
        using ‹Col R A B› ncop--ncols not-col-permutation-2 by blast
      moreover have Coplanar A B C X
      proof –
        have Col P Q X
          using P12 P13 P14 P28A col3 by blast
        moreover  have ¬ Col P Q C
          by (simp add: P2A)
        moreover have Coplanar P Q C A
          using assms(3) coplanar-perm-19 midpoint--coplanar by blast
        moreover have Coplanar P Q C B
          using assms(2) midpoint-col ncop--ncols not-col-permutation-5 by blast
        moreover have Coplanar P Q C C
          using ncop-distincts by auto
        moreover have Coplanar P Q C X
          using calculation(1) ncop--ncols by blast
        ultimately show ?thesis
          using coplanar-pseudo-trans by blast
      qed
      ultimately show ?thesis by blast
    qed
  }
  {
    assume B′ Midpoint B B′′
    have A′ B′ TS B B′′
    proof –
      have ¬ Col B A′ B′
        using Col-perm ‹¬ Col A′ B′ B› by blast
      moreover have ¬ Col B′′ A′ B′
        using ‹A′ B′ OS A B′′› col124--nos not-col-permutation-2 by blast
      moreover have ∃ T. Col T A′ B′ ∧ Bet B T B′′
        using ‹B′ Midpoint B B′′› col-trivial-3 midpoint-bet by blast
      ultimately show ?thesis
        by (simp add: TS-def)
    qed
    have A′ B′ OS B B′′
      using ‹A′ B′ OS A B′′› ‹A′ B′ OS A B› one-side-symmetry one-side-transitivity by blast
    have ¬ A′ B′ OS B B′′
      using ‹A′ B′ TS B B′′› l9-9-bis by blast
    then have False
      by (simp add: ‹A′ B′ OS B B′′›)
    then have ∃ X Y. (R PerpAt X Y A B ∧ X Y Perp P Q ∧ Coplanar A B C X ∧ Coplanar A B C Y)
      by auto
  }
  then show ?thesis
    using ‹B = B′′ ⟹ ∃X Y. R PerpAt X Y A B ∧ X Y Perp P Q ∧ Coplanar A B C X ∧ Coplanar A B C Y› ‹B
= B′′ ∨ B′ Midpoint B B′′› by blast
```

qed

lemma *l13-1*:
  assumes ¬ *Col A B C* **and**
    P Midpoint B C **and**
    Q Midpoint A C **and**
    R Midpoint A B
  shows
    ∃ *X Y*.(*R PerpAt X Y A B* ∧ *X Y Perp P Q*)
proof −
  obtain *X Y* **where** *R PerpAt X Y A B* ∧ *X Y Perp P Q* ∧ *Coplanar A B C X* ∧ *Coplanar A B C Y*
    using *l13-1-aux assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) **by** *blast*
  then show *?thesis* **by** *blast*
qed

lemma *per-lt*:
  assumes *A* ≠ *B* **and**
    C ≠ *B* **and**
    Per A B C
  shows *A B Lt A C* ∧ *C B Lt A C*
proof −
  have *B A Lt A C* ∧ *B C Lt A C*
    using *assms*(*1*) *assms*(*2*) *assms*(*3*) *l11-46* **by** *auto*
  then show *?thesis*
    using *lt-left-comm* **by** *blast*
qed

lemma *cong-perp-conga*:
  assumes *Cong A B C B* **and**
    A C Perp B P
  shows *A B P CongA C B P* ∧ *B P TS A C*
proof −
  have *P1*: *A* ≠ *C*
    using *assms*(*2*) *perp-distinct* **by** *auto*
  have *P2*: *B* ≠ *P*
    using *assms*(*2*) *perp-distinct* **by** *auto*
  have *P3*: *A* ≠ *B*
    by (*metis P1 assms*(*1*) *cong-diff-3*)
  have *P4*: *C* ≠ *B*
    using *P3 assms*(*1*) *cong-diff* **by** *blast*
  show *?thesis*
  proof *cases*
    assume *P5*: *Col A B C*
    have *P6*: ¬ *Col B A P*
      using *P3 P5 assms*(*2*) *col-transitivity-1 not-col-permutation-4 not-col-permutation-5 perp-not-col2* **by** *blast*
    have *Per P B A*
      using *P3 P5 Perp-perm assms*(*2*) *not-col-permutation-5 perp-col1 perp-per-1* **by** *blast*
    then have *P8*: *Per A B P*
      using *Per-cases* **by** *blast*
    have *Per P B C*
      using *P3 P5 P8 col-per2--per l8-2 l8-5* **by** *blast*
    then have *P10*: *Per C B P*
      using *Per-perm* **by** *blast*
    show *?thesis*
    proof −
      have *A B P CongA C B P*
        using *P2 P3 P4 P8 P10 l11-16* **by** *auto*
      moreover have *B P TS A C*
        by (*metis Col-cases P1 P5 P6 assms*(*1*) *bet--ts between-cong not-cong-2143 not-cong-4321 third-point*)
      ultimately show *?thesis*
        by *simp*
    qed
  next
    assume *T1*: ¬ *Col A B C*
    obtain *T* **where** *T2*: *T PerpAt A C B P*
      using *assms*(*2*) *perp-inter-perp-in* **by** *blast*

```
then have T3: Col A C T ∧ Col B P T
  using perp-in-col by auto
have T4: B ≠ T
  using Col-perm T1 T3 by blast
have T5: B T Perp A C
  using Perp-cases T3 T4 assms(2) perp-col1 by blast
{
  assume T5-1: A = T
  have B A Lt B C ∧ C A Lt B C
  proof −
    have B ≠ A
      using P3 by auto
    moreover have C ≠ A
      using P1 by auto
    moreover have Per B A C
      using T5 T5-1 perp-comm perp-per-1 by blast
    ultimately show ?thesis
      by (simp add: per-lt)
  qed
  then have False
    using Cong-perm assms(1) cong--nlt by blast
}
then have T6: A ≠ T by auto
{
  assume T6-1: C = T
  have B C Lt B A ∧ A C Lt B A
  proof −
    have B ≠ C
      using P4 by auto
    moreover have A ≠ C
      by (simp add: P1)
    moreover have Per B C A
      using T5 T6-1 perp-left-comm perp-per-1 by blast
    ultimately show ?thesis
      by (simp add: per-lt)
  qed
  then have False
    using Cong-perm assms(1) cong--nlt by blast
}
then have T7: C ≠ T by auto
have T8: T PerpAt B T T A
  by (metis Perp-in-cases T2 T3 T4 T6 perp-in-col-perp-in)
have T9: T PerpAt B T T C
  by (metis Col-cases T3 T7 T8 perp-in-col-perp-in)
have T10: Cong T A T C ∧ T A B CongA T C B ∧ T B A CongA T B C
proof −
  have A T B CongA C T B
  proof −
    have Per A T B
      using T2 perp-in-per-1 by auto
    moreover have Per C T B
      using T2 perp-in-per-3 by auto
    ultimately show ?thesis
      by (simp add: T4 T6 T7 l11-16)
  qed
  moreover have Cong A B C B
    by (simp add: assms(1))
  moreover have Cong T B T B
    by (simp add: cong-reflexivity)
  moreover have T B Le A B
  proof −
    have Per B T A
      using T8 perp-in-per by auto
    then have B T Lt B A ∧ A T Lt B A
      using T4 T6 per-lt by blast
    then show ?thesis
```

```
        using Le-cases Lt-def by blast
      qed
      ultimately show ?thesis
        using l11-52 by blast
    qed
    show ?thesis
    proof −
      have T11: A B P CongA C B P
      proof −
        have P B A CongA P B C
          using Col-cases P2 T10 T3 col-conga--conga by blast
        thus ?thesis
          using conga-comm by blast
      qed
      moreover have B P TS A C
      proof −
        have T12: A = C ∨ T Midpoint A C
          using T10 T3 l7-20-bis not-col-permutation-5 by blast
        {
          assume T Midpoint A C
          then have B P TS A C
            by (smt Col-perm P2 T1 T3 ‹A = T ⟹ False› ‹C = T ⟹ False› col2--eq l9-18 midpoint-bet)
        }
        then show ?thesis
          using P1 T12 by auto
      qed
      ultimately show ?thesis
        by simp
    qed
  qed
qed

lemma perp-per-bet:
  assumes ¬ Col A B C and

    Per A B C and
    P PerpAt P B A C
  shows Bet A P C
proof −
  have A ≠ C
    using assms(1) col-trivial-3 by auto
  then show ?thesis
    using assms(2) assms(3) l11-47 perp-in-left-comm by blast
qed

lemma ts-per-per-ts:
  assumes A B TS C D and
    Per B C A and
    Per B D A
  shows C D TS A B
proof −
  have P1: ¬ Col C A B
    using TS-def assms(1) by blast
  have P2: A ≠ B
    using P1 col-trivial-2 by auto
  obtain P where P3: Col P A B ∧ Bet C P D
    using TS-def assms(1) by blast
  have P4: C ≠ D
    using assms(1) not-two-sides-id by auto
  show ?thesis
  proof −
    {
      assume Col A C D
      then have C = D
        by (metis assms(1) assms(2) assms(3) col-per2-cases col-permutation-2 not-col-distincts ts-distincts)
      then have False
```

```
    using P4 by auto
}
then have ¬ Col A C D by auto
moreover have ¬ Col B C D
  using assms(1) assms(2) assms(3) per2-preserves-diff ts-distincts by blast
moreover have ∃ T. Col T C D ∧ Bet A T B
proof –
  have Col P C D
    using Col-def Col-perm P3 by blast
  moreover have Bet A P B
  proof –
    have ∃ X. Col A B X ∧ A B Perp C X
      using Col-perm P1 l8-18-existence by blast
    then obtain C' where P5: Col A B C' ∧ A B Perp C C' by blast
    have ∃ X. Col A B X ∧ A B Perp D X
      by (metis (no-types) Col-perm TS-def assms(1) l8-18-existence)
    then obtain D' where P6: Col A B D' ∧ A B Perp D D' by blast
    have P7: A ≠ C'
      using P5 assms(2) l8-7 perp-not-eq-2 perp-per-1 by blast
    have P8: A ≠ D'
      using P6 assms(3) l8-7 perp-not-eq-2 perp-per-1 by blast
    have P9: Bet A C' B
    proof –
      have ¬ Col A C B
        using Col-cases P1 by blast
      moreover have Per A C B
        by (simp add: assms(2) l8-2)
      moreover have C' PerpAt C' C A B
        using P5 Perp-in-perm l8-15-1 by blast
      ultimately show ?thesis
        using perp-per-bet by blast
    qed
    have P10: Bet A D' B
    proof –
      have ¬ Col A D B
        using P6 col-permutation-5 perp-not-col2 by blast
      moreover have Per A D B
        by (simp add: assms(3) l8-2)
      moreover have D' PerpAt D' D A B
        using P6 Perp-in-perm l8-15-1 by blast
      ultimately show ?thesis
        using perp-per-bet by blast
    qed
    show ?thesis
    proof cases
      assume P = C'
      then show ?thesis
        by (simp add: P9)
    next
      assume P ≠ C'
      show ?thesis
      proof cases
        assume P = D'
        then show ?thesis
          by (simp add: P10)
      next
        assume P ≠ D'
        show ?thesis
        proof cases
          assume A = P
          then show ?thesis
            by (simp add: between-trivial2)
        next
          assume A ≠ P
          show ?thesis
          proof cases
```

```
          assume B = P
          then show ?thesis
            using between-trivial by auto
        next
          assume B ≠ P
          have Bet C' P D'
          proof –
            have Bet C P D
              by (simp add: P3)
            moreover have P ≠ C'
              by (simp add: ‹P ≠ C'›)
            moreover have P ≠ D'
              by (simp add: ‹P ≠ D'›)
            moreover have Col C' P D'
              by (meson P2 P3 P5 P6 col3 col-permutation-2)
            moreover have Per P C' C
              using P3 P5 l8-16-1 l8-2 not-col-permutation-3 not-col-permutation-4 by blast
            moreover have Per P D' D
              by (metis P3 P6 calculation(3) not-col-permutation-2 perp-col2 perp-per-1)
            ultimately show ?thesis
              using per13-preserves-bet by blast
          qed
          then show ?thesis
            using P10 P9 bet3--bet by blast
        qed
      qed
    qed
  qed
 qed
 ultimately show ?thesis
  by auto
qed
ultimately show ?thesis
  by (simp add: TS-def)
qed
qed

lemma l13-2-1:
  assumes A B TS C D and
    Per B C A and
    Per B D A and
    Col C D E and
    A E Perp C D and
    C A B CongA D A B
  shows B A C CongA D A E ∧ B A D CongA C A E ∧ Bet C E D
proof –
  have P1: ¬ Col C A B
    using TS-def assms(1) by auto
  have P2: A ≠ C
    using P1 col-trivial-1 by blast
  have P3: A ≠ B
    using P1 col-trivial-2 by auto
  have P4: A ≠ D
    using assms(1) ts-distincts by auto
  have P5: Cong B C B D ∧ Cong A C A D ∧ C B A CongA D B A
  proof –
    have ¬ Col B A C
      by (simp add: P1 not-col-permutation-3)
    moreover have A C B CongA A D B
      using assms(1) assms(2) assms(3) l11-16 l8-2 ts-distincts by blast
    moreover have B A C CongA B A D
      by (simp add: assms(6) conga-comm)
    moreover have Cong B A B A
      by (simp add: cong-reflexivity)
    ultimately show ?thesis
      using l11-50-2 by blast
```

qed
then have *P6*: *C D Perp A B*
using *assms*(*1*) *assms*(*6*) *cong-conga-perp not-cong-2143* **by** *blast*
then have *P7*: *C D TS A B*
by (*simp add*: *assms*(*1*) *assms*(*2*) *assms*(*3*) *ts-per-per-ts*)
obtain *T1* **where** *P8*: *Col T1 C D* ∧ *Bet A T1 B*
using *P7 TS-def* **by** *auto*
obtain *T* **where** *P9*: *Col T A B* ∧ *Bet C T D*
using *TS-def assms*(*1*) **by** *blast*
have *P10*: *T1* = *T*
by (*metis* (*no-types*) *Col-def P1 P3 P8 P9 between-equality-2 between-trivial2 l6-16-1*)
have *P11*: *T* = *E*
proof −
have ¬ *Col A B C*
using *Col-perm P1* **by** *blast*
moreover have *C* ≠ *D*
using *assms*(*1*) *ts-distincts* **by** *blast*
moreover have *Col A B T*
using *Col-cases P9* **by** *auto*
moreover have *Col A B E*
by (*metis P7 Perp-cases P6 assms*(*1*) *assms*(*5*) *col-perp2-ncol-col col-trivial-3 not-col-permutation-3 one-side-not-col123 os-ts1324--os ts-ts-os*)
moreover have *Col C D T*
using *NCol-cases P9 bet-col* **by** *blast*
moreover have *Col C D E*
by (*simp add*: *assms*(*4*))
ultimately show *?thesis*
using *l6-21* **by** *blast*
qed
show *?thesis*
proof −
have *B A C CongA D A E*
proof −
have *A Out C C*
using *P2 out-trivial* **by** *auto*
moreover have *A Out B B*
using *P3 out-trivial* **by** *auto*
moreover have *A Out D D*
using *P4 out-trivial* **by** *auto*
moreover have *A Out E B*
by (*metis P10 P11 P7 P8 TS-def bet-out*)
ultimately show *?thesis*
by (*meson assms*(*6*) *conga-comm conga-right-comm l11-10*)
qed
moreover have *B A D CongA C A E*
proof −
have *C A E CongA D A B*
by (*meson Perp-cases P5 assms*(*5*) *assms*(*6*) *calculation cong-perp-conga conga-right-comm conga-trans not-cong-2143 not-conga-sym*)
then have *C A E CongA B A D*
by (*simp add*: *conga-right-comm*)
then show *?thesis*
by (*simp add*: *conga-sym*)
qed
moreover have *Bet C E D*
using *P11 P9* **by** *auto*
ultimately show *?thesis* **by** *simp*
qed
qed

lemma *triangle-mid-par*:
assumes ¬ *Col A B C* **and**
P Midpoint B C **and**
Q Midpoint A C
shows *A B ParStrict Q P*
proof −

obtain *R* **where** *P1*: *R Midpoint A B*
  using *midpoint-existence* **by** *auto*
then obtain *X Y* **where** *P2*: *R PerpAt X Y A B* ∧ *X Y Perp P Q* ∧ *Coplanar A B C X* ∧ *Coplanar A B C Y*
  using *l13-1-aux assms*(*1*) *assms*(*2*) *assms*(*3*) **by** *blast*
have *P3*: *Coplanar X Y A P* ∧ *Coplanar X Y A Q* ∧ *Coplanar X Y B P* ∧ *Coplanar X Y B Q*
proof −
  have *Coplanar A B C A*
    using *ncop-distincts* **by** *auto*
  moreover have *Coplanar A B C B*
    using *ncop-distincts* **by** *auto*
  moreover have *Coplanar A B C P*
    using *assms*(*2*) *coplanar-perm-21 midpoint--coplanar* **by** *blast*
  moreover have *Coplanar A B C Q*
    using *assms*(*3*) *coplanar-perm-11 midpoint--coplanar* **by** *blast*
  ultimately show *?thesis*
    using *P2 assms*(*1*) *coplanar-pseudo-trans* **by** *blast*
qed
have *P4*: *Col X Y R* ∧ *Col A B R*
  using *P2 perp-in-col* **by** *blast*
have *P5*: *R Y Perp A B* ∨ *X R Perp A B*
  using *P2 perp-in-perp-bis* **by** *auto*
have *P6*: *Col A R B*
  using *Col-perm P4* **by** *blast*
have *P7*: *X* ≠ *Y*
  using *P2 perp-not-eq-1* **by** *auto*
{
  assume *P8*: *R Y Perp A B*
  have *Col Y R X*
    using *P4 not-col-permutation-2* **by** *blast*
  then have *Y X Perp A B*
    using *P2 Perp-cases perp-in-perp* **by** *blast*
  then have *P10*: *X Y Perp A B*
    using *Perp-cases* **by** *blast*
  have *A B Par P Q*
  proof −
    have *Coplanar X Y A P*
      by (*simp add*: *P3*)
    moreover have *Coplanar X Y A Q*
      by (*simp add*: *P3*)
    moreover have *Coplanar X Y B P*
      by (*simp add*: *P3*)
    moreover have *Coplanar X Y B Q*
      by (*simp add*: *P3*)
    moreover have *A B Perp X Y*
      using *P10 Perp-cases* **by** *auto*
    moreover have *P Q Perp X Y*
      using *P2 Perp-cases* **by** *auto*
    ultimately show *?thesis*
      using *l12-9* **by** *blast*
  qed
  {
    assume *A B ParStrict P Q*
    then have *A B ParStrict Q P*
      using *Par-strict-perm* **by** *blast*
  }
  {
    assume *A* ≠ *B* ∧ *P* ≠ *Q* ∧ *Col A P Q* ∧ *Col B P Q*
    then have *Col A B P*
      using *l6-16-1 not-col-permutation-1* **by** *blast*
    then have *P* = *B*
      by (*metis Col-perm assms*(*1*) *assms*(*2*) *l6-16-1 midpoint-col*)
    then have *A B ParStrict Q P*
      using *assms*(*1*) *assms*(*2*) *col-trivial-2 is-midpoint-id* **by** *blast*
  }
  then have *A B ParStrict Q P*
    using *Par-def* ‹*A B Par P Q*› ‹*A B ParStrict P Q* ⟹ *A B ParStrict Q P*› **by** *auto*

```
  }
  {
    assume P10: X R Perp A B
    have Col X R Y
      by (simp add: Col-perm P4)
    then have P11: X Y Perp A B
      using P7 P10 perp-col by blast
    have A B Par P Q
    proof −
      have A B Perp X Y
        using P11 Perp-perm by blast
      moreover have P Q Perp X Y
        using P2 Perp-perm by blast
      ultimately show ?thesis
        using P3 l12-9 by blast
    qed
    {
      assume A B ParStrict P Q
      then have A B ParStrict Q P
        by (simp add: par-strict-right-comm)
    }
    {
      assume A ≠ B ∧ P ≠ Q ∧ Col A P Q ∧ Col B P Q
      then have Col A B P
        using Col-perm l6-16-1 by blast
      then have P = B
        by (metis Col-perm assms(1) assms(2) l6-16-1 midpoint-col)
      then have A B ParStrict Q P
        using assms(1) assms(2) col-trivial-2 is-midpoint-id by blast
    }
    then have A B ParStrict Q P
      using Par-def ‹A B Par P Q› ‹A B ParStrict P Q ⟹ A B ParStrict Q P› by auto
  }
  then show ?thesis
    using P5 ‹R Y Perp A B ⟹ A B ParStrict Q P› by blast
qed

lemma cop4-perp-in2--col:
  assumes Coplanar X Y A A′ and
    Coplanar X Y A B′ and
    Coplanar X Y B A′ and
    Coplanar X Y B B′ and
    P PerpAt A B X Y and
    P PerpAt A′ B′ X Y
  shows Col A B A′
proof −
  have P1: Col A B P ∧ Col X Y P
    using assms(5) perp-in-col by auto
  show ?thesis
  proof cases
    assume P2: A = P
    show ?thesis
    proof cases
      assume P3: P = X
      have Col B A′ P
      proof −
        have Coplanar Y B A′ P
          using P3 assms(3) ncoplanar-perm-18 by blast
        moreover have Y ≠ P
          using P3 assms(6) perp-in-distinct by blast
        moreover have Per B P Y
          using assms(5) perp-in-per-4 by auto
        moreover have Per A′ P Y
          using assms(6) perp-in-per-2 by auto
        ultimately show ?thesis
          using cop-per2--col by auto
```

qed
then show *?thesis*
using *Col-perm P2* **by** *blast*
next
assume *P4*: $P \neq X$
have *Col B A′ P*
proof −
have *Coplanar X B A′ P*
by (*metis P1 assms*(*3*) *assms*(*6*) *col2-cop--cop col-trivial-3 ncoplanar-perm-9 perp-in-distinct*)
moreover have *Per B P X*
using *assms*(*5*) *perp-in-per-3* **by** *auto*
moreover have *Per A′ P X*
using *assms*(*6*) *perp-in-per-1* **by** *auto*
ultimately show *?thesis*
using *cop-per2--col P4* **by** *auto*
qed
then show *?thesis*
using *Col-perm P2* **by** *blast*
qed
next
assume *P5*: $A \neq P$
have *P6*: *Per A P Y*
using *assms*(*5*) *perp-in-per-2* **by** *auto*
show *?thesis*
proof *cases*
assume *P7*: $P = A'$
have *P8*: *Per B′ P Y*
using *assms*(*6*) *perp-in-per-4* **by** *auto*
have *Col A B′ P*
proof −
have *Coplanar Y A B′ P*
using *assms*(*2*) **by** (*metis P1 assms*(*6*) *col-transitivity-2 coplanar-trans-1 ncop--ncols perp-in-distinct*)
then show *?thesis* **using** *P6 P8 cop-per2--col*
by (*metis assms*(*2*) *assms*(*5*) *assms*(*6*) *col-permutation-4 coplanar-perm-5 perp-in-distinct perp-in-per-1 perp-in-per-3*)
qed
then show *?thesis*
using *P1 P7* **by** *auto*
next
assume *T1*: $P \neq A'$
show *?thesis*
proof *cases*
assume *T2*: $Y = P$
{
assume *R1*: *Coplanar X P A A′* ∧ *P PerpAt A B X P* ∧ *P PerpAt A′ B′ X P* ∧ $A \neq P$
then have *R2*: *Per A P X*
using *perp-in-per-1* **by** *auto*
have *Per A′ P X*
using *R1 perp-in-per-1* **by** *auto*
then have *Col A B A′*
by (*metis R1 R2 PerpAt-def col-permutation-3 col-transitivity-2 cop-per2--col ncoplanar-perm-5*)
}
then show *?thesis*
using *P5 T1 T2 assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *assms*(*6*) **by** *blast*
next
assume *P10*: $Y \neq P$
have *Col A A′ P*
proof −
have *Coplanar Y A A′ P*
by (*metis P1 assms*(*1*) *assms*(*6*) *col2-cop--cop col-trivial-2 ncoplanar-perm-9 perp-in-distinct*)
moreover have *Per A P Y*
by (*simp add*: *P6*)
moreover have *Per A′ P Y*
using *assms*(*6*) *perp-in-per-2* **by** *auto*
ultimately show *?thesis*
using *cop-per2--col P10* **by** *auto*

qed
then show *?thesis*
using *P1 P5 col2--eq col-permutation-4* **by** *blast*
qed
qed
qed
qed

lemma *l13-2*:
assumes *A B TS C D* **and**
Per B C A **and**
Per B D A **and**
Col C D E **and**
A E Perp C D
shows *B A C CongA D A E* ∧ *B A D CongA C A E* ∧ *Bet C E D*
proof −
have *P2*: ¬ *Col C A B*
using *TS-def assms*(*1*) **by** *auto*
have *P3*: $C \neq D$
using *assms*(*1*) *not-two-sides-id* **by** *blast*
have *P4*: ∃ *C′*. *B A C CongA D A C′* ∧ *D A OS C′ B*
proof −
have ¬ *Col B A C*
using *Col-cases P2* **by** *auto*
moreover have ¬ *Col D A B*
using *TS-def assms*(*1*) **by** *blast*
ultimately show *?thesis*
by (*simp add*: *angle-construction-1*)
qed
then obtain *E′* **where** *P5*: *B A C CongA D A E′* ∧ *D A OS E′ B* **by** *blast*
have *P6*: $A \neq B$
using *P2 not-col-distincts* **by** *blast*
have *P7*: $A \neq C$
using *P2 not-col-distincts* **by** *blast*
have *P8*: $A \neq D$
using *P5 os-distincts* **by** *blast*
have *P9*: ((*A B TS C E′* ∧ *A E′ TS D B*) ∨ (*A B OS C E′* ∧ *A E′ OS D B* ∧ *C A B CongA D A E′* ∧ *B A E′ CongA E′ A B*)) ⟶ *C A E′ CongA D A B*
by (*metis P5 P6 conga-diff56 conga-left-comm conga-pseudo-refl l11-22*)
have *P10*: *C D TS A B*
by (*simp add*: *assms*(*1*) *assms*(*2*) *assms*(*3*) *ts-per-per-ts*)
have *P11*: ¬ *Col A C D*
using *P10 TS-def* **by** *auto*
obtain *T* **where** *P12*: *Col T A B* ∧ *Bet C T D*
using *TS-def assms*(*1*) **by** *blast*
obtain *T2* **where** *P13*: *Col T2 C D* ∧ *Bet A T2 B*
using *P10 TS-def* **by** *auto*
then have *P14*: *T* = *T2*
by (*metis Col-def Col-perm P12 P2 P3 P6 l6-16-1*)
have *P15*: *B InAngle D A C*
using *P10 assms*(*1*) *l11-24 ts2--inangle* **by** *blast*
have *P16*: *C A B LeA C A D*
by (*simp add*: *P10 assms*(*1*) *inangle--lea ts2--inangle*)
have *P17*: *E′ InAngle D A C*
proof −
have *D A E′ LeA D A C*
using *P16 P5 P7 P8 conga-left-comm conga-pseudo-refl l11-30* **by** *presburger*
moreover have *D A OS C E′*
by (*meson P11 P15 P5 col124--nos in-angle-one-side invert-one-side not-col-permutation-2 one-side-symmetry one-side-transitivity*)
ultimately show *?thesis*
by (*simp add*: *lea-in-angle*)
qed
obtain *E″* **where** *P18*: *Bet D E″ C* ∧ (*E″* = *A* ∨ *A Out E″ E′*)
using *InAngle-def P17* **by** *auto*
{

assume $E'' = A$
then have $B\ A\ C\ CongA\ D\ A\ E \wedge B\ A\ D\ CongA\ C\ A\ E \wedge Bet\ C\ E\ D$
using *Col-def P11 P18* **by** *auto*
}
{
assume *P19*: $A\ Out\ E''\ E'$
then have *P20*: $B\ A\ C\ CongA\ D\ A\ E''$
by (*meson OS-def P5 Tarski-neutral-dimensionless.out2--conga Tarski-neutral-dimensionless-axioms col-one-side-out col-trivial-2 l9-18-R1 not-conga one-side-reflexivity*)
have *P21*: $A \neq T$
using *P11 P13 P14* **by** *auto*
have $B\ A\ C\ CongA\ D\ A\ E \wedge B\ A\ D\ CongA\ C\ A\ E \wedge Bet\ C\ E\ D$
proof *cases*
assume *P22*: $E'' = T$
have *P23*: $C\ A\ B\ CongA\ D\ A\ B$
proof −
have $C\ A\ B\ CongA\ D\ A\ T$
using *P22 P20 conga-left-comm* **by** *blast*
moreover have $A\ Out\ C\ C$
using *P7 out-trivial* **by** *presburger*
moreover have $A\ Out\ B\ B$
using *P6 out-trivial* **by** *auto*
moreover have $A\ Out\ D\ D$
using *P8 out-trivial* **by** *auto*
moreover have $A\ Out\ B\ T$
using *Out-def P13 P14 P6 P21* **by** *blast*
ultimately show *?thesis*
using *l11-10* **by** *blast*
qed
then show *?thesis*
using *assms(1) assms(2) assms(3) assms(4) assms(5) l13-2-1* **by** *blast*
next
assume *P23A*: $E'' \neq T$
have *P24*: $D \neq E''$
using *P2 P20 col-trivial-3 ncol-conga-ncol not-col-permutation-3* **by** *blast*
{
assume *P24A*: $C = E''$
have *P24B*: $C\ A\ OS\ B\ D$
by (*meson P10 assms(1) invert-one-side ts-ts-os*)
have *P24C*: $A\ Out\ B\ D$
proof −
have $C\ A\ B\ CongA\ C\ A\ D$
using *P20 P24A conga-comm* **by** *blast*
moreover have $C\ A\ OS\ B\ D$
by (*simp add*: *P24B*)
ultimately show *?thesis*
using *conga-os--out* **by** *blast*
qed
then have *False*
using *Col-def P5 one-side-not-col124 out-col* **by** *blast*
}
then have *P25*: $C \neq E''$ **by** *auto*
have *P26*: $A \neq E''$
using *P19 out-diff1* **by** *auto*
{
assume $Col\ E''\ A\ B$
then have $E'' = T$
by (*smt P13 P14 P18 P2 P3 bet-col l6-21 not-col-permutation-2 not-col-permutation-3*)
then have *False*
using *P23A* **by** *auto*
}
then have *P27*: $\neg\ Col\ E''\ A\ B$ **by** *auto*
have $(A\ B\ TS\ C\ E'' \wedge A\ E''\ TS\ D\ B) \vee (A\ B\ OS\ C\ E'' \wedge A\ E''\ OS\ D\ B \wedge C\ A\ B\ CongA\ D\ A\ E'' \wedge B\ A\ E''\ CongA\ E''\ A\ B)$
proof *cases*
assume *P27-0*: $A\ B\ OS\ C\ E''$

```
        have A E'' OS D B
        proof -
          have P27-1: A E'' TS D C
            by (metis Col-def P10 P18 P24 TS-def P25 bet--ts invert-two-sides l6-16-1)
          moreover have A E'' TS B C
          proof -
            have A E'' TS T C
            proof -
              have ¬ Col T A E''
                by (metis NCol-cases P13 P14 P21 P27 bet-col col3 col-trivial-2)
              moreover have ¬ Col C A E''
                using P27-1 TS-def by auto
              moreover have ∃ T0. (Col T0 A E'' ∧ Bet T T0 C)
                by (meson P12 P18 P27-0 between-symmetry col-trivial-3 l5-3 one-side-chara)
              ultimately show ?thesis
                by (simp add: TS-def)
            qed
            moreover have A Out T B
              using Out-def P13 P14 P21 P6 by auto
            ultimately show ?thesis
              using col-trivial-1 l9-5 by blast
          qed
          ultimately show ?thesis
            using OS-def by auto
        qed
        thus ?thesis
          using P20 P27-0 conga-distinct conga-left-comm conga-pseudo-refl by blast
      next
        assume P27-2: ¬ A B OS C E''
        show ?thesis
        proof -
          have P27-3: A B TS C E''
            using P18 P2 P27-2 P27 assms(1) bet-cop--cop between-symmetry cop-nos--ts ts--coplanar by blast
          moreover have A E'' TS D B
          proof -
            have P27-3: A B OS D E''
              using P18 bet-ts--os between-symmetry calculation one-side-symmetry by blast
            have P27-4: A E'' TS T D
            proof -
              have ¬ Col T A E''
                by (metis NCol-cases P13 P14 P21 P27 bet-col col3 col-trivial-2)
              moreover have ¬ Col D A E''
                by (smt Col-def P11 P18 P24 P27-3 bet3--bet bet-col1 col3 col-permutation-5 col-two-sides-bet l5-1)
              moreover have ∃ T0. (Col T0 A E'' ∧ Bet T T0 D)
                  by (meson Bet-perm P12 P18 P27-3 bet-col1 bet-out--bet between-exchange3 col-trivial-3 not-bet-out
one-side-chara)
              ultimately show ?thesis
                by (simp add: TS-def)
            qed
            have A E'' TS B D
            proof -
              have A E'' TS T D
                using P27-4 by simp
              moreover have Col A A E''
                using col-trivial-1 by auto
              moreover have A Out T B
                using P13 P14 P21 bet-out by auto
              ultimately show ?thesis
                using l9-5 by blast
            qed
            thus ?thesis
              by (simp add: l9-2)
          qed
          ultimately show ?thesis
            by simp
        qed
```

qed
then have *P28*: *C A E'' CongA D A B* **using** *l11-22*
  by (*metis P20 P26 P6 conga-left-comm conga-pseudo-refl*)
obtain *C'* **where** *P29*: *Bet B C C'* ∧ *Cong C C' B C*
  using *segment-construction* **by** *blast*
obtain *D'* **where** *P30*: *Bet B D D'* ∧ *Cong D D' B D*
  using *segment-construction* **by** *blast*
have *P31*: *B A D Cong3 D' A D*
proof −
  have *Per A D B*
    by (*simp add*: *assms*(*3*) *l8-2*)
  then obtain *D''* **where** *P31-2*: *D Midpoint B D''* ∧ *Cong A B A D''*
    using *Per-def* **by** *auto*
  have *D Midpoint B D'*
    using *Cong-perm Midpoint-def P30* **by** *blast*
  then have *D' = D''*
    using *P31-2 symmetric-point-uniqueness* **by** *auto*
  thus *?thesis*
    using *Cong3-def Cong-perm P30 P31-2 cong-reflexivity* **by** *blast*
qed
then have *P32*: *B A D CongA D' A D*
  using *P6 P8 cong3-conga* **by** *auto*
have *B A C Cong3 C' A C*
proof −
  obtain *C''* **where** *P33-1*: *C Midpoint B C''* ∧ *Cong A B A C''*
    using *Per-def assms*(*2*) *l8-2* **by** *blast*
  have *C Midpoint B C'*
    using *Cong-perm Midpoint-def P29* **by** *blast*
  then have *C' = C''*
    using *P33-1 symmetric-point-uniqueness* **by** *auto*
  thus *?thesis*
    using *Cong3-def Cong-perm P29 P33-1 cong-reflexivity* **by** *blast*
qed
then have *P34*: *B A C CongA C' A C*
  using *P6 P7 cong3-conga* **by** *auto*
have *P35*: *E'' A C' CongA D' A E''*
proof −
  have (*A C TS E'' C'* ∧ *A D TS D' E''*) ∨ (*A C OS E'' C'* ∧ *A D OS D' E''*)
  proof −
    have *P35-1*: *C A OS D E''*
      by (*metis Col-perm P11 P18 P25 bet-out between-symmetry one-side-symmetry out-one-side*)
    have *P35-2*: *C A OS B D*
      using *P10 assms*(*1*) *one-side-symmetry ts-ts-os* **by** *blast*
    have *P35-3*: *C A TS B C'*
      by (*metis P2 P29 bet--ts cong-diff-4 not-col-distincts*)
    have *P35-4*: *C A OS B E''*
      using *P35-1 P35-2 one-side-transitivity* **by** *blast*
    have *P35-5*: *D A OS C E''*
      by (*metis Col-perm P18 P24 P35-1 bet2--out l5-1 one-side-not-col123 out-one-side*)
    have *P35-6*: *D A OS B C*
      by (*simp add*: *P10 assms*(*1*) *invert-two-sides l9-2 one-side-symmetry ts-ts-os*)
    have *P35-7*: *D A TS B D'*
      by (*metis P30 TS-def assms*(*1*) *bet--ts cong-diff-3 ts-distincts*)
    have *P35-8*: *D A OS B E''*
      using *P35-5 P35-6 one-side-transitivity* **by** *blast*
    have *P35-9*: *A C TS E'' C'*
      using *P35-3 P35-4 invert-two-sides l9-8-2* **by** *blast*
    have *A D TS D' E''*
      using *P35-7 P35-8 invert-two-sides l9-2 l9-8-2* **by** *blast*
    thus *?thesis*
      using *P35-9* **by** *simp*
  qed
  moreover have *E'' A C CongA D' A D*
  proof −
    have *E'' A C CongA B A D*
      by (*simp add*: *P28 conga-comm*)

moreover have *B A D CongA D′ A D*
by (*simp add*: *P32*)
ultimately show *?thesis*
using *conga-trans* **by** *blast*
qed
moreover have *C A C′ CongA D A E′′*
proof −
have *D A E′′ CongA C A C′*
proof −
have *D A E′′ CongA B A C*
by (*simp add*: *P20 conga-sym*)
moreover have *B A C CongA C A C′*
by (*simp add*: *P34 conga-right-comm*)
ultimately show *?thesis*
using *conga-trans* **by** *blast*
qed
thus *?thesis*
using *not-conga-sym* **by** *blast*
qed
ultimately show *?thesis*
using *l11-22* **by** *auto*
qed
have *P36*: $D' \neq B$
using *P30 assms*(*1*) *bet-neq32--neq ts-distincts* **by** *blast*
have *P37*: $C' \neq B$
using *P29 assms*(*1*) *bet-neq32--neq ts-distincts* **by** *blast*
then have *P38*: ¬ *Col C′ D′ B*
by (*metis Col-def P10 P29 P30 P36 TS-def col-transitivity-2*)
have *P39*: *C′ D′ ParStrict C D*
proof −
have ¬ *Col C′ D′ B*
by (*simp add*: *P38*)
moreover have *D Midpoint D′ B*
using *P30 l7-2 midpoint-def not-cong-3412* **by** *blast*
moreover have *C Midpoint C′ B*
using *P29 l7-2 midpoint-def not-cong-3412* **by** *blast*
ultimately show *?thesis*
using *triangle-mid-par* **by** *auto*
qed
have *P40*: *A E′′ TS C D*
by (*metis Bet-perm Col-def P10 P18 P24 TS-def* ‹$C = E'' \Longrightarrow$ *False*› *bet--ts col-transitivity-2 invert-two-sides*)
have *P41*: *B A TS C D*
by (*simp add*: *assms*(*1*) *invert-two-sides*)
have *P42*: *A B OS C C′*
proof −
have ¬ *Col A B C*
by (*simp add*: *P2 not-col-permutation-1*)
moreover have *Col A B B*
by (*simp add*: *col-trivial-2*)
moreover have *B Out C C′*
by (*metis P29 P37 bet-out cong-identity*)
ultimately show *?thesis*
using *out-one-side-1* **by** *blast*
qed
have *P43*: *A B OS D D′* **using** *out-one-side-1*
by (*metis Col-perm P30 TS-def assms*(*1*) *bet-out col-trivial-1*)
then have *P44*: *A B OS D D′* **using** *invert-two-sides* **by** *blast*
have *P45*: *A B TS C′ D*
using *P42 assms*(*1*) *l9-8-2* **by** *blast*
then have *P46*: *A B TS C′ D′*
using *P44 l9-2 l9-8-2* **by** *blast*
have *P47*: *C′ D′ Perp A E′′*
proof −
have *A E′′ TS C′ D′*
proof −
have *A Out C′ D′* ∨ *E′′ A TS C′ D′*

```
        proof -
          have E'' A C' CongA E'' A D'
            by (simp add: P35 conga-right-comm)
          moreover have Coplanar E'' A C' D'
          proof -
            have f1: B A OS C C'
              by (metis P42 invert-one-side)
            have f2: Coplanar B A C' C
              by (meson P42 ncoplanar-perm-7 os--coplanar)
            have f3: Coplanar D' A C' D
            by (meson P44 P46 col124--nos coplanar-trans-1 invert-one-side ncoplanar-perm-7 os--coplanar ts--coplanar)
            have Coplanar D' A C' C
             using f2 f1 by (meson P46 col124--nos coplanar-trans-1 ncoplanar-perm-6 ncoplanar-perm-8 ts--coplanar)
            then show ?thesis
              using f3 by (meson P18 bet-cop2--cop ncoplanar-perm-6 ncoplanar-perm-7 ncoplanar-perm-8)
          qed
          ultimately show ?thesis using conga-cop--or-out-ts
            by simp
        qed
        then show ?thesis
          using P46 col-two-sides-bet invert-two-sides not-bet-and-out out-col by blast
      qed
      moreover have Cong C' A D' A
        using Cong3-def P31 ‹B A C Cong3 C' A C› cong-inner-transitivity by blast
      moreover have C' A E'' CongA D' A E''
        by (simp add: P35 conga-left-comm)
      ultimately show ?thesis
        by (simp add: cong-conga-perp)
    qed
    have T1: Cong A C' A D'
    proof -
      have Cong A C' A B
        using Cong3-def Cong-perm ‹B A C Cong3 C' A C› by blast
      moreover have Cong A D' A B
        using Cong3-def P31 not-cong-4321 by blast
      ultimately show ?thesis
        using Cong-perm ‹Cong A C' A B› ‹Cong A D' A B› cong-inner-transitivity by blast
    qed
    obtain R where T2: R Midpoint C' D'
      using midpoint-existence by auto
    have ∃ X Y. (R PerpAt X Y C' D' ∧ X Y Perp D C ∧ Coplanar C' D' B X ∧ Coplanar C' D' B Y)
    proof -
      have ¬ Col C' D' B
        by (simp add: P38)
      moreover have D Midpoint D' B
        using P30 l7-2 midpoint-def not-cong-3412 by blast
      moreover have C Midpoint C' B
        using Cong-perm Mid-perm Midpoint-def P29 by blast
      moreover have R Midpoint C' D'
        by (simp add: T2)
      ultimately show ?thesis using l13-1-aux by blast
    qed
    then obtain X Y where T3: R PerpAt X Y C' D' ∧ X Y Perp D C ∧ Coplanar C' D' B X ∧ Coplanar C' D'
B Y
      by blast
    then have X ≠ Y
      using perp-not-eq-1 by blast
    have C D Perp A E''
    proof cases
      assume A = R
      then have W1: A PerpAt C' D' A E''
        using Col-def P47 T2 between-trivial2 l8-14-2-1b-bis midpoint-col by blast
      have Coplanar B C' D' E''
      proof -
        have ¬ Col B C D
          using P10 TS-def by auto
```

moreover have *Coplanar B C D B*
using *ncop-distincts* **by** *auto*
moreover have *Coplanar B C D C′*
using *P29 bet-col ncop--ncols* **by** *blast*
moreover have *Coplanar B C D D′*
using *P30 bet-col ncop--ncols* **by** *blast*
moreover have *Coplanar B C D E′′*
by (*simp add*: *P18 bet--coplanar coplanar-perm-22*)
ultimately show *?thesis*
using *coplanar-pseudo-trans* **by** *blast*
qed
have *Coplanar C′ D′ X E′′*
proof −
have ¬ *Col B C′ D′*
by (*simp add*: *P38 not-col-permutation-2*)
moreover have *Coplanar B C′ D′ X*
using *T3 ncoplanar-perm-8* **by** *blast*
moreover have *Coplanar B C′ D′ E′′*
by (*simp add*: ‹*Coplanar B C′ D′ E′′*›)
ultimately show *?thesis*
using *coplanar-trans-1* **by** *blast*
qed
have *Coplanar C′ D′ Y E′′*
proof −
have ¬ *Col B C′ D′*
by (*simp add*: *P38 not-col-permutation-2*)
moreover have *Coplanar B C′ D′ Y*
by (*simp add*: *T3 coplanar-perm-12*)
moreover have *Coplanar B C′ D′ E′′*
by (*simp add*: ‹*Coplanar B C′ D′ E′′*›)
ultimately show *?thesis*
using *coplanar-trans-1* **by** *blast*
qed
have *Coplanar C′ D′ X A*
proof −
have *Col C′ D′ A*
using *T2* ‹*A* = *R*› *midpoint-col not-col-permutation-2* **by** *blast*
moreover have *Col X A A*
by (*simp add*: *col-trivial-2*)
ultimately show *?thesis*
using *ncop--ncols* **by** *blast*
qed
have *Coplanar C′ D′ Y A*
proof −
have *Col C′ D′ A*
using *T2* ‹*A* = *R*› *midpoint-col not-col-permutation-2* **by** *blast*
moreover have *Col Y A A*
by (*simp add*: *col-trivial-2*)
ultimately show *?thesis*
using *ncop--ncols* **by** *blast*
qed
have *Col X Y A*
proof −
have *Coplanar C′ D′ X A*
by (*simp add*: ‹*Coplanar C′ D′ X A*›)
moreover have *Coplanar C′ D′ X E′′*
by (*simp add*: ‹*Coplanar C′ D′ X E′′*›)
moreover have *Coplanar C′ D′ Y A*
by (*simp add*: ‹*Coplanar C′ D′ Y A*›)
moreover have *Coplanar C′ D′ Y E′′*
by (*simp add*: ‹*Coplanar C′ D′ Y E′′*›)
moreover have *A PerpAt X Y C′ D′*
using *T3* ‹*A* = *R*› *Perp-in-cases* **by** *auto*
moreover have *A PerpAt A E′′ C′ D′*
using *Perp-in-cases* ‹*A PerpAt C′ D′ A E′′*› **by** *blast*
ultimately show *?thesis*

```
      using cop4-perp-in2--col by blast
  qed
  have Col X Y E''
  proof -
    have Coplanar C' D' X E''
      using ‹Coplanar C' D' X E''› by auto
    moreover have Coplanar C' D' X A
      by (simp add: ‹Coplanar C' D' X A›)
    moreover have Coplanar C' D' Y E''
      by (simp add: ‹Coplanar C' D' Y E''›)
    moreover have Coplanar C' D' Y A
      using ‹Coplanar C' D' Y A› by auto
    moreover have A PerpAt X Y C' D'
      using T3 ‹A = R› Perp-in-cases by auto
    moreover have A PerpAt E'' A C' D'
      using Perp-in-perm W1 by blast
    ultimately show ?thesis
      using cop4-perp-in2--col by blast
  qed
  have A E'' Perp C D
  proof cases
    assume Y = A
    show ?thesis
    proof -
      have A ≠ E''
        by (simp add: P26)
      moreover have A X Perp C D
        using T3 Perp-cases ‹Y = A› by blast
      moreover have Col A X E''
        using Col-perm ‹Col X Y E''› ‹Y = A› by blast
      ultimately show ?thesis
        using perp-col by blast
    qed
  next
    assume Y ≠ A
    show ?thesis
    proof -
      have A ≠ E''
        by (simp add: P26)
      moreover have A Y Perp C D
      proof -
        have Y X Perp C D
          using T3 by (simp add: perp-comm)
        then have Y A Perp C D
          using ‹Col X Y A› ‹Y ≠ A› col-trivial-2 perp-col2 perp-left-comm by blast
        then show ?thesis
          using Perp-cases by blast
      qed
      moreover have Col A Y E''
        using Col-perm ‹Col X Y A› ‹Col X Y E''› ‹X ≠ Y› col-transitivity-2 by blast
      ultimately show ?thesis
        using perp-col by blast
    qed
  qed
  thus ?thesis
    using Perp-perm by blast
next
  assume A ≠ R
  have R ≠ C'
    using P46 T2 is-midpoint-id ts-distincts by blast
  have Per A R C' using T1 T2 Per-def by blast
  then have R PerpAt A R R C'
    by (simp add: ‹A ≠ R› ‹R ≠ C'› per-perp-in)
  then have R PerpAt R C' A R
    using Perp-in-perm by blast
  then have R C' Perp A R ∨ R R Perp A R
```

using *perp-in-perp* **by** *auto*

{

assume $R\ C'\ Perp\ A\ R$

then have $C'\ R\ Perp\ A\ R$

by (*simp add*: ‹$R\ C'\ Perp\ A\ R$› *Perp-perm*)

have $C'\ D'\ Perp\ R\ A$

by (*metis P47 T2* ‹$A \neq R$› ‹$Per\ A\ R\ C'$› ‹$R \neq C'$› *col-per-perp midpoint-col perp-distinct perp-right-comm*)

then have $R\ PerpAt\ C'\ D'\ R\ A$

using *T2 l8-14-2-1b-bis midpoint-col not-col-distincts* **by** *blast*

have $Col\ B\ D\ D'$

by (*simp add*: *Col-def P30*)

have $Col\ B\ C\ C'$

using *Col-def P29* **by** *auto*

have $Col\ D\ E''\ C$

using *P18 bet-col* **by** *auto*

have $Col\ R\ C'\ D'$

using ‹$R\ PerpAt\ C'\ D'\ R\ A$› **by** (*simp add*: *T2 midpoint-col*)

have $Col\ A\ E''\ E'$

by (*simp add*: *P19 out-col*)

have $Coplanar\ C'\ D'\ X\ A$

proof –

have ¬ $Col\ B\ C'\ D'$

using *Col-perm P38* **by** *blast*

moreover have $Coplanar\ B\ C'\ D'\ X$

using *T3 ncoplanar-perm-8* **by** *blast*

moreover have $Coplanar\ B\ C'\ D'\ A$

using *P46 ncoplanar-perm-18 ts--coplanar* **by** *blast*

ultimately show *?thesis*

using *coplanar-trans-1* **by** *auto*

qed

have $Coplanar\ C'\ D'\ Y\ A$

proof –

have ¬ $Col\ B\ C'\ D'$

using *Col-perm P38* **by** *blast*

moreover have $Coplanar\ B\ C'\ D'\ Y$

using *T3 ncoplanar-perm-8* **by** *blast*

moreover have $Coplanar\ B\ C'\ D'\ A$

using *P46 ncoplanar-perm-18 ts--coplanar* **by** *blast*

ultimately show *?thesis*

using *coplanar-trans-1* **by** *auto*

qed

have $Coplanar\ C'\ D'\ X\ R$

proof –

have $Col\ C'\ D'\ R$

using *Col-perm* ‹$Col\ R\ C'\ D'$› **by** *blast*

moreover have $Col\ X\ R\ R$

by (*simp add*: *col-trivial-2*)

ultimately show *?thesis*

using *ncop--ncols* **by** *blast*

qed

have $Coplanar\ C'\ D'\ Y\ R$

using *Col-perm T2 midpoint-col ncop--ncols* **by** *blast*

have $Col\ X\ Y\ A$

proof –

have $R\ PerpAt\ X\ Y\ C'\ D'$

using *T3* **by** *simp*

moreover have $R\ PerpAt\ A\ R\ C'\ D'$

using *Perp-in-perm* ‹$R\ PerpAt\ C'\ D'\ R\ A$› **by** *blast*

ultimately show *?thesis*

using ‹$Coplanar\ C'\ D'\ Y\ R$› ‹$Coplanar\ C'\ D'\ X\ R$› *cop4-perp-in2--col* ‹$Coplanar\ C'\ D'\ X\ A$› ‹$Coplanar\ C'\ D'\ Y\ A$› **by** *blast*

qed

have *Z1*: $Col\ X\ Y\ R$

using *T3 perp-in-col* **by** *blast*

have $Col\ A\ E''\ R$

proof –

have *Coplanar* C' D' E'' *R*
using *Col-cases* ‹*Col R* C' D'› *ncop--ncols* **by** *blast*
moreover have *A* E'' *Perp* C' D'
using *P47 Perp-perm* **by** *blast*
moreover have *A R Perp* C' D'
using *Perp-perm* ‹C' D' *Perp R A*› **by** *blast*
ultimately show *?thesis*
using *cop-perp2--col* **by** *blast*
qed
then have *Col X Y* E'' **using** *Z1*
by (*metis* (*full-types*) ‹$A \neq R$› ‹*Col X Y A*› *col-permutation-4 col-trivial-2 l6-21*)
have *Col A* E'' *R*
proof −
have *Coplanar* C' D' E'' *R*
using *Col-cases* ‹*Col R* C' D'› *ncop--ncols* **by** *blast*
moreover have *A* E'' *Perp* C' D'
using *P47 Perp-perm* **by** *blast*
moreover have *A R Perp* C' D'
using *Perp-perm* ‹C' D' *Perp R A*› **by** *blast*
ultimately show *?thesis*
using *cop-perp2--col* **by** *blast*
qed
have *Col A R X*
using ‹*Col X Y A*› ‹*Col X Y R*› ‹$X \neq Y$› *col-transitivity-1 not-col-permutation-3* **by** *blast*
have *Col A R Y*
using ‹*Col X Y A*› ‹*Col X Y R*› ‹$X \neq Y$› *col-transitivity-2 not-col-permutation-3* **by** *blast*
have *A* E'' *Perp C D*
proof *cases*
assume $X = A$
show *?thesis*
proof −
have $A \neq E''$
by (*simp add*: *P26*)
moreover have *A Y Perp C D*
using *T3* ‹$X = A$› *perp-right-comm* **by** *blast*
moreover have *Col A Y* E''
using *Col-perm* ‹$A \neq R$› ‹*Col A* E'' *R*› ‹*Col A R Y*› *col-transitivity-1* **by** *blast*
ultimately show *?thesis*
using *perp-col* **by** *auto*
qed
next
assume $X \neq A$
show *?thesis*
proof −
have *A X Perp C D*
by (*smt P3 T3* ‹*Col X Y A*› ‹$X \neq A$› *col-trivial-2 col-trivial-3 perp-col4*)
moreover have *Col A X* E''
using *Col-perm* ‹$A \neq R$› ‹*Col A* E'' *R*› ‹*Col A R X*› *col-transitivity-1* **by** *blast*
ultimately show *?thesis*
using *P26 perp-col* **by** *blast*
qed
qed
}
{
assume *R R Perp A R*
then have *A* E'' *Perp C D*
using *perp-distinct* **by** *blast*
}
then have *A* E'' *Perp C D*
using *Perp-cases* ‹*R* C' *Perp A R* $\Longrightarrow$ *A* E'' *Perp C D*› ‹*R* C' *Perp A R* $\vee$ *R R Perp A R*› **by** *auto*
then show *?thesis*
using *Perp-perm* **by** *blast*
qed
show *?thesis*
proof −
have *Col A E* E''

```
proof –
  have Coplanar C D E E'
    using assms(4) col--coplanar by auto
  moreover have A E Perp C D
    using assms(5) by auto
  moreover have A E'' Perp C D
    using Perp-perm ‹C D Perp A E''› by blast
  ultimately show ?thesis
    by (meson P11 col-perp2-ncol-col col-trivial-3 not-col-permutation-2)
qed
moreover have E'' = E
proof –
  have f1: C = E'' ∨ Col C E'' D
    by (metis P18 bet-out-1 out-col)
  then have f2: C = E'' ∨ Col C E'' E
    using Col-perm P3 assms(4) col-transitivity-1 by blast
  have ∀ p. (C = E'' ∨ Col C p D) ∨ ¬ Col C E'' p
    using f1 by (meson col-transitivity-1)
  then have ∃ p. ¬ Col E'' p A ∧ Col E'' E p
    using f2 by (metis (no-types) Col-perm P11 assms(4))
  then show ?thesis
    using Col-perm calculation col-transitivity-1 by blast
qed
ultimately show ?thesis
 by (metis Bet-perm P18 P20 P28 Tarski-neutral-dimensionless.conga-left-comm Tarski-neutral-dimensionless-axioms
not-conga-sym)
      qed
    qed
    then have B A C CongA D A E ∧ B A D CongA C A E ∧ Bet C E D
      by blast
  }
  thus ?thesis
    using P18 ‹E'' = A ⟹ B A C CongA D A E ∧ B A D CongA C A E ∧ Bet C E D› by blast
qed

lemma perp2-refl:
  assumes A ≠ B
  shows P Perp2 A B A B
proof cases
  assume Col A B P
  obtain X where ¬ Col A B X
    using assms not-col-exists by blast
  then obtain Q where A B Perp Q P ∧ A B OS X Q
    using ‹Col A B P› l10-15 by blast
  thus ?thesis
    using Perp2-def Perp-cases col-trivial-3 by blast
next
  assume ¬ Col A B P
  then obtain Q where Col A B Q ∧ A B Perp P Q
    using l8-18-existence by blast
  thus ?thesis
    using Perp2-def Perp-cases col-trivial-3 by blast
qed

lemma perp2-sym:
  assumes P Perp2 A B C D
  shows P Perp2 C D A B
proof –
  obtain X Y where Col P X Y ∧ X Y Perp A B ∧ X Y Perp C D
    using Perp2-def assms by auto
  thus ?thesis
    using Perp2-def by blast
qed

lemma perp2-left-comm:
  assumes P Perp2 A B C D
```

shows *P Perp2 B A C D*
proof –
obtain *X Y* **where** *Col P X Y* ∧ *X Y Perp A B* ∧ *X Y Perp C D*
using *Perp2-def assms* **by** *auto*
thus *?thesis*
using *Perp2-def perp-right-comm* **by** *blast*
qed

lemma *perp2-right-comm*:
assumes *P Perp2 A B C D*
shows *P Perp2 A B D C*
proof –
obtain *X Y* **where** *Col P X Y* ∧ *X Y Perp A B* ∧ *X Y Perp C D*
using *Perp2-def assms* **by** *auto*
thus *?thesis*
using *Perp2-def perp-right-comm* **by** *blast*
qed

lemma *perp2-comm*:
assumes *P Perp2 A B C D*
shows *P Perp2 B A D C*
proof –
obtain *X Y* **where** *Col P X Y* ∧ *X Y Perp A B* ∧ *X Y Perp C D*
using *Perp2-def assms* **by** *auto*
thus *?thesis*
using *assms perp2-left-comm perp2-right-comm* **by** *blast*
qed

lemma *perp2-pseudo-trans*:
assumes *P Perp2 A B C D* **and**
P Perp2 C D E F **and**
¬ *Col C D P*
shows *P Perp2 A B E F*
proof –
obtain *X Y* **where** *P1*: *Col P X Y* ∧ *X Y Perp A B* ∧ *X Y Perp C D*
using *Perp2-def assms*(*1*) **by** *auto*
obtain *X′ Y′* **where** *P2*: *Col P X′ Y′* ∧ *X′ Y′ Perp C D* ∧ *X′ Y′ Perp E F*
using *Perp2-def assms*(*2*) **by** *auto*
have *X Y Par X′ Y′*
proof –
have *Coplanar P C D X*
proof *cases*
assume *X* = *P*
thus *?thesis*
using *ncop-distincts* **by** *blast*
next
assume *X* ≠ *P*
then have *X P Perp C D*
using *Col-cases P1 perp-col* **by** *blast*
then have *Coplanar X P C D*
by (*simp add*: *perp--coplanar*)
thus *?thesis*
using *ncoplanar-perm-18* **by** *blast*
qed
have *Coplanar P C D Y*
proof *cases*
assume *Y* = *P*
thus *?thesis*
using *ncop-distincts* **by** *blast*
next
assume *Y* ≠ *P*
then have *Y P Perp C D*
by (*metis* (*full-types*) *Col-cases P1 Perp-cases col-transitivity-2 perp-col2*)
then have *Coplanar Y P C D*
by (*simp add*: *perp--coplanar*)
thus *?thesis*

```
      using ncoplanar-perm-18 by blast
    qed
    have Coplanar P C D X'
    proof cases
      assume X' = P
      thus ?thesis
        using ncop-distincts by blast
    next
      assume X' ≠ P
      then have X' P Perp C D
        using Col-cases P2 perp-col by blast
      then have Coplanar X' P C D
        by (simp add: perp--coplanar)
      thus ?thesis
        using ncoplanar-perm-18 by blast
    qed
    have Coplanar P C D Y'
    proof cases
      assume Y' = P
      thus ?thesis
        using ncop-distincts by blast
    next
      assume Y' ≠ P
      then have Y' P Perp C D
        by (metis (full-types) Col-cases P2 Perp-cases col-transitivity-2 perp-col2)
      then have Coplanar Y' P C D
        by (simp add: perp--coplanar)
      thus ?thesis
        using ncoplanar-perm-18 by blast
    qed
    show ?thesis
    proof -
      have Coplanar C D X X'
        using Col-cases ‹Coplanar P C D X'› ‹Coplanar P C D X› assms(3) coplanar-trans-1 by blast
      moreover have Coplanar C D X Y'
        using Col-cases ‹Coplanar P C D X› ‹Coplanar P C D Y'› assms(3) coplanar-trans-1 by blast
      moreover have Coplanar C D Y X'
        using Col-cases ‹Coplanar P C D X'› ‹Coplanar P C D Y› assms(3) coplanar-trans-1 by blast
      moreover have Coplanar C D Y Y'
        using Col-cases ‹Coplanar P C D Y'› ‹Coplanar P C D Y› assms(3) coplanar-trans-1 by blast
      ultimately show ?thesis
        using l12-9 P1 P2 by blast
    qed
  qed
  thus ?thesis
  proof -
    {
      assume X Y ParStrict X' Y'
      then have Col X X' Y'
        using P1 P2 ‹X Y ParStrict X' Y'› par-not-col by blast
    }
    then have Col X X' Y'
      using Par-def ‹X Y Par X' Y'› by blast
    moreover have Col Y X' Y'
    proof -
      {
        assume X Y ParStrict X' Y'
        then have Col Y X' Y'
          using P1 P2 ‹X Y ParStrict X' Y'› par-not-col by blast
      }
      thus ?thesis
        using Par-def ‹X Y Par X' Y'› by blast
    qed
    moreover have X ≠ Y
      using P1 perp-not-eq-1 by auto
    ultimately show ?thesis
```

by (*meson Perp2-def P1 P2 col-permutation-1 perp-col2*)
qed
qed

lemma *col-cop-perp2--pars-bis*:
assumes ¬ *Col A B P* **and**
Col C D P **and**
Coplanar A B C D **and**
P Perp2 A B C D
shows *A B ParStrict C D*
proof –
obtain *X Y* **where** *P1*: *Col P X Y* ∧ *X Y Perp A B* ∧ *X Y Perp C D*
using *Perp2-def assms*(*4*) **by** *auto*
then have *Col X Y P*
using *Col-perm* **by** *blast*
obtain *Q* **where** $X \neq Q \wedge Y \neq Q \wedge P \neq Q \wedge$ *Col X Y Q*
using ‹*Col X Y P*› *diff-col-ex3* **by** *blast*
thus *?thesis*
by (*smt P1 Perp-perm assms*(*1*) *assms*(*2*) *assms*(*3*) *col-cop-perp2-pars col-permutation-1 col-transitivity-2 not-col-distincts perp-col4 perp-distinct*)
qed

lemma *perp2-preserves-bet23*:
assumes *Bet PO A B* **and**
Col PO A′ B′ **and**
¬ *Col PO A A′* **and**
PO Perp2 A A′ B B′
shows *Bet PO A′ B′*
proof –
have $A \neq A'$
using *assms*(*3*) *not-col-distincts* **by** *auto*
show *?thesis*
proof *cases*
assume $A' = B'$
thus *?thesis*
using *between-trivial* **by** *auto*
next
assume $A' \neq B'$
{
assume $A = B$
then obtain *X Y* **where** *P1*: *Col PO X Y* ∧ *X Y Perp A A′* ∧ *X Y Perp A B′*
using *Perp2-def assms*(*4*) **by** *blast*
have *Col A A′ B′*
proof –
have *Coplanar X Y A′ B′*
using *Col-cases Coplanar-def P1 assms*(*2*) **by** *auto*
moreover have *A A′ Perp X Y*
using *P1 Perp-perm* **by** *blast*
moreover have *A B′ Perp X Y*
using *P1 Perp-perm* **by** *blast*
ultimately show *?thesis*
using *cop-perp2--col* **by** *blast*
qed
then have *False*
using *Col-perm* ‹$A' \neq B'$› *assms*(*2*) *assms*(*3*) *l6-16-1* **by** *blast*
}
then have $A \neq B$ **by** *auto*
have *A A′ Par B B′*
proof –
obtain *X Y* **where** *P2*: *Col PO X Y* ∧ *X Y Perp A A′* ∧ *X Y Perp B B′*
using *Perp2-def assms*(*4*) **by** *auto*
then have *Coplanar X Y A B*
using *Coplanar-def assms*(*1*) *bet-col not-col-permutation-2* **by** *blast*
show *?thesis*
proof –
have *Coplanar X Y A B′*

by (*metis* (*full-types*) *Col-cases P2 assms*(*2*) *assms*(*3*) *col-cop2--cop col-trivial-3 ncop--ncols perp--coplanar*)
moreover have *Coplanar X Y A′ B*
proof *cases*
assume *Col A X Y*
then have *Col Y X A*
by (*metis* (*no-types*) *Col-cases*)
then show *?thesis*
by (*metis Col-cases P2 assms*(*1*) *assms*(*3*) *bet-col colx ncop--ncols not-col-distincts*)
next
assume ¬ *Col A X Y*
moreover have *Coplanar A X Y A′*
using *Coplanar-def P2 perp-inter-exists* **by** *blast*
moreover have *Coplanar A X Y B*
using ‹*Coplanar X Y A B*› *ncoplanar-perm-8* **by** *blast*
ultimately show *?thesis*
using *coplanar-trans-1* **by** *auto*
qed
moreover have *Coplanar X Y A′ B′*
using *Col-cases Coplanar-def P2 assms*(*2*) **by** *auto*
moreover have *A A′ Perp X Y*
using *P2 Perp-perm* **by** *blast*
moreover have *B B′ Perp X Y*
using *P2 Perp-perm* **by** *blast*
ultimately show *?thesis*
using ‹*Coplanar X Y A B*› *l12-9* **by** *auto*
qed
qed
{
assume *A A′ ParStrict B B′*
then have *A A′ OS B B′*
by (*simp add*: *l12-6*)
have *A A′ TS PO B*
using *Col-cases* ‹$A \neq B$› *assms*(*1*) *assms*(*3*) *bet--ts* **by** *blast*
then have *A A′ TS B′ PO*
using ‹*A A′ OS B B′*› *l9-2 l9-8-2* **by** *blast*
then have *Bet PO A′ B′*
using *Col-cases assms*(*2*) *between-symmetry col-two-sides-bet invert-two-sides* **by** *blast*
}
thus *?thesis*
by (*metis Col-cases Par-def* ‹*A A′ Par B B′*› ‹$A \neq B$› *assms*(*1*) *assms*(*3*) *bet-col col-trivial-3 l6-21*)
qed
qed

lemma *perp2-preserves-bet13*:
assumes *Bet B PO C* **and**
Col PO B′ C′ **and**
¬ *Col PO B B′* **and**
PO Perp2 B C′ C B′
shows *Bet B′ PO C′*
proof *cases*
assume $C' = PO$
thus *?thesis*
using *not-bet-distincts* **by** *blast*
next
assume $C' \neq PO$
show *?thesis*
proof *cases*
assume $B' = PO$
thus *?thesis*
using *between-trivial2* **by** *auto*
next
assume $B' \neq PO$
have $B \neq PO$
using *assms*(*3*) *col-trivial-1* **by** *auto*
have *Col B PO C*
by (*simp add*: *Col-def assms*(*1*))

show *?thesis*
proof *cases*
assume $B = C$
thus *?thesis*
using ‹$B = C$› ‹$B \neq PO$› *assms*(*1*) *between-identity* **by** *blast*
next
assume $B \neq C$
have $B\ C'$ *Par* $C\ B'$
proof −
obtain $X\ Y$ **where** *P1*: *Col PO X Y* ∧ *X Y Perp B* C' ∧ *X Y Perp C* B'
using *Perp2-def assms*(*4*) **by** *auto*
have *Coplanar X Y B C*
by (*meson P1* ‹*Col B PO C*› *assms*(*1*) *l9-18-R2 ncop--ncols not-col-permutation-2 not-col-permutation-5 ts--coplanar*)
have *Coplanar X Y* $C'\ B'$
using *Col-cases Coplanar-def P1 assms*(*2*) **by** *auto*
show *?thesis*
proof −
have *Coplanar X Y B C*
by (*simp add*: ‹*Coplanar X Y B C*›)
moreover have *Coplanar X Y B* B'
by (*metis P1* ‹$C' \neq PO$› *assms*(*1*) *assms*(*2*) *bet-cop--cop calculation col-cop2--cop not-col-permutation-5 perp--coplanar*)
moreover have *Coplanar X Y* C' *C*
by (*smt P1* ‹$B \neq PO$› ‹*Col B PO C*› ‹*Coplanar X Y* $C'\ B'$› *assms*(*2*) *col2-cop--cop col-cop2--cop col-permutation-1 col-transitivity-2 coplanar-perm-1 perp--coplanar*)
moreover have *Coplanar X Y* $C'\ B'$
by (*simp add*: ‹*Coplanar X Y* $C'\ B'$›)
moreover have *B* C' *Perp X Y*
using *P1 Perp-perm* **by** *blast*
moreover have *C* B' *Perp X Y*
by (*simp add*: *P1 Perp-perm*)
ultimately show *?thesis*
using *l12-9* **by** *blast*
qed
qed
have *B* C' *ParStrict C* B'
by (*metis Out-def Par-def* ‹*B* C' *Par C* B'› ‹$B \neq C$› ‹$B \neq PO$› *assms*(*1*) *assms*(*3*) *col-transitivity-1 not-col-permutation-4 out-col*)
have $B' \neq PO$
by (*simp add*: ‹$B' \neq PO$›)
obtain $X\ Y$ **where** *P5*: *Col PO X Y* ∧ *X Y Perp B* C' ∧ *X Y Perp C* B'
using *Perp2-def assms*(*4*) **by** *auto*
have $X \neq Y$
using *P5 perp-not-eq-1* **by** *auto*
show *?thesis*
proof *cases*
assume *Col X Y B*
have *Col X Y C*
using *P5* ‹$B \neq PO$› ‹*Col B PO C*› ‹*Col X Y B*› *col-permutation-1 colx* **by** *blast*
show *?thesis*
proof −
have *Col* B' *PO* C'
using *Col-cases assms*(*2*) **by** *auto*
moreover have *Per PO C* B'
by (*metis P5* ‹*B* C' *ParStrict C* B'› ‹*Col X Y C*› *assms*(*2*) *col-permutation-2 par-strict-not-col-2 perp-col2 perp-per-2*)
moreover have *Per PO B* C'
using *P5* ‹$B \neq PO$› ‹*Col X Y B*› *col-permutation-1 perp-col2 perp-per-2* **by** *blast*
ultimately show *?thesis*
by (*metis Tarski-neutral-dimensionless.per13-preserves-bet-inv Tarski-neutral-dimensionless-axioms* ‹*B* C' *ParStrict C* B'› *assms*(*1*) *assms*(*3*) *between-symmetry not-col-distincts not-col-permutation-3 par-strict-not-col-2*)
qed
next
assume ¬ *Col X Y B*
then obtain *B0* **where** *U1*: *Col X Y B0* ∧ *X Y Perp B B0*

using *l8-18-existence* **by** *blast*
have ¬ *Col X Y C*
by (*smt P5* ‹*B C′ ParStrict C B′*› ‹*Col B PO C*› ‹¬ *Col X Y B*› *assms*(*2*) *col-permutation-2 colx par-strict-not-col-2*)
then obtain *C0* **where** *U2*: *Col X Y C0* ∧ *X Y Perp C C0*
using *l8-18-existence* **by** *blast*
have *B0* ≠ *PO*
by (*metis P5 Perp-perm* ‹*Col B PO C*› ‹*Col X Y B0* ∧ *X Y Perp B B0*› ‹¬ *Col X Y C*› *assms*(*3*) *col-permutation-2 col-permutation-3 col-perp2-ncol-col*)
{
assume *C0* = *PO*
then have *C PO Par C B′*
by (*metis P5 Par-def Perp-cases* ‹*Col X Y C0* ∧ *X Y Perp C C0*› ‹¬ *Col X Y C*› *col-perp2-ncol-col not-col-distincts not-col-permutation-3 perp-distinct*)
then have *False*
by (*metis* ‹*B C′ ParStrict C B′*› *assms*(*2*) *assms*(*3*) *col3 not-col-distincts par-id-2 par-strict-not-col-2*)
}
then have *C0* ≠ *PO* **by** *auto*
have *Bet B0 PO C0*
proof −
have *Bet B PO C*
by (*simp add*: *assms*(*1*))
moreover have *PO* ≠ *B0*
using ‹*B0* ≠ *PO*› **by** *auto*
moreover have *PO* ≠ *C0*
using ‹*C0* ≠ *PO*› **by** *auto*
moreover have *Col B0 PO C0*
using *U1 U2 P5* ‹*X* ≠ *Y*› *col3 not-col-permutation-2* **by** *blast*
moreover have *Per PO B0 B*
proof −
have *B0 PerpAt PO B0 B0 B*
proof *cases*
assume *X* = *B0*
have *B0 PO Perp B B0*
by (*metis P5 U1 calculation*(*2*) *col3 col-trivial-2 col-trivial-3 perp-col2*)
show *?thesis*
proof −
have *B0* ≠ *PO*
using *calculation*(*2*) **by** *auto*
moreover have *B0 Y Perp B B0*
using *U1* ‹*X* = *B0*› **by** *auto*
moreover have *Col B0 Y PO*
using *Col-perm P5* ‹*X* = *B0*› **by** *blast*
ultimately show *?thesis*
using ‹*B0 PO Perp B B0*› *perp-in-comm perp-perp-in* **by** *blast*
qed
next
assume *X* ≠ *B0*
have *X B0 Perp B B0*
using *U1* ‹*X* ≠ *B0*› *perp-col* **by** *blast*
have *B0 PO Perp B B0*
by (*metis P5 U1 calculation*(*2*) *not-col-permutation-2 perp-col2*)
then have *B0 PerpAt B0 PO B B0*
by (*simp add*: *perp-perp-in*)
thus *?thesis*
using *Perp-in-perm* **by** *blast*
qed
then show *?thesis*
by (*simp add*: *perp-in-per*)
qed
moreover have *Per PO C0 C*
proof −
have *C0 PO Perp C C0*
by (*metis P5 U2 calculation*(*3*) *col3 col-trivial-2 col-trivial-3 perp-col2*)
then have *C0 PerpAt PO C0 C0 C*
by (*simp add*: *perp-in-comm perp-perp-in*)

```
          thus ?thesis
            using perp-in-per-2 by auto
        qed
        ultimately show ?thesis
          using per13-preserves-bet by blast
      qed
      show ?thesis
      proof cases
        assume C' = B0
        have B' = C0
        proof -
          have ¬ Col C' PO C
             using P5 U1 ‹B0 ≠ PO› ‹C' = B0› ‹¬ Col X Y C› colx not-col-permutation-3 not-col-permutation-4 by
blast
          moreover have C ≠ C0
            using U2 ‹¬ Col X Y C› by auto
          moreover have Col C C0 B'
          proof -
            have Coplanar X Y C0 B'
            proof -
              have Col X Y C0
                by (simp add: U2)
              moreover have Col C0 B' C0
                by (simp add: col-trivial-3)
              ultimately show ?thesis
                using ncop--ncols by blast
            qed
            moreover have C C0 Perp X Y
              using Perp-perm U2 by blast
            moreover have C B' Perp X Y
              using P5 Perp-perm by blast
            ultimately show ?thesis
              using cop-perp2--col by auto
          qed
          ultimately show ?thesis
            by (metis Col-def ‹C' = B0› ‹Bet B0 PO C0› assms(2) colx)
        qed
        show ?thesis
          using Bet-cases ‹B' = C0› ‹C' = B0› ‹Bet B0 PO C0› by blast
      next
        assume C' ≠ B0
        then have B' ≠ C0
          by (metis P5 U1 U2 ‹C0 ≠ PO› assms(2) col-permutation-1 colx l8-18-uniqueness)
        have B C' Par B B0
        proof -
          have Coplanar X Y B B
            using ncop-distincts by auto
          moreover have Coplanar X Y B B0
            using U1 ncop--ncols by blast
          moreover have Coplanar X Y C' B
            using P5 ncoplanar-perm-1 perp--coplanar by blast
          moreover have Coplanar X Y C' B0
             using ‹¬ Col X Y B› calculation(2) calculation(3) col-permutation-1 coplanar-perm-12 coplanar-perm-18
coplanar-trans-1 by blast
          moreover have B C' Perp X Y
            using P5 Perp-perm by blast
          moreover have B B0 Perp X Y
            using Perp-perm U1 by blast
          ultimately show ?thesis
            using l12-9 by blast
        qed
        {
          assume B C' ParStrict B B0
          have Col B B0 C'
            by (simp add: ‹B C' Par B B0› par-id-3)
        }
```

```
then have Col B B0 C′
  using ‹B C′ Par B B0› par-id-3 by blast
have Col C C0 B′
proof −
  have Coplanar X Y C0 B′
    by (simp add: U2 col--coplanar)
  moreover have C C0 Perp X Y
    by (simp add: Perp-perm U2)
  moreover have C B′ Perp X Y
    using P5 Perp-perm by blast
  ultimately show ?thesis
    using cop-perp2--col by auto
qed
show ?thesis
proof −
  have Col B′ PO C′
    using assms(2) not-col-permutation-4 by blast
  moreover have Per PO C0 B′
  proof −
    have C0 PerpAt PO C0 C0 B′
    proof cases
      assume X = C0
      have C0 PO Perp C B′
      proof −
        have C0 ≠ PO
          by (simp add: ‹C0 ≠ PO›)
        moreover have C0 Y Perp C B′
          using P5 ‹X = C0› by auto
        moreover have Col C0 Y PO
          using Col-perm P5 ‹X = C0› by blast
        ultimately show ?thesis
          using perp-col by blast
      qed
      then have B′ C0 Perp C0 PO
        using Perp-perm ‹B′ ≠ C0› ‹Col C C0 B′› not-col-permutation-1 perp-col1 by blast
      then have C0 PerpAt C0 B′ PO C0
        using Perp-perm perp-perp-in by blast
      thus ?thesis
        using Perp-in-perm by blast
    next
      assume X ≠ C0
      then have X C0 Perp C B′
        using P5 U2 perp-col by blast
      have C0 PO Perp C B′
        using Col-cases P5 U2 ‹C0 ≠ PO› perp-col2 by blast
      then have B′ C0 Perp C0 PO
        using Perp-cases ‹B′ ≠ C0› ‹Col C C0 B′› col-permutation-2 perp-col by blast
      thus ?thesis
        using Perp-in-perm Perp-perm perp-perp-in by blast
    qed
    then show ?thesis
      using perp-in-per-2 by auto
  qed
  moreover have Per PO B0 C′
  proof −
    have B0 PerpAt PO B0 B0 C′
    proof −
      have Col C′ B B0
        using Col-cases ‹Col B B0 C′› by blast
      then have C′ B0 Perp X Y using perp-col P5 Perp-cases ‹C′ ≠ B0› by blast
      show ?thesis
      proof −
        have PO B0 Perp B0 C′
          by (smt P5 U1 ‹B0 ≠ PO› ‹C′ ≠ B0› ‹Col B B0 C′› col-trivial-2 not-col-permutation-2 perp-col4)
        then show ?thesis
          using Perp-in-cases Perp-perm perp-perp-in by blast
```

qed
qed
thus *?thesis*
by (*simp add*: *perp-in-per*)
qed
ultimately show *?thesis*
using ‹*B0* ≠ *PO*› ‹*C0* ≠ *PO*› ‹*Bet B0 PO C0*› *between-symmetry per13-preserves-bet-inv* **by** *blast*
qed
qed
qed
qed
qed
qed

lemma *is-image-perp-in*:
assumes $A \neq A'$ **and**
$X \neq Y$ **and**
A A' Reflect X Y
shows ∃ *P*. *P PerpAt A A' X Y*
by (*metis Perp-def Tarski-neutral-dimensionless.Perp-perm Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*2*) *assms*(*3*) *ex-sym1 l10-6-uniqueness*)

lemma *perp-inter-perp-in-n*:
assumes *A B Perp C D*
shows ∃ *P*. *Col A B P* ∧ *Col C D P* ∧ *P PerpAt A B C D*
by (*simp add*: *assms perp-inter-perp-in*)

lemma *perp2-perp-in*:
assumes *PO Perp2 A B C D* **and**
¬ *Col PO A B* **and**
¬ *Col PO C D*
shows ∃ *P Q*. *Col A B P* ∧ *Col C D Q* ∧ *Col PO P Q* ∧ *P PerpAt PO P A B* ∧ *Q PerpAt PO Q C D*
proof −
obtain *X Y* **where** *P1*: *Col PO X Y* ∧ *X Y Perp A B* ∧ *X Y Perp C D*
using *Perp2-def assms*(*1*) **by** *blast*
have *X* ≠ *Y*
using *P1 perp-not-eq-1* **by** *auto*
obtain *P* **where** *P2*: *Col X Y P* ∧ *Col A B P* ∧ *P PerpAt X Y A B*
using *P1 perp-inter-perp-in-n* **by** *blast*
obtain *Q* **where** *P3*: *Col X Y Q* ∧ *Col C D Q* ∧ *Q PerpAt X Y C D*
using *P1 perp-inter-perp-in-n* **by** *blast*
have *Col A B P*
using *P2* **by** *simp*
moreover have *Col C D Q*
using *P3* **by** *simp*
moreover have *Col PO P Q*
using *P2 P3 P1* ‹*X* ≠ *Y*› *col3 not-col-permutation-2* **by** *blast*
moreover have *P PerpAt PO P A B*
proof *cases*
assume *X* = *PO*
thus *?thesis*
by (*metis P2 assms*(*2*) *not-col-permutation-3 not-col-permutation-4 perp-in-col-perp-in perp-in-sym*)
next
assume *X* ≠ *PO*
then have *P PerpAt A B X PO*
by (*meson Col-cases P1 P2 perp-in-col-perp-in perp-in-sym*)
then have *P PerpAt A B PO X*
using *Perp-in-perm* **by** *blast*
then have *P PerpAt A B PO P*
by (*metis Col-cases assms*(*2*) *perp-in-col perp-in-col-perp-in*)
thus *?thesis*
by (*simp add*: *perp-in-sym*)
qed
moreover have *Q PerpAt PO Q C D*
by (*metis P1 P3* ‹*X* ≠ *Y*› *assms*(*3*) *col-trivial-2 colx not-col-permutation-3 not-col-permutation-4 perp-in-col-perp-in perp-in-right-comm perp-in-sym*)

ultimately show *?thesis*
by *blast*
qed

lemma *l13-8*:
assumes *U* ≠ *PO* **and**
V ≠ *PO* **and**
Col PO P Q **and**
Col PO U V **and**
Per P U PO **and**
Per Q V PO
shows *PO Out P Q* ⟷ *PO Out U V*
by (*smt Out-def assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *assms*(*6*) *l8-2 not-col-permutation-5 per23-preserves-bet per23-preserves-bet-inv per-distinct-1*)

lemma *perp-in-rewrite*:
assumes *P PerpAt A B C D*
shows *P PerpAt A P P C* ∨ *P PerpAt A P P D* ∨ *P PerpAt B P P C* ∨ *P PerpAt B P P D*
by (*metis assms per-perp-in perp-in-distinct perp-in-per-1 perp-in-per-3 perp-in-per-4*)

lemma *perp-out-acute*:
assumes *B Out A C'* **and**
A B Perp C C'
shows *Acute A B C*
proof −
have *A* ≠ *B*
using *assms*(*1*) *out-diff1* **by** *auto*
have *C'* ≠ *B*
using *Out-def assms*(*1*) **by** *auto*
then have *B C' Perp C C'*
by (*metis assms*(*1*) *assms*(*2*) *out-col perp-col perp-comm perp-right-comm*)
then have *Per C C' B*
using *Perp-cases perp-per-2* **by** *blast*
then have *Acute C' C B* ∧ *Acute C' B C*
by (*metis* ‹*C'* ≠ *B*› *assms*(*2*) *l11-43 perp-not-eq-2*)
have *C* ≠ *B*
using ‹*B C' Perp C C'*› *l8-14-1* **by** *auto*
show *?thesis*
proof −
have *B Out A C'*
by (*simp add*: *assms*(*1*))
moreover have *B Out C C*
by (*simp add*: ‹*C* ≠ *B*› *out-trivial*)
moreover have *Acute C' B C*
by (*simp add*: ‹*Acute C' C B* ∧ *Acute C' B C*›)
ultimately show *?thesis*
using *acute-out2--acute* **by** *auto*
qed
qed

lemma *perp-bet-obtuse*:
assumes *B* ≠ *C'* **and**
A B Perp C C' **and**
Bet A B C'
shows *Obtuse A B C*
proof −
have *Acute C' B C*
proof −
have *B Out C' C'*
using *assms*(*1*) *out-trivial* **by** *auto*
moreover have *Col A B C'*
by (*simp add*: *Col-def assms*(*3*))
then have *C' B Perp C C'*
using *Out-def assms*(*2*) *assms*(*3*) *bet-col1 calculation perp-col2* **by** *auto*
ultimately show *?thesis*
using *perp-out-acute* **by** *blast*

qed
thus *?thesis*
using *acute-bet--obtuse assms*(*2*) *assms*(*3*) *between-symmetry perp-not-eq-1* **by** *blast*
qed

end

3.12.2 Part 1: 2D

context *Tarski-2D*
begin

lemma *perp-in2--col*:
assumes *P PerpAt A B X Y* **and**
P PerpAt A′ B′ X Y
shows *Col A B A′*
using *cop4-perp-in2--col all-coplanar assms* **by** *blast*

lemma *perp2-trans*:
assumes *P Perp2 A B C D* **and**
P Perp2 C D E F
shows *P Perp2 A B E F*
proof −
obtain *X Y* **where** *P1*: *Col P X Y* ∧ *X Y Perp A B* ∧ *X Y Perp C D*
using *Perp2-def assms*(*1*) **by** *blast*
obtain *X′ Y′* **where** *P2*: *Col P X′ Y′* ∧ *X′ Y′ Perp C D* ∧ *X′ Y′ Perp E F*
using *Perp2-def assms*(*2*) **by** *blast*
{
assume *X Y Par X′ Y′*
then have *P3*: *X Y ParStrict X′ Y′* ∨ (*X* ≠ *Y* ∧ *X′* ≠ *Y′* ∧ *Col X X′ Y′* ∧ *Col Y X′ Y′*)
using *Par-def* **by** *blast*
{
assume *X Y ParStrict X′ Y′*
then have *P Perp2 A B E F*
using *P1 P2 par-not-col* **by** *auto*
}
{
assume *X* ≠ *Y* ∧ *X′* ≠ *Y′* ∧ *Col X X′ Y′* ∧ *Col Y X′ Y′*
then have *P Perp2 A B E F*
by (*meson P1 P2 Perp2-def col-permutation-1 perp-col2*)
}
then have *P Perp2 A B E F*
using *P3* ‹*X Y ParStrict X′ Y′* ⟹ *P Perp2 A B E F*› **by** *blast*
}
{
assume ¬ *X Y Par X′ Y′*
then have *P Perp2 A B E F*
using *P1 P2 l12-9-2D* **by** *blast*
}
thus *?thesis*
using ‹*X Y Par X′ Y′* ⟹ *P Perp2 A B E F*› **by** *blast*
qed

lemma *perp2-par*:
assumes *PO Perp2 A B C D*
shows *A B Par C D*
using *Perp2-def l12-9-2D Perp-perm assms* **by** *blast*

end

3.12.3 Part 2: length

context *Tarski-neutral-dimensionless*

begin

lemma *lg-exists*:

∃ *l*. (*QCong l* ∧ *l A B*)
using *QCong-def cong-pseudo-reflexivity* **by** *blast*

lemma *lg-cong*:
assumes *QCong l* **and**
l A B **and**
l C D
shows *Cong A B C D*
by (*metis QCong-def assms*(*1*) *assms*(*2*) *assms*(*3*) *cong-inner-transitivity*)

lemma *lg-cong-lg*:
assumes *QCong l* **and**
l A B **and**
Cong A B C D
shows *l C D*
by (*metis QCong-def assms*(*1*) *assms*(*2*) *assms*(*3*) *cong-transitivity*)

lemma *lg-sym*:
assumes *QCong l*
and *l A B*
shows *l B A*
using *assms*(*1*) *assms*(*2*) *cong-pseudo-reflexivity lg-cong-lg* **by** *blast*

lemma *ex-points-lg*:
assumes *QCong l*
shows ∃ *A B*. *l A B*
using *QCong-def assms cong-pseudo-reflexivity* **by** *fastforce*

lemma *is-len-cong*:
assumes *TarskiLen A B l* **and**
TarskiLen C D l
shows *Cong A B C D*
using *TarskiLen-def assms*(*1*) *assms*(*2*) *lg-cong* **by** *auto*

lemma *is-len-cong-is-len*:
assumes *TarskiLen A B l* **and**
Cong A B C D
shows *TarskiLen C D l*
using *TarskiLen-def assms*(*1*) *assms*(*2*) *lg-cong-lg* **by** *fastforce*

lemma *not-cong-is-len*:
assumes ¬ *Cong A B C D* **and**
TarskiLen A B l
shows ¬ *l C D*
using *TarskiLen-def assms*(*1*) *assms*(*2*) *lg-cong* **by** *auto*

lemma *not-cong-is-len1*:
assumes ¬ *Cong A B C D*
and *TarskiLen A B l*
shows ¬ *TarskiLen C D l*
using *assms*(*1*) *assms*(*2*) *is-len-cong* **by** *blast*

lemma *lg-null-instance*:
assumes *QCongNull l*
shows *l A A*
by (*metis QCongNull-def QCong-def assms cong-diff cong-trivial-identity*)

lemma *lg-null-trivial*:
assumes *QCong l*
and *l A A*
shows *QCongNull l*
using *QCongNull-def assms*(*1*) *assms*(*2*) **by** *auto*

lemma *lg-null-dec*:

shows *QCongNull l* ∨ ¬ *QCongNull l* by *simp*

lemma *ex-point-lg*:
assumes *QCong l*
shows ∃ *B*. *l A B*
by (*metis QCong-def assms not-cong-3412 segment-construction*)

lemma *ex-point-lg-out*:
assumes *A* ≠ *P* **and**
QCong l **and**
¬ *QCongNull l*
shows ∃ *B*. (*l A B* ∧ *A Out B P*)
proof −
obtain *X Y* **where** *P1*: ∀ *X0 Y0*. (*Cong X Y X0 Y0* ⟷ *l X0 Y0*)
using *QCong-def assms*(*2*) **by** *auto*
then have *l X Y*
using *cong-reflexivity* **by** *auto*
then have *X* ≠ *Y*
using *assms*(*2*) *assms*(*3*) *lg-null-trivial* **by** *auto*
then obtain *B* **where** *A Out P B* ∧ *Cong A B X Y*
using *assms*(*1*) *segment-construction-3* **by** *blast*
thus *?thesis*
using *Cong-perm Out-cases P1* **by** *blast*
qed

lemma *ex-point-lg-bet*:
assumes *QCong l*
shows ∃ *B*. (*l M B* ∧ *Bet A M B*)
proof −
obtain *X Y* **where** *P1*: ∀ *X0 Y0*. (*Cong X Y X0 Y0* ⟷ *l X0 Y0*)
using *QCong-def assms* **by** *auto*
then have *l X Y*
using *cong-reflexivity* **by** *blast*
obtain *B* **where** *Bet A M B* ∧ *Cong M B X Y*
using *segment-construction* **by** *blast*
thus *?thesis*
using *Cong-perm P1* **by** *blast*
qed

lemma *ex-points-lg-not-col*:
assumes *QCong l*
and ¬ *QCongNull l*
shows ∃ *A B*. (*l A B* ∧ ¬ *Col A B P*)
proof −
have ∃ *B*::′*p*. *A* ≠ *B*
using *another-point* **by** *blast*
then obtain *A*::′*p* **where** *P* ≠ *A*
by *metis*
then obtain *Q* **where** ¬ *Col P A Q*
using *not-col-exists* **by** *auto*
then have *A* ≠ *Q*
using *col-trivial-2* **by** *auto*
then obtain *B* **where** *l A B* ∧ *A Out B Q*
using *assms*(*1*) *assms*(*2*) *ex-point-lg-out* **by** *blast*
thus *?thesis*
by (*metis* ‹¬ *Col P A Q*› *col-transitivity-1 not-col-permutation-1 out-col out-diff1*)
qed

lemma *ex-eql*:
assumes ∃ *A B*. (*TarskiLen A B l1* ∧ *TarskiLen A B l2*)
shows *l1* = *l2*
proof −
obtain *A B* **where** *P1*: *TarskiLen A B l1* ∧ *TarskiLen A B l2*
using *assms* **by** *auto*
have ∀ *A0 B0*. (*l1 A0 B0* ⟶ *l2 A0 B0*)
by (*metis TarskiLen-def* ‹*TarskiLen A B l1* ∧ *TarskiLen A B l2*› *lg-cong lg-cong-lg*)

have ∀ *A0 B0*. (*l1 A0 B0* ⟷ *l2 A0 B0*)
proof −
have ∀ *A0 B0*. (*l1 A0 B0* ⟶ *l2 A0 B0*)
by (*metis TarskiLen-def* ‹*TarskiLen A B l1* ∧ *TarskiLen A B l2*› *lg-cong lg-cong-lg*)
moreover have ∀ *A0 B0*. (*l2 A0 B0* ⟶ *l1 A0 B0*)
by (*metis TarskiLen-def* ‹*TarskiLen A B l1* ∧ *TarskiLen A B l2*› *lg-cong lg-cong-lg*)
ultimately show *?thesis* **by** *blast*
qed
thus *?thesis* **by** *blast*
qed

lemma *all-eql*:
assumes *TarskiLen A B l1* **and**
TarskiLen A B l2
shows *l1* = *l2*
using *assms*(*1*) *assms*(*2*) *ex-eql* **by** *auto*

lemma *null-len*:
assumes *TarskiLen A A la* **and**
TarskiLen B B lb
shows *la* = *lb*
by (*metis TarskiLen-def all-eql assms*(*1*) *assms*(*2*) *lg-null-instance lg-null-trivial*)

lemma *eqL-equivalence*:
assumes *QCong la* **and**
QCong lb **and**
QCong lc
shows *la* = *la* ∧ (*la* = *lb* ⟶ *lb* = *la*) ∧ (*la* = *lb* ∧ *lb* = *lc* ⟶ *la* = *lc*)
by *simp*

lemma *ex-lg*:
∃ *l*. (*QCong l* ∧ *l A B*)
by (*simp add*: *lg-exists*)

lemma *lg-eql-lg*:
assumes *QCong l1* **and**
l1 = *l2*
shows *QCong l2*
using *assms*(*1*) *assms*(*2*) **by** *auto*

lemma *ex-eqL*:
assumes *QCong l1* **and**
QCong l2 **and**
∃ *A B*. (*l1 A B* ∧ *l2 A B*)
shows *l1* = *l2*
using *TarskiLen-def all-eql assms*(*1*) *assms*(*2*) *assms*(*3*) **by** *auto*

3.12.4 Part 3 : angles

lemma *ang-exists*:
assumes *A* ≠ *B* **and**
C ≠ *B*
shows ∃ *a*. (*QCongA a* ∧ *a A B C*)
proof −
have *A B C CongA A B C*
by (*simp add*: *assms*(*1*) *assms*(*2*) *conga-refl*)
thus *?thesis*
using *QCongA-def assms*(*1*) *assms*(*2*) **by** *auto*
qed

lemma *ex-points-eng*:
assumes *QCongA a*
shows ∃ *A B C*. (*a A B C*)
proof −
obtain *A B C* **where** *A* ≠ *B* ∧ *C* ≠ *B* ∧ (∀ *X Y Z*. (*A B C CongA X Y Z* ⟷ *a X Y Z*))
using *QCongA-def assms* **by** *auto*

thus *?thesis*
using *conga-pseudo-refl* **by** *blast*
qed

lemma *ang-conga*:
assumes *QCongA a* **and**
a A B C **and**
a A′ B′ C′
shows *A B C CongA A′ B′ C′*
proof −
obtain *A0 B0 C0* **where** $A0 \neq B0 \wedge C0 \neq B0 \wedge (\forall X Y Z. (A0\ B0\ C0\ CongA\ X\ Y\ Z \longleftrightarrow a\ X\ Y\ Z))$
using *QCongA-def assms*(*1*) **by** *auto*
thus *?thesis*
by (*meson assms*(*2*) *assms*(*3*) *not-conga not-conga-sym*)
qed

lemma *is-ang-conga*:
assumes *A B C Ang a* **and**
A′ B′ C′ Ang a
shows *A B C CongA A′ B′ C′*
using *Ang-def ang-conga assms*(*1*) *assms*(*2*) **by** *auto*

lemma *is-ang-conga-is-ang*:
assumes *A B C Ang a* **and**
A B C CongA A′ B′ C′
shows *A′ B′ C′ Ang a*
proof −
have *QCongA a*
using *Ang-def assms*(*1*) **by** *auto*
then obtain *A0 B0 C0* **where** $A0 \neq B0 \wedge C0 \neq B0 \wedge (\forall X Y Z. (A0\ B0\ C0\ CongA\ X\ Y\ Z \longleftrightarrow a\ X\ Y\ Z))$
using *QCongA-def* **by** *auto*
thus *?thesis*
by (*metis Ang-def assms*(*1*) *assms*(*2*) *not-conga*)
qed

lemma *not-conga-not-ang*:
assumes *QCongA a* **and**
¬ *A B C CongA A′ B′ C′* **and**
a A B C
shows ¬ *a A′ B′ C′*
using *ang-conga assms*(*1*) *assms*(*2*) *assms*(*3*) **by** *auto*

lemma *not-conga-is-ang*:
assumes ¬ *A B C CongA A′ B′ C′* **and**
A B C Ang a
shows ¬ *a A′ B′ C′*
using *Ang-def ang-conga assms*(*1*) *assms*(*2*) **by** *auto*

lemma *not-cong-is-ang1*:
assumes ¬ *A B C CongA A′ B′ C′* **and**
A B C Ang a
shows ¬ *A′ B′ C′ Ang a*
using *assms*(*1*) *assms*(*2*) *is-ang-conga* **by** *blast*

lemma *ex-eqa*:
assumes $\exists A B C.(A\ B\ C\ Ang\ a1 \wedge A\ B\ C\ Ang\ a2)$
shows *a1* = *a2*
proof −
obtain *A B C* **where** *P1*: $A\ B\ C\ Ang\ a1 \wedge A\ B\ C\ Ang\ a2$
using *assms* **by** *auto*
{
fix *x y z*
assume *a1 x y z*
then have *x y z Ang a1*
using *Ang-def assms* **by** *auto*
then have *x y z CongA A B C*

```
      using P1 not-cong-is-ang1 by blast
    then have x y z Ang a2
      using P1 is-ang-conga-is-ang not-conga-sym by blast
    then have a2 x y z
      using Ang-def assms by auto
  }
  {
    fix x y z
    assume a2 x y z
    then have x y z Ang a2
      using Ang-def assms by auto
    then have x y z CongA A B C
      using P1 not-cong-is-ang1 by blast
    then have x y z Ang a1
      using P1 is-ang-conga-is-ang not-conga-sym by blast
    then have a1 x y z
      using Ang-def assms by auto
  }
  then have ∀ x y z. (a1 x y z) ⟷ (a2 x y z)
    using ‹⋀z y x. a1 x y z ⟹ a2 x y z› by blast
  then have ⋀x y. (∀ z. (a1 x y) z = (a2 x y) z)
    by simp
  then have ⋀ x y. (a1 x y) = (a2 x y) using fun-eq-iff by auto
  thus ?thesis using fun-eq-iff by auto
qed

lemma all-eqa:
  assumes A B C Ang a1 and
    A B C Ang a2
  shows a1 = a2
  using assms(1) assms(2) ex-eqa by blast

lemma is-ang-distinct:
  assumes A B C Ang a
  shows A ≠ B ∧ C ≠ B
  using assms conga-diff1 conga-diff2 is-ang-conga by blast

lemma null-ang:
  assumes A B A Ang a1 and
    C D C Ang a2
  shows a1 = a2
  using all-eqa assms(1) assms(2) conga-trivial-1 is-ang-conga-is-ang is-ang-distinct by auto

lemma flat-ang:
  assumes Bet A B C and
    Bet A' B' C' and
    A B C Ang a1 and
    A' B' C' Ang a2
  shows a1 = a2
proof −
  have A B C Ang a2
  proof −
    have A' B' C' Ang a2
      by (simp add: assms(4))
    moreover have A' B' C' CongA A B C
      by (metis assms(1) assms(2) assms(3) calculation conga-line is-ang-distinct)
    ultimately show ?thesis
      using is-ang-conga-is-ang by blast
  qed
  then show ?thesis
    using assms(3) all-eqa by auto
qed

lemma ang-distinct:
  assumes QCongA a and
    a A B C
```

shows *A* ≠ *B* ∧ *C* ≠ *B*
proof −
have *A B C Ang a*
by (*simp add*: *Ang-def assms*(*1*) *assms*(*2*))
thus *?thesis*
using *is-ang-distinct* **by** *auto*
qed

lemma *ex-ang*:
assumes *B* ≠ *A* **and**
B ≠ *C*
shows ∃ *a*. (*QCongA a* ∧ *a A B C*)
using *ang-exists assms*(*1*) *assms*(*2*) **by** *auto*

lemma *anga-exists*:
assumes *A* ≠ *B* **and**
C ≠ *B* **and**
Acute A B C
shows ∃ *a*. (*QCongAAcute a* ∧ *a A B C*)
proof −
have *A B C CongA A B C*
by (*simp add*: *assms*(*1*) *assms*(*2*) *conga-refl*)
thus *?thesis*
using *assms*(*1*) *QCongAAcute-def assms*(*3*) **by** *blast*
qed

lemma *anga-is-ang*:
assumes *QCongAAcute a*
shows *QCongA a*
proof −
obtain *A0 B0 C0* **where** *P1*: *Acute A0 B0 C0* ∧ (∀ *X Y Z*.(*A0 B0 C0 CongA X Y Z* ⟷ *a X Y Z*))
using *QCongAAcute-def assms* **by** *auto*
thus *?thesis*
using *QCongA-def* **by** (*metis acute-distincts*)
qed

lemma *ex-points-anga*:
assumes *QCongAAcute a*
shows ∃ *A B C*. *a A B C*
by (*simp add*: *anga-is-ang assms ex-points-eng*)

lemma *anga-conga*:
assumes *QCongAAcute a* **and**
a A B C **and**
a A′ B′ C′
shows *A B C CongA A′ B′ C′*
by (*meson Tarski-neutral-dimensionless.ang-conga Tarski-neutral-dimensionless-axioms anga-is-ang assms*(*1*) *assms*(*2*) *assms*(*3*))

lemma *is-anga-to-is-ang*:
assumes *A B C AngAcute a*
shows *A B C Ang a*
using *AngAcute-def Ang-def anga-is-ang assms* **by** *auto*

lemma *is-anga-conga*:
assumes *A B C AngAcute a* **and**
A′ B′ C′ AngAcute a
shows *A B C CongA A′ B′ C′*
using *AngAcute-def anga-conga assms*(*1*) *assms*(*2*) **by** *auto*

lemma *is-anga-conga-is-anga*:
assumes *A B C AngAcute a* **and**
A B C CongA A′ B′ C′
shows *A′ B′ C′ AngAcute a*
using *Tarski-neutral-dimensionless.AngAcute-def Tarski-neutral-dimensionless.Ang-def Tarski-neutral-dimensionless.is-ang-conga-is-a*
Tarski-neutral-dimensionless-axioms assms(*1*) *assms*(*2*) *is-anga-to-is-ang* **by** *fastforce*

lemma *not-conga-is-anga*:
assumes ¬ $A\ B\ C\ CongA\ A'\ B'\ C'$ **and**
$A\ B\ C\ AngAcute\ a$
shows ¬ $a\ A'\ B'\ C'$
using *AngAcute-def anga-conga assms(1) assms(2)* **by** *auto*

lemma *not-cong-is-anga1*:
assumes ¬ $A\ B\ C\ CongA\ A'\ B'\ C'$ **and**
$A\ B\ C\ AngAcute\ a$
shows ¬ $A'\ B'\ C'\ AngAcute\ a$
using *assms(1) assms(2) is-anga-conga* **by** *auto*

lemma *ex-eqaa*:
assumes $\exists\ A\ B\ C.\ (A\ B\ C\ AngAcute\ a1 \wedge A\ B\ C\ AngAcute\ a2)$
shows $a1 = a2$
using *all-eqa assms is-anga-to-is-ang* **by** *blast*

lemma *all-eqaa*:
assumes $A\ B\ C\ AngAcute\ a1$ **and**
$A\ B\ C\ AngAcute\ a2$
shows $a1 = a2$
using *assms(1) assms(2) ex-eqaa* **by** *blast*

lemma *is-anga-distinct*:
assumes $A\ B\ C\ AngAcute\ a$
shows $A \neq B \wedge C \neq B$
using *assms is-ang-distinct is-anga-to-is-ang* **by** *blast*

lemma *null-anga*:
assumes $A\ B\ A\ AngAcute\ a1$ **and**
$C\ D\ C\ AngAcute\ a2$
shows $a1 = a2$
using *assms(1) assms(2) is-anga-to-is-ang null-ang* **by** *blast*

lemma *anga-distinct*:
assumes $QCongAAcute\ a$ **and**
$a\ A\ B\ C$
shows $A \neq B \wedge C \neq B$
using *ang-distinct anga-is-ang assms(1) assms(2)* **by** *blast*

lemma *out-is-len-eq*:
assumes $A\ Out\ B\ C$ **and**
$TarskiLen\ A\ B\ l$ **and**
$TarskiLen\ A\ C\ l$
shows $B = C$
using *Out-def assms(1) assms(2) assms(3) between-cong not-cong-is-len1* **by** *fastforce*

lemma *out-len-eq*:
assumes $QCong\ l$ **and**
$A\ Out\ B\ C$ **and**
$l\ A\ B$ **and**
$l\ A\ C$
shows $B = C$ **using** *out-is-len-eq*
using *TarskiLen-def assms(1) assms(2) assms(3) assms(4)* **by** *auto*

lemma *ex-anga*:
assumes $Acute\ A\ B\ C$
shows $\exists\ a.\ (QCongAAcute\ a \wedge a\ A\ B\ C)$
using *acute-distincts anga-exists assms* **by** *blast*

lemma *not-null-ang-ang*:
assumes $QCongAnNull\ a$
shows $QCongA\ a$
using *QCongAnNull-def assms* **by** *blast*

lemma *not-null-ang-def-equiv*:
QCongAnNull a ⟷ (*QCongA a* ∧ (∃ *A B C*. (*a A B C* ∧ ¬ *B Out A C*)))
proof −
{
assume *QCongAnNull a*
have *QCongA a* ∧ (∃ *A B C*. (*a A B C* ∧ ¬ *B Out A C*))
using *QCongAnNull-def* ‹*QCongAnNull a*› *ex-points-eng* **by** *fastforce*
}
{
assume *QCongA a* ∧ (∃ *A B C*. (*a A B C* ∧ ¬ *B Out A C*))
have *QCongAnNull a*
by (*metis Ang-def QCongAnNull-def Tarski-neutral-dimensionless.l11-21-a Tarski-neutral-dimensionless-axioms* ‹*QCongA a* ∧ (∃ *A B C*. *a A B C* ∧ ¬ *B Out A C*)› *not-conga-is-ang*)
}
thus *?thesis*
using ‹*QCongAnNull a* ⟹ *QCongA a* ∧ (∃ *A B C*. *a A B C* ∧ ¬ *B Out A C*)› **by** *blast*
qed

lemma *not-flat-ang-def-equiv*:
QCongAnFlat a ⟷ (*QCongA a* ∧ (∃ *A B C*. (*a A B C* ∧ ¬ *Bet A B C*)))
proof −
{
assume *QCongAnFlat a*
then have *QCongA a* ∧ (∃ *A B C*. (*a A B C* ∧ ¬ *Bet A B C*))
using *QCongAnFlat-def ex-points-eng* **by** *fastforce*
}
{
assume *QCongA a* ∧ (∃ *A B C*. (*a A B C* ∧ ¬ *Bet A B C*))
have *QCongAnFlat a*
proof −
obtain *pp* :: *'p* **and** *ppa* :: *'p* **and** *ppb* :: *'p* **where**
f1: *QCongA a* ∧ *a pp ppa ppb* ∧ ¬ *Bet pp ppa ppb*
using ‹*QCongA a* ∧ (∃ *A B C*. *a A B C* ∧ ¬ *Bet A B C*)› **by** *blast*
then have *f2*: ∀ *p pa pb*. *pp ppa ppb CongA pb pa p* ∨ ¬ *a pb pa p*
by (*metis* (*no-types*) *Ang-def Tarski-neutral-dimensionless.not-cong-is-ang1 Tarski-neutral-dimensionless-axioms*)
then have *f3*: ∀ *p pa pb*. (*Col pp ppa ppb* ∨ ¬ *a pb pa p*) ∨ ¬ *Bet pb pa p*
by (*metis* (*no-types*) *Col-def Tarski-neutral-dimensionless.ncol-conga-ncol Tarski-neutral-dimensionless-axioms*)
have *f4*: ∀ *p pa pb*. (¬ *Bet ppa ppb pp* ∨ ¬ *Bet pb pa p*) ∨ ¬ *a pb pa p*
using *f2 f1* **by** (*metis Col-def Tarski-neutral-dimensionless.l11-21-a Tarski-neutral-dimensionless-axioms not-bet-and-out not-out-bet*)
have *f5*: ∀ *p pa pb*. (¬ *Bet ppb pp ppa* ∨ ¬ *Bet pb pa p*) ∨ ¬ *a pb pa p*
using *f2 f1* **by** (*metis Col-def Tarski-neutral-dimensionless.l11-21-a Tarski-neutral-dimensionless-axioms not-bet-and-out not-out-bet*)
{ **assume** *Bet ppa ppb pp*
then have *?thesis*
using *f4 f1 QCongAnFlat-def* **by** *blast* }
moreover
{ **assume** *Bet ppb pp ppa*
then have *?thesis*
using *f5 f1 QCongAnFlat-def* **by** *blast* }
ultimately show *?thesis*
using *f3 f1 Col-def QCongAnFlat-def* **by** *blast*
qed
}
thus *?thesis*
using ‹*QCongAnFlat a* ⟹ *QCongA a* ∧ (∃ *A B C*. *a A B C* ∧ ¬ *Bet A B C*)› **by** *blast*
qed

lemma *ang-const*:
assumes *QCongA a* **and**
A ≠ *B*
shows ∃ *C*. *a A B C*
proof −
obtain *A0 B0 C0* **where** *A0* ≠ *B0* ∧ *C0* ≠ *B0* ∧ (∀ *X Y Z*. (*A0 B0 C0 CongA X Y Z* ⟶ *a X Y Z*))
by (*metis QCongA-def assms*(*1*))
then have (*A0 B0 C0 CongA A0 B0 C0*) ⟷ *a A0 B0 C0*

by (*simp add*: *conga-refl*)
then have *a A0 B0 C0*
using ‹*A0* ≠ *B0* ∧ *C0* ≠ *B0* ∧ (∀ *X Y Z*. *A0 B0 C0 CongA X Y Z* ⟶ *a X Y Z*)› *conga-refl* **by** *blast*
then show *?thesis*
using ‹*A0* ≠ *B0* ∧ *C0* ≠ *B0* ∧ (∀ *X Y Z*. *A0 B0 C0 CongA X Y Z* ⟶ *a X Y Z*)› *angle-construction-3 assms*(*2*)
by *blast*
qed

lemma *ang-sym*:
assumes *QCongA a* **and**
a A B C
shows *a C B A*
proof −
obtain *A0 B0 C0* **where** *A0* ≠ *B0* ∧ *C0* ≠ *B0* ∧ (∀ *X Y Z*. (*A0 B0 C0 CongA X Y Z* ⟶ *a X Y Z*))
by (*metis QCongA-def assms*(*1*))
then show *?thesis*
by (*metis Tarski-neutral-dimensionless.ang-conga Tarski-neutral-dimensionless-axioms assms*(*1*) *assms*(*2*) *conga-left-comm conga-refl not-conga-sym*)
qed

lemma *ang-not-null-lg*:
assumes *QCongA a* **and**
QCong l **and**
a A B C **and**
l A B
shows ¬ *QCongNull l*
by (*metis QCongNull-def TarskiLen-def ang-distinct assms*(*1*) *assms*(*3*) *assms*(*4*) *cong-reverse-identity not-cong-is-len*)

lemma *ang-distincts*:
assumes *QCongA a* **and**
a A B C
shows *A* ≠ *B* ∧ *C* ≠ *B*
using *ang-distinct assms*(*1*) *assms*(*2*) **by** *auto*

lemma *anga-sym*:
assumes *QCongAAcute a* **and**
a A B C
shows *a C B A*
by (*simp add*: *ang-sym anga-is-ang assms*(*1*) *assms*(*2*))

lemma *anga-not-null-lg*:
assumes *QCongAAcute a* **and**
QCong l **and**
a A B C **and**
l A B
shows ¬ *QCongNull l*
using *ang-not-null-lg anga-is-ang assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) **by** *blast*

lemma *anga-distincts*:
assumes *QCongAAcute a* **and**
a A B C
shows *A* ≠ *B* ∧ *C* ≠ *B*
using *anga-distinct assms*(*1*) *assms*(*2*) **by** *blast*

lemma *ang-const-o*:
assumes ¬ *Col A B P* **and**
QCongA a **and**
QCongAnNull a **and**
QCongAnFlat a
shows ∃ *C*. *a A B C* ∧ *A B OS C P*
proof −
obtain *A0 B0 C0* **where** *P1*: *A0* ≠ *B0* ∧ *C0* ≠ *B0* ∧ (∀ *X Y Z*. (*A0 B0 C0 CongA X Y Z* ⟶ *a X Y Z*))
by (*metis QCongA-def assms*(*2*))
then have *a A0 B0 C0*
by (*simp add*: *conga-refl*)
then have *T1*: *A0* ≠ *C0*

using *P1 Tarski-neutral-dimensionless.QCongAnNull-def Tarski-neutral-dimensionless-axioms assms(3) out-trivial*
by *fastforce*
have *A ≠ B*
using *assms(1) col-trivial-1* **by** *blast*
have *A0 ≠ B0 ∧ B0 ≠ C0*
using *P1* **by** *auto*
then obtain *C* **where** *P2: A0 B0 C0 CongA A B C ∧ (A B OS C P ∨ Col A B C)*
using *angle-construction-2 assms(1)* **by** *blast*
then have *a A B C*
by (*simp add: P1*)
have *P3: A B OS C P ∨ Col A B C*
using *P2* **by** *simp*
have *P4: ∀ A B C. (a A B C ⟶ ¬ B Out A C)*
using *assms(3)* **by** (*simp add: QCongAnNull-def*)
have *P5: ∀ A B C. (a A B C ⟶ ¬ Bet A B C)*
using *assms(4) QCongAnFlat-def* **by** *auto*
{
assume *Col A B C*
have *¬ B Out A C*
using *P4* **by** (*simp add: ‹a A B C›*)
have *¬ Bet A B C*
using *P5* **by** (*simp add: ‹a A B C›*)
then have *A B OS C P*
using *‹Col A B C› ‹¬ B Out A C› l6-4-2* **by** *blast*
then have *∃ C1. (a A B C1 ∧ A B OS C1 P)*
using *‹a A B C›* **by** *blast*
}
then have *∃ C1. (a A B C1 ∧ A B OS C1 P)*
using *P3 ‹a A B C›* **by** *blast*
then show *?thesis*
by *simp*
qed

lemma *anga-const*:
assumes *QCongAAcute a* **and**
A ≠ B
shows *∃ C. a A B C*
using *Tarski-neutral-dimensionless.ang-const Tarski-neutral-dimensionless-axioms anga-is-ang assms(1) assms(2)* **by**
fastforce

lemma *null-anga-null-angaP*:
QCongANullAcute a ⟷ IsNullAngaP a
proof −
have *QCongANullAcute a ⟶ IsNullAngaP a*
using *IsNullAngaP-def QCongANullAcute-def ex-points-anga* **by** *fastforce*
moreover have *IsNullAngaP a ⟶ QCongANullAcute a*
by (*metis IsNullAngaP-def QCongAnNull-def Tarski-neutral-dimensionless.QCongANullAcute-def Tarski-neutral-dimensionless-axio*
anga-is-ang not-null-ang-def-equiv)
ultimately show *?thesis*
by *blast*
qed

lemma *is-null-anga-out*:
assumes
a A B C **and**
QCongANullAcute a
shows *B Out A C*
using *QCongANullAcute-def assms(1) assms(2)* **by** *auto*

lemma *acute-not-bet*:
assumes *Acute A B C*
shows *¬ Bet A B C*
using *acute-col--out assms bet-col not-bet-and-out* **by** *blast*

lemma *anga-acute*:
assumes *QCongAAcute a* **and**

a A B C
shows *Acute A B C*
by (*smt Tarski-neutral-dimensionless.QCongAAcute-def Tarski-neutral-dimensionless-axioms acute-conga--acute assms*(*1*) *assms*(*2*))

lemma *not-null-not-col*:
assumes *QCongAAcute a* **and**
¬ *QCongANullAcute a* **and**
a A B C
shows ¬ *Col A B C*
proof −
have *Acute A B C*
using *anga-acute assms*(*1*) *assms*(*3*) **by** *blast*
then show *?thesis*
using *Tarski-neutral-dimensionless.IsNullAngaP-def Tarski-neutral-dimensionless-axioms acute-col--out assms*(*1*) *assms*(*2*) *assms*(*3*) *null-anga-null-angaP* **by** *blast*
qed

lemma *ang-cong-ang*:
assumes *QCongA a* **and**
a A B C **and**
A B C CongA A′ B′ C′
shows *a A′ B′ C′*
by (*metis QCongA-def assms*(*1*) *assms*(*2*) *assms*(*3*) *not-conga*)

lemma *is-null-ang-out*:
assumes
a A B C **and**
QCongANull a
shows *B Out A C*
proof −
have *a A B C* ⟶ *B Out A C*
using *QCongANull-def assms*(*2*) **by** *auto*
then show *?thesis*
by (*simp add*: *assms*(*1*))
qed

lemma *out-null-ang*:
assumes *QCongA a* **and**
a A B C **and**
B Out A C
shows *QCongANull a*
by (*metis QCongANull-def QCongAnNull-def assms*(*1*) *assms*(*2*) *assms*(*3*) *not-null-ang-def-equiv*)

lemma *bet-flat-ang*:
assumes *QCongA a* **and**
a A B C **and**
Bet A B C
shows *AngFlat a*
by (*metis AngFlat-def QCongAnFlat-def assms*(*1*) *assms*(*2*) *assms*(*3*) *not-flat-ang-def-equiv*)

lemma *out-null-anga*:
assumes *QCongAAcute a* **and**
a A B C **and**
B Out A C
shows *QCongANullAcute a*
using *IsNullAngaP-def assms*(*1*) *assms*(*2*) *assms*(*3*) *null-anga-null-angaP* **by** *auto*

lemma *anga-not-flat*:
assumes *QCongAAcute a*
shows *QCongAnFlat a*
by (*metis* (*no-types*, *lifting*) *Tarski-neutral-dimensionless.QCongAnFlat-def Tarski-neutral-dimensionless.anga-is-ang Tarski-neutral-dimensionless-axioms assms bet-col is-null-anga-out not-bet-and-out not-null-not-col*)

lemma *anga-const-o*:
assumes ¬ *Col A B P* **and**

¬ *QCongANullAcute a* **and**
QCongAAcute a
shows ∃ *C*. (*a A B C* ∧ *A B OS C P*)
proof −
have *QCongA a*
by (*simp add*: *anga-is-ang assms*(*3*))
moreover have *QCongAnNull a*
using *QCongANullAcute-def assms*(*2*) *assms*(*3*) *calculation not-null-ang-def-equiv* **by** *auto*
moreover have *QCongAnFlat a*
by (*simp add*: *anga-not-flat assms*(*3*))
ultimately show *?thesis*
by (*simp add*: *ang-const-o assms*(*1*))
qed

lemma *anga-conga-anga*:
assumes *QCongAAcute a* **and**
a A B C **and**
A B C CongA A′ B′ C′
shows *a A′ B′ C′*
using *ang-cong-ang anga-is-ang assms*(*1*) *assms*(*2*) *assms*(*3*) **by** *blast*

lemma *anga-out-anga*:
assumes *QCongAAcute a* **and**
a A B C **and**
B Out A A′ **and**
B Out C C′
shows *a A′ B C′*
proof −
have *A B C CongA A′ B C′*
by (*simp add*: *assms*(*3*) *assms*(*4*) *l6-6 out2--conga*)
thus *?thesis*
using *anga-conga-anga assms*(*1*) *assms*(*2*) **by** *blast*
qed

lemma *out-out-anga*:
assumes *QCongAAcute a* **and**
B Out A C **and**
B′ Out A′ C′ **and**
a A B C
shows *a A′ B′ C′*
proof −
have *A B C CongA A′ B′ C′*
by (*simp add*: *assms*(*2*) *assms*(*3*) *l11-21-b*)
thus *?thesis*
using *anga-conga-anga assms*(*1*) *assms*(*4*) **by** *blast*
qed

lemma *is-null-all*:
assumes *A* ≠ *B* **and**
QCongANullAcute a
shows *a A B A*
proof −
obtain *A0 B0 C0* **where** *Acute A0 B0 C0* ∧ (∀ *X Y Z*. (*A0 B0 C0 CongA X Y Z* ⟷ *a X Y Z*))
using *QCongAAcute-def QCongANullAcute-def assms*(*2*) **by** *auto*
then have *a A0 B0 C0*
using *acute-distincts conga-refl* **by** *blast*
thus *?thesis*
by (*smt QCongANullAcute-def assms*(*1*) *assms*(*2*) *out-out-anga out-trivial*)
qed

lemma *anga-col-out*:
assumes *QCongAAcute a* **and**
a A B C **and**
Col A B C
shows *B Out A C*
proof −

have *Acute A B C*
using *anga-acute assms*(*1*) *assms*(*2*) **by** *auto*
then have *P1*: *Bet A B C* ⟶ *B Out A C*
using *acute-not-bet* **by** *auto*
then have *Bet C A B* ⟶ *B Out A C*
using *assms*(*3*) *l6-4-2* **by** *auto*
thus *?thesis*
using *P1 assms*(*3*) *l6-4-2* **by** *blast*
qed

lemma *ang-not-lg-null*:
assumes *QCong la* **and**
QCong lc **and**
QCongA a **and**
la A B **and**
lc C B **and**
a A B C
shows ¬ *QCongNull la* ∧ ¬ *QCongNull lc*
by (*metis ang-not-null-lg ang-sym assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *assms*(*6*))

lemma *anga-not-lg-null*:
assumes
QCongAAcute a **and**
la A B **and**
lc C B **and**
a A B C
shows ¬ *QCongNull la* ∧ ¬ *QCongNull lc*
by (*metis QCongNull-def anga-not-null-lg anga-sym assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*))

lemma *anga-col-null*:
assumes *QCongAAcute a* **and**
a A B C **and**
Col A B C
shows *B Out A C* ∧ *QCongANullAcute a*
using *anga-col-out assms*(*1*) *assms*(*2*) *assms*(*3*) *out-null-anga* **by** *blast*

lemma *eqA-preserves-ang*:
assumes *QCongA a* **and**
a = *b*
shows *QCongA b*
using *assms*(*1*) *assms*(*2*) **by** *auto*

lemma *eqA-preserves-anga*:
assumes *QCongAAcute a* **and**

a = *b*
shows *QCongAAcute b*
using *assms*(*1*) *assms*(*2*) **by** *auto*

4 Some postulates of the parallels

lemma *euclid-5--original-euclid*:
assumes *Euclid5*
shows *EuclidSParallelPostulate*
proof −
{
fix *A B C D P Q R*
assume *P1*: *B C OS A D* ∧ *SAMS A B C B C D* ∧ *A B C B C D SumA P Q R* ∧ ¬ *Bet P Q R*
obtain *M* **where** *P2*: *M Midpoint B C*
using *midpoint-existence* **by** *auto*
obtain *D′* **where** *P3*: *C Midpoint D D′*
using *symmetric-point-construction* **by** *auto*
obtain *E* **where** *P4*: *M Midpoint D′ E*
using *symmetric-point-construction* **by** *auto*
have *P5*: *A* ≠ *B*

using *P1 os-distincts* **by** *blast*
have *P6*: $B \neq C$
using *P1 os-distincts* **by** *blast*
have *P7*: $C \neq D$
using *P1 os-distincts* **by** *blast*
have *P10*: $M \neq B$
using *P2 P6 is-midpoint-id* **by** *auto*
have *P11*: $M \neq C$
using *P2 P6 is-midpoint-id-2* **by** *auto*
have *P13*: $C \neq D'$
using *P3 P7 is-midpoint-id-2* **by** *blast*
have *P16*: ¬ *Col B C A*
using *one-side-not-col123 P1* **by** *blast*
have *B C OS D A*
using *P1 one-side-symmetry* **by** *blast*
then have *P17*: ¬ *Col B C D*
using *one-side-not-col123 P1* **by** *blast*
then have *P18*: ¬ *Col M C D*
using *P2 Col-perm P11 col-transitivity-2 midpoint-col* **by** *blast*
then have *P19*: ¬ *Col M C D′*
by (*metis P13 P3 Col-perm col-transitivity-2 midpoint-col*)
then have *P20*: ¬ *Col D′ C B*
by (*metis Col-perm P13 P17 P3 col-transitivity-2 midpoint-col*)
then have *P21*: ¬ *Col M C E*
by (*metis P19 P4 bet-col col2--eq col-permutation-4 midpoint-bet midpoint-distinct-2*)
have *P22*: *M C D′ CongA M B E* ∧ *M D′ C CongA M E B* **using** *P13 l11-49*
by (*metis Cong-cases P19 P2 P4 l11-51 l7-13-R1 l7-2 midpoint-cong not-col-distincts*)
have *P23*: *Cong C D′ B E*
using *P11 P2 P4 l7-13-R1 l7-2* **by** *blast*
have *P27*: *C B TS D D′*
by (*simp add*: *P13 P17 P3 bet--ts midpoint-bet not-col-permutation-4*)
have *P28*: *A InAngle C B E*
proof −
have *C B A LeA C B E*
proof −
have *A B C LeA B C D′*
proof −
have *Bet D C D′*
by (*simp add*: *P3 midpoint-bet*)
then show *?thesis* **using** *P1 P7 P13 sams-chara*
by (*metis sams-left-comm sams-sym*)
qed
moreover have *A B C CongA C B A*
using *P5 P6 conga-pseudo-refl* **by** *auto*
moreover have *B C D′ CongA C B E*
by (*metis CongA-def Mid-cases P2 P22 P4 P6 symmetry-preserves-conga*)
ultimately show *?thesis*
using *l11-30* **by** *blast*
qed
moreover have *C B OS E A*
proof −
have *C B TS E D′*
using *P2 P20 P4 l7-2 l9-2 mid-two-sides not-col-permutation-1* **by** *blast*
moreover have *C B TS A D′*
using *P27* ‹*B C OS D A*› *invert-two-sides l9-8-2* **by** *blast*
ultimately show *?thesis*
using *OS-def* **by** *blast*
qed
ultimately show *?thesis*
using *lea-in-angle* **by** *simp*
qed
obtain *A′* **where** *P30*: *Bet C A′ E* ∧ (*A′* = *B* ∨ *B Out A′ A*) **using** *P28 InAngle-def* **by** *auto*
{
assume *A′* = *B*
then have *Col D′ C B*
by (*metis Col-def P2 P21 P30 P6 col-transitivity-1 midpoint-col*)

```
    then have False
      by (simp add: P20)
    then have ∃ Y. B Out A Y ∧ C Out D Y by auto
  }
  {
    assume P31: B Out A' A
    have ∃ I. BetS D' C I ∧ BetS B A' I
    proof -
      have P32: BetS B M C
        using BetS-def Midpoint-def P10 P11 P2 by auto
      moreover have BetS E M D'
        using BetS-def Bet-cases P19 P21 P4 midpoint-bet not-col-distincts by fastforce
      moreover have BetS C A' E
      proof -
        have P32A: C ≠ A'
          using P16 P31 out-col by auto
        {
          assume A' = E
          then have P33: B Out A E
            using P31 l6-6 by blast
          then have A B C B C D SumA D' C D
          proof -
            have D' C B CongA A B C
            proof -
              have D' C M CongA E B M
                by (simp add: P22 conga-comm)
              moreover have C Out D' D'
                using P13 out-trivial by auto
              moreover have C Out B M
                using BetSEq Out-cases P32 bet-out-1 by blast
              moreover have B Out A E
                using P33 by auto
              moreover have B Out C M
                using BetSEq Out-def P32 by blast
              ultimately show ?thesis
                using l11-10 by blast
            qed
            moreover have D' C B B C D SumA D' C D
              by (simp add: P27 l9-2 ts--suma-1)
            moreover have B C D CongA B C D
              using P6 P7 conga-refl by auto
            moreover have D' C D CongA D' C D
              using P13 P7 conga-refl by presburger
            ultimately show ?thesis
              using conga3-suma--suma by blast
          qed
          then have D' C D CongA P Q R
            using P1 suma2--conga by auto
          then have Bet P Q R
            using Bet-cases P3 bet-conga--bet midpoint-bet by blast
          then have False using P1 by simp
        }
        then have A' ≠ E by auto
        then show ?thesis
          by (simp add: BetS-def P30 P32A)
      qed
      moreover have ¬ Col B C D'
        by (simp add: P20 not-col-permutation-3)
      moreover have Cong B M C M
        using Midpoint-def P2 not-cong-1243 by blast
      moreover have Cong E M D' M
        using Cong-perm Midpoint-def P4 by blast
      ultimately show ?thesis
        using euclid-5-def assms by blast
    qed
    then obtain Y where P34: Bet D' C Y ∧ BetS B A' Y using BetSEq by blast
```

```
      then have ∃ Y. B Out A Y ∧ C Out D Y
      proof −
        have P35: B Out A Y
          by (metis BetSEq Out-def P31 P34 l6-7)
        moreover have C Out D Y
        proof −
          have D ≠ C
            using P7 by auto
          moreover have Y ≠ C
            using P16 P35 l6-6 out-col by blast
          moreover have D′ ≠ C
            using P13 by auto
          moreover have Bet D C D′
            by (simp add: P3 midpoint-bet)
          moreover have Bet Y C D′
            by (simp add: Bet-perm P34)
          ultimately show ?thesis
            using l6-2 by blast
        qed
        ultimately show ?thesis by auto
      qed
    }
    then have ∃ Y. B Out A Y ∧ C Out D Y
      using P30 ‹A′ = B ⟹ ∃ Y. B Out A Y ∧ C Out D Y› by blast
  }
  then show ?thesis using euclid-s-parallel-postulate-def  by blast
qed

lemma tarski-s-euclid-implies-euclid-5:
  assumes TarskiSParallelPostulate
  shows Euclid5
proof −
  {
    fix P Q R S T U
    assume
      P1: BetS P T Q ∧ BetS R T S ∧ BetS Q U R ∧ ¬ Col P Q S ∧ Cong P T Q T ∧ Cong R T S T
    have P1A: BetS P T Q using P1 by simp
    have P1B: BetS R T S using P1 by simp
    have P1C: BetS Q U R using P1 by simp
    have P1D: ¬ Col P Q S using P1 by simp
    have P1E: Cong P T Q T using P1 by simp
    have P1F: Cong R T S T using P1 by simp
    obtain V where P2: P Midpoint R V
      using symmetric-point-construction by auto
    have P3: Bet V P R
      using Mid-cases P2 midpoint-bet by blast
    then obtain W where P4: Bet P W Q ∧ Bet U W V using inner-pasch
      using BetSEq P1C by blast
    {
      assume P = W
      have P ≠ V
           by (metis BetSEq Bet-perm Col-def Cong-perm Midpoint-def P1A P1B P1D P1E P1F P2 between-trivial
is-midpoint-id-2 l7-9)
      have Col P Q S
      proof −
        have f1: Col V P R
          by (meson Col-def P3)
        have f2: Col U R Q
          by (simp add: BetSEq Col-def P1)
        have f3: Bet P T Q
          using BetSEq P1 by fastforce
        have f4: R = P ∨ Col V P U
          by (metis (no-types) Col-def P4 ‹P = W› ‹P ≠ V› l6-16-1)
        have f5: Col Q P T
          using f3 by (meson Col-def)
        have f6: Col T Q P
```

```
      using f3 by (meson Col-def)
    have f7: Col P T Q
      using f3 by (meson Col-def)
    have f8: Col P Q P
      using Col-def P4 ‹P = W› by blast
    have Col R T S
      by (meson BetSEq Col-def P1)
    then have T = P ∨ Q = P
      using f8 f7 f6 f5 f4 f2 f1 by (metis (no-types) BetSEq P1 ‹P ≠ V› colx l6-16-1)
    then show ?thesis
      by (metis BetSEq P1)
  qed
  then have False
    by (simp add: P1D)
}
then have P5: P ≠ W by auto
have Bet V W U
  using Bet-cases P4 by auto
then obtain X Y where P7: Bet P V X ∧ Bet P U Y ∧ Bet X Q Y
  using assms(1) P1 P4 P5 tarski-s-parallel-postulate-def by blast
have Q S Par P R
proof –
  have Q ≠ S
    using P1D col-trivial-2 by auto
  moreover have T Midpoint Q P
    using BetSEq P1A P1E l7-2 midpoint-def not-cong-1243 by blast
  moreover have T Midpoint S R
    using BetSEq P1B P1F l7-2 midpoint-def not-cong-1243 by blast
  ultimately show ?thesis
    using l12-17 by auto
qed
then have P9: Q S ParStrict P R
  using P1D Par-def par-strict-symmetry par-symmetry by blast
have P10: Q S TS P Y
proof –
  have P10A: P ≠ R
    using P9 par-strict-distinct by auto
  then have P11: P ≠ X
    by (metis P2 P7 bet-neq12--neq midpoint-not-midpoint)
  have P12: ¬ Col X Q S
  proof –
    have Q S ParStrict P R
      by (simp add: P9)
    then have Col P R X
   by (metis P2 P3 P7 bet-col between-symmetry midpoint-not-midpoint not-col-permutation-4 outer-transitivity-between)
    then have P X ParStrict Q S
      using P9 Par-strict-perm P11 par-strict-col-par-strict by blast
    then show ?thesis
      using par-strict-not-col-2 by auto
  qed
  {
    assume W1: Col Y Q S
    have W2: Q = Y
      by (metis P12 P7 W1 bet-col bet-col1 colx)
    then have ¬ Col Q P R
      using P9 W1 par-not-col by auto
    then have W3: Q = U
      by (smt BetS-def Col-def P1C P7 W2 col-transitivity-2)
    then have False
      using BetS-def P1C by auto
  }
  then have ¬ Col Y Q S by auto
  then have Q S TS X Y
    by (metis P7 P12 bet--ts not-col-distincts not-col-permutation-1)
  moreover have Q S OS X P
  proof –
```

have *P* ≠ *V*
using *P10A P2 is-midpoint-id-2* **by** *blast*
then have *Q S ParStrict P X*
by (*meson Bet-perm P3 P7 P9 P11 bet-col not-col-permutation-4 par-strict-col-par-strict*)
then have *Q S ParStrict X P*
by (*simp add*: *par-strict-right-comm*)
then show *?thesis*
by (*simp add*: *l12-6*)
qed
ultimately show *?thesis*
using *l9-8-2* **by** *auto*
qed
then obtain *I* **where** *W4*: *Col I Q S* ∧ *Bet P I Y*
using *TS-def* **by** *blast*
have ∃ *I*. (*BetS S Q I* ∧ *BetS P U I*)
proof −
have *BetS P U I*
proof −
have *P* ≠ *Y*
using *P10 not-two-sides-id* **by** *auto*
have *W4A*: *Bet P U I*
proof −
have *W5*: *Col P U I*
using *P7 W4 bet-col1* **by** *auto*
{
assume *W6*: *Bet U I P*
have *W7*: *Q S OS P U*
proof −
have *Q S OS R U*
proof −
have ¬ *Col Q S R*
using *P9 par-strict-not-col-4* **by** *auto*
moreover have *Q Out R U*
using *BetSEq Out-def P1C* **by** *blast*
ultimately show *?thesis*
by (*simp add*: *out-one-side*)
qed
moreover have *Q S OS P R*
by (*simp add*: *P9 l12-6*)
ultimately show *?thesis*
using *one-side-transitivity* **by** *blast*
qed
have *W8*: *I Out P U* ∨ ¬ *Col Q S P*
by (*simp add*: *P1D not-col-permutation-1*)
have *False*
proof −
have *I Out U P*
using *W4 W6 W7 between-symmetry one-side-chara* **by** *blast*
then show *?thesis*
using *W6 not-bet-and-out* **by** *blast*
qed
}
{
assume *V1*: *Bet I P U*
have *P R OS I U*
proof −
have *P R OS I Q*
proof −
{
assume *Q* = *I*
then have *Col P Q S*
by (*metis BetSEq Col-def P1C P7 P9 V1 W4 between-equality outer-transitivity-between par-not-col*)
then have *False*
using *P1D* **by** *blast*
}
then have *Q* ≠ *I* **by** *blast*

```
        moreover have P R ParStrict Q S
          using P9 par-strict-symmetry by blast
        moreover have Col Q S I
          using Col-cases W4 by blast
        ultimately show ?thesis
          using one-side-symmetry par-strict-all-one-side by blast
      qed
      moreover have P R OS Q U
      proof −
        have Q S ParStrict P R
          using P9 by blast
        have R Out Q U ∧ ¬ Col P R Q
          by (metis BetSEq Bet-cases Out-def P1C calculation col124--nos)
        then show ?thesis
          by (metis P7 V1 W4 ‹Bet U I P ⟹ False› between-equality col-permutation-2 not-bet-distincts out-col
outer-transitivity-between)
      qed
      ultimately show ?thesis
        using one-side-transitivity by blast
    qed
    then have V2: P Out I U
      using P7 W4 bet2--out os-distincts by blast
    then have Col P I U
      using V1 not-bet-and-out by blast
    then have False
      using V1 V2 not-bet-and-out by blast
  }
  then moreover have ¬ (Bet U I P ∨ Bet I P U)
    using ‹Bet U I P ⟹ False› by auto
  ultimately show ?thesis
    using Col-def W5 by blast
qed
{
  assume P = U
  then have Col P R Q
    using BetSEq Col-def P1C by blast
  then have False
    using P9 par-strict-not-col-3 by blast
}
then have V6: P ≠ U by auto
{
  assume U = I
  have Q = U
  proof −
    have f1: BetS Q I R
      using P1C ‹U = I› by blast
    then have f2: Col Q I R
      using BetSEq Col-def by blast
    have f3: Col I R Q
      using f1 by (simp add: BetSEq Col-def)
    { assume R ≠ Q
      moreover
      { assume (R ≠ Q ∧ R ≠ I) ∧ ¬ Col I Q R
        moreover
        { assume ∃ p. (R ≠ Q ∧ ¬ Col I p I) ∧ Col Q I p
          then have I = Q
            using f1 by (metis (no-types) BetSEq Col-def col-transitivity-2) }
        ultimately have (∃ p pa. ((pa ≠ I ∧ ¬ Col pa p R) ∧ Col Q I pa) ∧ Col I pa p) ∨ I = Q
          using f3 f2 by (metis (no-types) col-transitivity-2) }
      ultimately have (∃ p pa. ((pa ≠ I ∧ ¬ Col pa p R) ∧ Col Q I pa) ∧ Col I pa p) ∨ I = Q
        using f1 by (metis (no-types) BetSEq P9 W4 col-transitivity-2 par-strict-not-col-4) }
    then show ?thesis
      using f2 by (metis P9 W4 ‹U = I› col-transitivity-2 par-strict-not-col-4)
  qed
  then have False
    using BetSEq P1C by blast
```

```
  }
  then have U ≠ I by auto
  then show ?thesis
    by (simp add: W4A V6 BetS-def)
qed
moreover have BetS S Q I
proof –
  have Q R TS S I
  proof –
    have Q R TS P I
    proof –
      have ¬ Col P Q R
        using P9 col-permutation-5 par-strict-not-col-3 by blast
      moreover have ¬ Col I Q R
      proof –
        {
          assume Col I Q R
          then have Col Q S R
          proof –
            have f1: ∀ p pa pb. Col p pa pb ∨ ¬ BetS pb p pa
              by (meson BetSEq Col-def)
            then have f2: Col U I P
              using ‹BetS P U I› by blast
            have f3: Col I P U
              by (simp add: BetSEq Col-def ‹BetS P U I›)
            have f4: ∀ p. (U = Q ∨ Col Q p R) ∨ ¬ Col Q U p
              by (metis BetSEq Col-def P1C col-transitivity-1)
            { assume P ≠ Q
              moreover
              { assume (P ≠ Q ∧ U ≠ Q) ∧ Col Q P Q
                then have (P ≠ Q ∧ U ≠ Q) ∧ ¬ Col Q P R
                  using Col-cases ‹¬ Col P Q R› by blast
                moreover
                { assume ∃ p. ((U ≠ Q ∧ P ≠ Q) ∧ ¬ Col Q p P) ∧ Col Q P p
                  then have U ≠ Q ∧ ¬ Col Q P P
                    by (metis col-transitivity-1)
                  then have ¬ Col U Q P
                    using col-transitivity-2 by blast }
                ultimately have ¬ Col U Q P ∨ I ≠ Q
                  using f4 f3 by blast }
              ultimately have I ≠ Q
                using f2 f1 by (metis BetSEq P1C col-transitivity-1 col-transitivity-2) }
            then have I ≠ Q
              using BetSEq ‹BetS P U I› by blast
            then show ?thesis
              by (simp add: W4 ‹Col I Q R› col-transitivity-2)
          qed
          then have False
            using P9 par-strict-not-col-4 by blast
        }
        then show ?thesis by blast
      qed
      moreover have Col U Q R
        using BetSEq Bet-cases Col-def P1C by blast
      moreover have Bet P U I
        by (simp add: BetSEq ‹BetS P U I›)
      ultimately show ?thesis
        using TS-def by blast
    qed
    moreover have Q R OS P S
    proof –
      have Q R Par P S
      proof –
        have Q ≠ R
          using BetSEq P1 by blast
        moreover have T Midpoint Q P
```

```
          using BetSEq Bet-cases P1A P1E cong-3421 midpoint-def by blast
        moreover have T Midpoint R S
          using BetSEq P1B P1F midpoint-def not-cong-1243 by blast
        ultimately show ?thesis
          using l12-17 by blast
      qed
      then have Q R ParStrict P S
        by (simp add: P1D Par-def not-col-permutation-4)
      then show ?thesis
        using l12-6 by blast
    qed
    ultimately show ?thesis
      using l9-8-2 by blast
  qed
  then show ?thesis
    by (metis BetS-def W4 col-two-sides-bet not-col-permutation-2 ts-distincts)
 qed
 ultimately show ?thesis
  by auto
qed
}
then show ?thesis using euclid-5-def by blast
qed

lemma tarski-s-implies-euclid-s-parallel-postulate:
 assumes TarskiSParallelPostulate
 shows EuclidSParallelPostulate
 by (simp add: assms euclid-5--original-euclid tarski-s-euclid-implies-euclid-5)

theorem tarski-s-euclid-implies-playfair-s-postulate:
 assumes TarskiSParallelPostulate
 shows PlayfairSPostulate
proof −
 {
  fix A1 A2 B1 B2 P C1 C2
  assume P1: ¬ Col P A1 A2 ∧ A1 A2 Par B1 B2 ∧ Col P B1 B2 ∧ A1 A2 Par C1 C2 ∧ Col P C1 C2
  have P1A: ¬ Col P A1 A2
   by (simp add: P1)
  have P2: A1 A2 Par B1 B2
   by (simp add: P1)
  have P3: Col P B1 B2
   by (simp add: P1)
  have P4: A1 A2 Par C1 C2
   by (simp add: P1)
  have P5: Col P C1 C2
   by (simp add: P1)
  have P6: A1 A2 ParStrict B1 B2
  proof −
   have A1 A2 Par B1 B2
    by (simp add: P1)
   moreover have Col B1 B2 P
    using P3 not-col-permutation-2 by blast
   moreover have ¬ Col A1 A2 P
    by (simp add: P1A not-col-permutation-1)
   ultimately show ?thesis
    using par-not-col-strict by auto
  qed
  have P7: A1 A2 ParStrict C1 C2
  proof −
   have A1 A2 Par C1 C2
    by (simp add: P1)
   moreover have Col C1 C2 P
    using Col-cases P1 by blast
   moreover have ¬ Col A1 A2 P
    by (simp add: P1A not-col-permutation-1)
   ultimately show ?thesis
```

using *par-not-col-strict* **by** *auto*
qed
{
assume ¬ *Col C1 B1 B2* ∨ ¬ *Col C2 B1 B2*
have ∃ *C′. Col C1 C2 C′* ∧ *B1 B2 TS A1 C′*
proof −
have *T2*: *Coplanar A1 A2 P A1*
using *ncop-distincts* **by** *auto*
have *T3*: *Coplanar A1 A2 B1 B2*
by (*simp add*: *P1 par--coplanar*)
have *T4*: *Coplanar A1 A2 C1 C2*
by (*simp add*: *P7 pars--coplanar*)
have *T5*: *Coplanar A1 A2 P B1*
using *P1 col-trivial-2 ncop-distincts par--coplanar par-col2-par-bis* **by** *blast*
then have *T6*: *Coplanar A1 A2 P B2*
using *P3 T3 col-cop--cop* **by** *blast*
have *T7*: *Coplanar A1 A2 P C1*
using *P1 T4 col-cop--cop coplanar-perm-1 not-col-permutation-2 par-distincts* **by** *blast*
then have *T8*: *Coplanar A1 A2 P C2*
using *P5 T4 col-cop--cop* **by** *blast*
{
assume ¬ *Col C1 B1 B2*
moreover have *C1* ≠ *C2*
using *P1 par-neq2* **by** *auto*
moreover have *Col B1 B2 P*
using *P1 not-col-permutation-2* **by** *blast*
moreover have *Col C1 C2 P*
using *Col-cases P5* **by** *auto*
moreover have ¬ *Col B1 B2 C1*
using *Col-cases calculation*(*1*) **by** *auto*
moreover have ¬ *Col B1 B2 A1*
using *P6 par-strict-not-col-3* **by** *auto*
moreover have *Coplanar B1 B2 C1 A1*
using *Col-cases P1A T5 T2 T6 T7 coplanar-pseudo-trans* **by** *blast*
ultimately have ∃ *C′. Col C1 C2 C′* ∧ *B1 B2 TS A1 C′*
using *cop-not-par-other-side* **by** *blast*
}
{
assume ¬ *Col C2 B1 B2*
moreover have *C2* ≠ *C1*
using *P1 par-neq2* **by** *blast*
moreover have *Col B1 B2 P*
using *Col-cases P3* **by** *auto*
moreover have *Col C2 C1 P*
using *Col-cases P5* **by** *auto*
moreover have ¬ *Col B1 B2 C2*
by (*simp add*: *calculation*(*1*) *not-col-permutation-1*)
moreover have ¬ *Col B1 B2 A1*
using *P6 par-strict-not-col-3* **by** *auto*
moreover have *Coplanar B1 B2 C2 A1*
using *Col-cases P1A T2 T5 T6 T8 coplanar-pseudo-trans* **by** *blast*
ultimately have ∃ *C′. Col C1 C2 C′* ∧ *B1 B2 TS A1 C′* **using** *cop-not-par-other-side*
by (*meson not-col-permutation-4*)
}
then show *?thesis*
using ‹¬ *Col C1 B1 B2* ⟹ ∃ *C′. Col C1 C2 C′* ∧ *B1 B2 TS A1 C′*› ‹¬ *Col C1 B1 B2* ∨ ¬ *Col C2 B1 B2*›
by *blast*
qed
then obtain *C′* **where** *W1*: *Col C1 C2 C′* ∧ *B1 B2 TS A1 C′* **by** *auto*
then have *W2*: ¬ *Col A1 B1 B2*
using *TS-def* **by** *blast*
obtain *B* **where** *W3*: *Col B B1 B2* ∧ *Bet A1 B C′*
using *TS-def W1* **by** *blast*
obtain *C* **where** *W4*: *P Midpoint C′ C*
using *symmetric-point-construction* **by** *blast*
then have *W4A*: *Bet A1 B C′* ∧ *Bet C P C′*

using *Mid-cases W3 midpoint-bet* **by** *blast*
then obtain *D* **where** *W5*: *Bet B D C* ∧ *Bet P D A1* **using** *inner-pasch* **by** *blast*
have *W6*: *C'* ≠ *P*
using *P3 TS-def W1* **by** *blast*
then have *A1 A2 Par C' P*
by (*meson P1 W1 not-col-permutation-2 par-col2-par*)
have *W9*: *A1 A2 ParStrict C' P*
using *Col-cases P5 P7 W1 W6 par-strict-col2-par-strict* **by** *blast*
then have *W10*: *B* ≠ *P*
by (*metis W6 W4A bet-out-1 out-col par-strict-not-col-3*)
have *W11*: *P* ≠ *C*
using *W6 W4 is-midpoint-id-2* **by** *blast*
{
assume *P* = *D*
then have *False*
by (*metis Col-def P3 W1 W3 W4A W5 W10 W11 col-trivial-2 colx l9-18-R1*)
}
then have *P* ≠ *D* **by** *auto*
then obtain *X Y* **where** *W12*: *Bet P B X* ∧ *Bet P C Y* ∧ *Bet X A1 Y*
using *W5 assms tarski-s-parallel-postulate-def* **by** *blast*
then have *P* ≠ *X*
using *W10 bet-neq12--neq* **by** *auto*
then have *A1 A2 ParStrict P X*
by (*metis Col-cases P3 P6 W10 W12 W3 bet-col colx par-strict-col2-par-strict*)
then have *W15*: *A1 A2 OS P X*
by (*simp add*: *l12-6*)
have *P* ≠ *Y*
using *W11 W12 between-identity* **by** *blast*
then have *A1 A2 ParStrict P Y*
by (*metis Col-def W11 W12 W4A W9 col-trivial-2 par-strict-col2-par-strict*)
then have *W16*: *A1 A2 OS P Y*
using *l12-6* **by** *auto*
have *Col A1 X Y*
by (*simp add*: *W12 bet-col col-permutation-4*)
then have *A1 Out X Y* **using** *col-one-side-out W15 W16*
using *one-side-symmetry one-side-transitivity* **by** *blast*
then have *False*
using *W12 not-bet-and-out* **by** *blast*
}
then have *Col C1 B1 B2* ∧ *Col C2 B1 B2*
by *auto*
}
{
fix *A1 A2 B1 B2 P C1 C2*
assume *P1*: *Col P A1 A2* ∧ *A1 A2 Par B1 B2* ∧ *Col P B1 B2* ∧ *A1 A2 Par C1 C2* ∧ *Col P C1 C2*
have *Col C1 B1 B2*
by (*smt P1 l9-10 not-col-permutation-3 not-strict-par2 par-col2-par par-comm par-id-5 par-symmetry ts-distincts*)
moreover have *Col C2 B1 B2*
by (*smt P1 l9-10 not-col-permutation-3 not-strict-par2 par-col2-par par-id-5 par-left-comm par-symmetry ts-distincts*)
ultimately have *Col C1 B1 B2* ∧ *Col C2 B1 B2* **by** *auto*
}
then show *?thesis*
using *playfair-s-postulate-def*
by (*metis* ‹⋀*P C2 C1 B2 B1 A2 A1*. ¬ *Col P A1 A2* ∧ *A1 A2 Par B1 B2* ∧ *Col P B1 B2* ∧ *A1 A2 Par C1 C2* ∧ *Col P C1 C2* ⟹ *Col C1 B1 B2* ∧ *Col C2 B1 B2*›)
qed

end
end

References

[1] M. Beeson, P. Boutry, G. Braun, C. Gries, and J. Narboux. GeoCoq, June 2018.

[2] M. Beeson, J. Narboux, and F. Wiedijk. Proof-checking euclid. *Annals of Mathematics and Artificial Intelligence*, 85(2):213–257, 2019.

[3] M. Beeson and L. Wos. Otter proofs in tarskian geometry. In *International Joint Conference on Automated Reasoning*, pages 495–510. Springer, 2014.

[4] M. Beeson and L. Wos. Finding proofs in tarskian geometry. *Journal of Automated Reasoning*, 58(1):181–207, 2017.

[5] P. Boutry. *On the formalization of foundations of geometry.* PhD thesis, Université de Strasbourg, 2018.

[6] P. Boutry, G. Braun, and J. Narboux. Formalization of the Arithmetization of Euclidean Plane Geometry and Applications. *Journal of Symbolic Computation*, 90:149–168, 2019.

[7] P. Boutry, C. Gries, J. Narboux, and P. Schreck. Parallel postulates and continuity axioms: a mechanized study in intuitionistic logic using Coq. *Journal of Automated Reasoning*, page 68, 2017. online first.

[8] R. Coghetto and A. Grabowski. Tarski geometry axioms - part II. *Formalized Mathematics*, 24(2):157–166, 2016.

[9] R. Coghetto and A. Grabowski. Tarski geometry axioms. part III. *Formalized Mathematics*, 25(4):289–313, 2017.

[10] R. Coghetto and A. Grabowski. Tarski geometry axioms. part IV - right angle. *Formalized Mathematics*, 27(1):75–85, 2019.

[11] M. Doré and K. Broda. Intuitive reasoning in formalized mathematics with elfe. In B. M. McLaren, R. Reilly, S. Zvacek, and J. Uhomoibhi, editors, *Computer Supported Education - 10th International Conference, CSEDU 2018, Funchal, Madeira, Portugal, March 15-17, 2018, Revised Selected Papers*, volume 1022 of *Communications in Computer and Information Science*, pages 549–571. Springer, 2018.

[12] C. Gries, P. Boutry, and J. Narboux. Somme des angles d'un triangle et unicité de la parallèle : une preuve d'équivalence formalisée en Coq. In *Les vingt-septièmes Journées Francophones des Langages Applicatifs (JFLA 2016)*, Actes des Vingt-septièmes Journées Francophones des Langages Applicatifs (JFLA 2016), page 15, Saint Malo, France, Jan. 2016. Jade Algave and Julien Signoles.

[13] T. J. M. Makarios. The independence of Tarski's Euclidean axiom. *Archive of Formal Proofs*, Oct. 2012. https://isa-afp.org/entries/Tarskis_Geometry.html, Formal proof development.

[14] G. E. Martin. *The foundations of geometry and the non-Euclidean plane.* Springer Science & Business Media, 2012.

[15] J. Narboux. Mechanical Theorem Proving in Tarski's geometry. In F. B. Eugenio Roanes Lozano, editor, *Automated Deduction in Geometry 2006*, volume 4869 of *LNCS*, pages 139–156, Pontevedra, Spain, Aug. 2006. Francisco Botana, Springer.

[16] J. Narboux, P. Janicic, and J. Fleuriot. Computer-assisted Theorem Proving in Synthetic Geometry. In M. Sitharam, A. S. John, and J. Sidman, editors, *Handbook of Geometric Constraint Systems Principles*, Discrete Mathematics and Its Applications. Chapman and Hall/CRC , 2018.

[17] J. Narboux, P. Janicic, and J. Fleuriot. Computer-assisted theorem proving in synthetic geometry. *Handbook of Geometric Constraint Systems Principles*, pages 25–73, 2018.

[18] T. Nipkow. Structured proofs in isar/hol. In *International Workshop on Types for Proofs and Programs*, pages 259–278. Springer, 2002.

[19] W. Richter, A. Grabowski, and J. Alama. Tarski geometry axioms. *Formalized Mathematics*, 22(2):167–176, 2014.

[20] W. Schwabhäuser, W. Szmielew, and A. Tarski. *Metamathematische Methoden in der Geometrie.* Springer-Verlag, Berlin, 1983.

[21] D. Simić, F. Marić, and P. Boutry. Poincaré disc model. *Archive of Formal Proofs*, Dec. 2019. https://isa-afp.org/entries/Poincare_Disc.html, Formal proof development.

[22] D. Simic, F. Maric, and P. Boutry. Formalization of the poincaré disc model of hyperbolic geometry. *J. Autom. Reasoning*, 64(4), 2020.

[23] S. Stojanović, V. Pavlović, and P. Janičić. A coherent logic based geometry theorem prover capable of producing formal and readable proofs. In *International Workshop on Automated Deduction in Geometry*, pages 201–220. Springer, 2010.

[24] G. Sutcliffe and C. Suttner. The tptp problem library. *Journal of Automated Reasoning*, 21(2):177–203, 1998.

[25] S. S. Đurđević, J. Narboux, and P. Janičić. Automated generation of machine verifiable and readable proofs: a case study of tarski's geometry. *Annals of Mathematics and Artificial Intelligence*, 74(3-4):249–269, 2015.

[26] F. Wiedijk. A synthesis of the procedural and declarative styles of interactive theorem proving. *Logical Methods in Computer Science*, 8, 2012.