Projective Geometry

Anthony Bordg

September 13, 2023

Abstract

We formalize the basics of projective geometry.
In particular, we give a proof of the so-called Hessenberg's theorem in projective plane geometry (see [1] for an alternative proof using a Coherent Logic prover in Prolog which generates Coq proof scripts).
We also provide a proof of the so-called Desargues's theorem based on an axiomatization [2] of (higher) projective space geometry using the notion of rank of a matroid. This last approach allows to handle incidence relations in an homogeneous way dealing only with points and without the need of talking explicitly about lines, planes or any higher entity.

Contents

theory *Projective-Plane-Axioms*
imports *Main*
begin

Contents:

- We introduce the types of points and lines and an incidence relation between them.
- A set of axioms for the projective plane (the models of these axioms are n-dimensional with $n \geq 2$).

1 The Axioms of the Projective Plane

locale *projective-plane* =

fixes *incid* :: *'point* ⇒ *'line* ⇒ *bool*

assumes *ax1*: $\exists\, l.\ incid\ P\ l \wedge incid\ Q\ l$

assumes *ax2*: $\exists\, P.\ incid\ P\ l \wedge incid\ P\ m$

assumes *ax-uniqueness*: $[\![incid\ P\ l;\ incid\ Q\ l;\ incid\ P\ m;\ incid\ Q\ m]\!] \Longrightarrow P = Q \vee l = m$

assumes *ax3*: $\exists\, A\ B\ C\ D.\ distinct\ [A,B,C,D] \wedge (\forall\, l.$
$(incid\ A\ l \wedge incid\ B\ l \longrightarrow \neg(incid\ C\ l) \wedge \neg(incid\ D\ l)) \wedge$
$(incid\ A\ l \wedge incid\ C\ l \longrightarrow \neg(incid\ B\ l) \wedge \neg(incid\ D\ l)) \wedge$
$(incid\ A\ l \wedge incid\ D\ l \longrightarrow \neg(incid\ B\ l) \wedge \neg(incid\ C\ l)) \wedge$
$(incid\ B\ l \wedge incid\ C\ l \longrightarrow \neg(incid\ A\ l) \wedge \neg(incid\ D\ l)) \wedge$
$(incid\ B\ l \wedge incid\ D\ l \longrightarrow \neg(incid\ A\ l) \wedge \neg(incid\ C\ l)) \wedge$
$(incid\ C\ l \wedge incid\ D\ l \longrightarrow \neg(incid\ A\ l) \wedge \neg(incid\ B\ l)))$

end
theory *Pappus-Property*
imports *Main Projective-Plane-Axioms*
begin

Contents:

- We give two formulations of Pappus's property for a configuration of nine points *is-pappus1* *is-pappus2*.
- We prove the equivalence of these two formulations *pappus-equiv*.
- We state Pappus property for a plane *is-pappus*.

2 Pappus's Property

context *projective-plane*
begin

definition *col* :: $['point, 'point, 'point] \Rightarrow bool$ **where**
$col\ A\ B\ C \equiv \exists l.\ incid\ A\ l \wedge incid\ B\ l \wedge incid\ C\ l$

lemma *distinct6-def*:
$distinct\ [A,B,C,D,E,F] \equiv (A \neq B) \wedge (A \neq C) \wedge (A \neq D) \wedge (A \neq E) \wedge (A \neq F)$
$\wedge$
$(B \neq C) \wedge (B \neq D) \wedge (B \neq E) \wedge (B \neq F) \wedge$
$(C \neq D) \wedge (C \neq E) \wedge (C \neq F) \wedge$
$(D \neq E) \wedge (D \neq F) \wedge$
$(E \neq F)$
by *auto*

definition *lines* :: $'point \Rightarrow 'point \Rightarrow 'line\ set$ **where**
$lines\ P\ Q \equiv \{l.\ incid\ P\ l \wedge incid\ Q\ l\}$

lemma *uniq-line*:
assumes $P \neq Q$ **and** $l \in lines\ P\ Q$ **and** $m \in lines\ P\ Q$
shows $l = m$
using *assms lines-def ax-uniqueness*
by *blast*

definition *line* :: $'point \Rightarrow 'point \Rightarrow 'line$ **where**
$line\ P\ Q \equiv @l.\ incid\ P\ l \wedge incid\ Q\ l$

definition *is-a-proper-intersec* :: $['point, 'point, 'point, 'point, 'point] \Rightarrow bool$ **where**
$is\text{-}a\text{-}proper\text{-}intersec\ P\ A\ B\ C\ D \equiv (A \neq B) \wedge (C \neq D) \wedge (line\ A\ B \neq line\ C\ D)$
$\wedge\ col\ P\ A\ B \wedge col\ P\ C\ D$

definition *is-pappus1* ::
$['point, 'point, 'point, 'point, 'point, 'point, 'point, 'point, 'point] => bool$ **where**
$is\text{-}pappus1\ A\ B\ C\ A'\ B'\ C'\ P\ Q\ R \equiv$
$distinct[A,B,C,A',B',C'] \longrightarrow col\ A\ B\ C \longrightarrow col\ A'\ B'\ C'$
$\longrightarrow is\text{-}a\text{-}proper\text{-}intersec\ P\ A\ B'\ A'\ B \longrightarrow is\text{-}a\text{-}proper\text{-}intersec\ Q\ B\ C'\ B'\ C$

$\longrightarrow$ *is-a-proper-intersec* $R\ A\ C'\ A'\ C$
$\longrightarrow$ *col* $P\ Q\ R$

definition *is-a-intersec* :: $['point, 'point, 'point, 'point, 'point] \Rightarrow bool$ **where**
is-a-intersec $P\ A\ B\ C\ D \equiv$ *col* $P\ A\ B \wedge$ *col* $P\ C\ D$

definition *is-pappus2* ::
$['point, 'point, 'point, 'point, 'point, 'point, 'point, 'point, 'point] \Rightarrow bool$ **where**
is-pappus2 $A\ B\ C\ A'\ B'\ C'\ P\ Q\ R \equiv$
(*distinct* $[A,B,C,A',B',C'] \vee (A \neq B' \wedge A' \neq B \wedge$ *line* $A\ B' \neq$ *line* $A'\ B \wedge$
$B \neq C' \wedge B' \neq C \wedge$ *line* $B\ C' \neq$ *line* $B'\ C \wedge$
$A \neq C' \wedge A' \neq C \wedge$ *line* $A\ C' \neq$ *line* $A'\ C$))
$\longrightarrow$ *col* $A\ B\ C \longrightarrow$ *col* $A'\ B'\ C' \longrightarrow$ *is-a-intersec* $P\ A\ B'\ A'\ B$
$\longrightarrow$ *is-a-intersec* $Q\ B\ C'\ B'\ C \longrightarrow$ *is-a-intersec* $R\ A\ C'\ A'\ C$
$\longrightarrow$ *col* $P\ Q\ R$

lemma *is-a-proper-intersec-is-a-intersec*:
assumes *is-a-proper-intersec* $P\ A\ B\ C\ D$
shows *is-a-intersec* $P\ A\ B\ C\ D$
using *assms is-a-intersec-def is-a-proper-intersec-def*
by *auto*

lemma *pappus21*:
assumes *is-pappus2* $A\ B\ C\ A'\ B'\ C'\ P\ Q\ R$
shows *is-pappus1* $A\ B\ C\ A'\ B'\ C'\ P\ Q\ R$
using *assms is-pappus2-def is-pappus1-def is-a-proper-intersec-is-a-intersec*
by *auto*

lemma *col-AAB*: *col* $A\ A\ B$
by (*simp add*: *ax1 col-def*)

lemma *col-ABA*: *col* $A\ B\ A$
using *ax1 col-def*
by *blast*

lemma *col-ABB*: *col* $A\ B\ B$
by (*simp add*: *ax1 col-def*)

lemma *incidA-lAB*: *incid* A (*line* $A\ B$)
by (*metis* (*no-types*, *lifting*) *ax1 line-def someI-ex*)

lemma *incidB-lAB*: *incid* B (*line* $A\ B$)
by (*metis* (*no-types*, *lifting*) *ax1 line-def someI-ex*)

lemma *degenerate-hexagon-is-pappus*:
assumes *distinct* $[A,B,C,A',B',C']$ **and** *col* $A\ B\ C$ **and** *col* $A'\ B'\ C'$ **and**
is-a-intersec $P\ A\ B'\ A'\ B$ **and** *is-a-intersec* $Q\ B\ C'\ B'\ C$ **and** *is-a-intersec* $R\ A$

$C'\ A'\ C$
and $\mathit{line}\ A\ B' = \mathit{line}\ A'\ B \lor \mathit{line}\ B\ C' = \mathit{line}\ B'\ C \lor \mathit{line}\ A\ C' = \mathit{line}\ A'\ C$
  shows $\mathit{col}\ P\ Q\ R$
proof −
  have $\mathit{col}\ P\ Q\ R$ **if** $\mathit{line}\ A\ B' = \mathit{line}\ A'\ B$
    by (*smt assms*(*1*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *assms*(*6*) *ax-uniqueness col-def distinct6-def*
      incidA-lAB incidB-lAB is-a-intersec-def that)
  have $\mathit{col}\ P\ Q\ R$ **if** $\mathit{line}\ B\ C' = \mathit{line}\ B'\ C$
    by (*smt* ‹$\mathit{line}\ A\ B' = \mathit{line}\ A'\ B \Longrightarrow \mathit{col}\ P\ Q\ R$› *assms*(*1*) *assms*(*2*) *assms*(*3*) *ax-uniqueness col-def*
      distinct6-def incidA-lAB incidB-lAB that)
  have $\mathit{col}\ P\ Q\ R$ **if** $\mathit{line}\ A\ C' = \mathit{line}\ A'\ C$
    by (*smt* ‹$\mathit{line}\ B\ C' = \mathit{line}\ B'\ C \Longrightarrow \mathit{col}\ P\ Q\ R$› *assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*7*) *ax-uniqueness*
      col-def distinct6-def incidA-lAB incidB-lAB)
  show $\mathit{col}\ P\ Q\ R$
    using ‹$\mathit{line}\ A\ B' = \mathit{line}\ A'\ B \Longrightarrow \mathit{col}\ P\ Q\ R$› ‹$\mathit{line}\ A\ C' = \mathit{line}\ A'\ C \Longrightarrow \mathit{col}\ P\ Q\ R$›
      ‹$\mathit{line}\ B\ C' = \mathit{line}\ B'\ C \Longrightarrow \mathit{col}\ P\ Q\ R$› *assms*(*7*)
    by *blast*
qed

lemma *pappus12*:
  assumes *is-pappus1* $A\ B\ C\ A'\ B'\ C'\ P\ Q\ R$
  shows *is-pappus2* $A\ B\ C\ A'\ B'\ C'\ P\ Q\ R$
proof −
  have $\mathit{col}\ P\ Q\ R$ **if** $(A \neq B' \land A' \neq B \land \mathit{line}\ A\ B' \neq \mathit{line}\ A'\ B \land$
  $B \neq C' \land B' \neq C \land \mathit{line}\ B\ C' \neq \mathit{line}\ B'\ C \land$
  $A \neq C' \land A' \neq C \land \mathit{line}\ A\ C' \neq \mathit{line}\ A'\ C)$ **and** *h1*:$\mathit{col}\ A\ B\ C$ **and** *h2*:$\mathit{col}\ A'\ B'\ C'$
  and *is-a-intersec* $P\ A\ B'\ A'\ B$ **and** *is-a-intersec* $Q\ B\ C'\ B'\ C$
  and *is-a-intersec* $R\ A\ C'\ A'\ C$
  proof −
    have $\mathit{col}\ P\ Q\ R$ **if** $A = B$
    proof −
      have $P = A$
        by (*metis* ‹$A \neq B' \land A' \neq B \land \mathit{line}\ A\ B' \neq \mathit{line}\ A'\ B \land B \neq C' \land B' \neq C \land \mathit{line}\ B\ C' \neq \mathit{line}\ B'\ C$
  $\land A \neq C' \land A' \neq C \land \mathit{line}\ A\ C' \neq \mathit{line}\ A'\ C$› ‹*is-a-intersec* $P\ A\ B'\ A'\ B$› *ax-uniqueness col-def*
          incidA-lAB incidB-lAB is-a-intersec-def that)
      have $\mathit{col}\ P\ A\ C' \land \mathit{col}\ Q\ A\ C' \land \mathit{col}\ R\ A\ C'$
        using ‹$P = A$› ‹*is-a-intersec* $Q\ B\ C'\ B'\ C$› ‹*is-a-intersec* $R\ A\ C'\ A'\ C$› *col-AAB*
          is-a-intersec-def that
        by *auto*
      then show $\mathit{col}\ P\ Q\ R$
        by (*smt* ‹$A \neq B' \land A' \neq B \land \mathit{line}\ A\ B' \neq \mathit{line}\ A'\ B \land B \neq C' \land B' \neq C$

$\wedge$ *line* $B\ C' \neq$ *line* $B'\ C$
$\wedge\ A \neq C' \wedge A' \neq C \wedge$ *line* $A\ C' \neq$ *line* $A'\ C$› *ax-uniqueness col-def*)
qed
have *col* $P\ Q\ R$ **if** $A = C$
proof −
have $R = A$
by (*metis* ‹$A \neq B' \wedge A' \neq B \wedge$ *line* $A\ B' \neq$ *line* $A'\ B \wedge B \neq C' \wedge B' \neq C$
$\wedge$ *line* $B\ C' \neq$ *line* $B'\ C$
$\wedge\ A \neq C' \wedge A' \neq C \wedge$ *line* $A\ C' \neq$ *line* $A'\ C$› ‹*is-a-intersec* $R\ A\ C'\ A'\ C$›
ax-uniqueness col-def incidA-lAB incidB-lAB is-a-intersec-def that)
have *col* $P\ A\ B' \wedge$ *col* $Q\ A\ B' \wedge$ *col* $R\ A\ B'$
using ‹$R = A$› ‹*is-a-intersec* $P\ A\ B'\ A'\ B$› ‹*is-a-intersec* $Q\ B\ C'\ B'\ C$› *col-def*

is-a-intersec-def that
by *auto*
then show *col* $P\ Q\ R$
by (*smt* ‹$A \neq B' \wedge A' \neq B \wedge$ *line* $A\ B' \neq$ *line* $A'\ B \wedge B \neq C' \wedge B' \neq C \wedge$
line $B\ C' \neq$ *line* $B'\ C$
$\wedge\ A \neq C' \wedge A' \neq C \wedge$ *line* $A\ C' \neq$ *line* $A'\ C$› *ax-uniqueness col-def*)
qed
have *col* $P\ Q\ R$ **if** $A = A'$
by (*smt* ‹$A \neq B' \wedge A' \neq B \wedge$ *line* $A\ B' \neq$ *line* $A'\ B \wedge B \neq C' \wedge B' \neq C \wedge$
line $B\ C' \neq$ *line* $B'\ C\ \wedge$
$A \neq C' \wedge A' \neq C \wedge$ *line* $A\ C' \neq$ *line* $A'\ C$› ‹*is-a-intersec* $P\ A\ B'\ A'\ B$›
‹*is-a-intersec* $R\ A\ C'\ A'\ C$›
ax-uniqueness col-ABA col-def incidA-lAB incidB-lAB is-a-intersec-def that)
have *col* $P\ Q\ R$ **if** $B = C$
by (*smt* ‹$A \neq B' \wedge A' \neq B \wedge$ *line* $A\ B' \neq$ *line* $A'\ B \wedge B \neq C' \wedge B' \neq C \wedge$
line $B\ C' \neq$ *line* $B'\ C\ \wedge$
$A \neq C' \wedge A' \neq C \wedge$ *line* $A\ C' \neq$ *line* $A'\ C$› ‹*is-a-intersec* $P\ A\ B'\ A'\ B$›
‹*is-a-intersec* $Q\ B\ C'\ B'\ C$›
‹*is-a-intersec* $R\ A\ C'\ A'\ C$› *ax-uniqueness col-def incidA-lAB incidB-lAB*
is-a-intersec-def that)
have *col* $P\ Q\ R$ **if** $B = B'$
by (*smt* ‹$A \neq B' \wedge A' \neq B \wedge$ *line* $A\ B' \neq$ *line* $A'\ B \wedge B \neq C' \wedge B' \neq C \wedge$
line $B\ C' \neq$ *line* $B'\ C\ \wedge$
$A \neq C' \wedge A' \neq C \wedge$ *line* $A\ C' \neq$ *line* $A'\ C$› ‹*is-a-intersec* $P\ A\ B'\ A'\ B$›
‹*is-a-intersec* $Q\ B\ C'\ B'\ C$›
ax-uniqueness col-AAB col-def incidA-lAB incidB-lAB is-a-intersec-def that)
have *col* $P\ Q\ R$ **if** $C = C'$
by (*smt* ‹$A \neq B' \wedge A' \neq B \wedge$ *line* $A\ B' \neq$ *line* $A'\ B \wedge B \neq C' \wedge B' \neq C \wedge$
line $B\ C' \neq$ *line* $B'\ C\ \wedge$
$A \neq C' \wedge A' \neq C \wedge$ *line* $A\ C' \neq$ *line* $A'\ C$› ‹*is-a-intersec* $Q\ B\ C'\ B'\ C$›
‹*is-a-intersec* $R\ A\ C'\ A'\ C$›
ax-uniqueness col-ABB col-def incidA-lAB incidB-lAB is-a-intersec-def that)
have *col* $P\ Q\ R$ **if** $A' = B'$
by (*smt* ‹$A \neq B' \wedge A' \neq B \wedge$ *line* $A\ B' \neq$ *line* $A'\ B \wedge B \neq C' \wedge B' \neq C \wedge$
line $B\ C' \neq$ *line* $B'\ C\ \wedge$
$A \neq C' \wedge A' \neq C \wedge$ *line* $A\ C' \neq$ *line* $A'\ C$› ‹*is-a-intersec* $P\ A\ B'\ A'\ B$›

‹*is-a-intersec Q B C′ B′ C*›
‹*is-a-intersec R A C′ A′ C*› *ax-uniqueness col-def incidA-lAB incidB-lAB*
is-a-intersec-def that)
have *col P Q R* **if** A′ = C′
by (*smt* ‹A ≠ B′ ∧ A′ ≠ B ∧ *line* A B′ ≠ *line* A′ B ∧ B ≠ C′ ∧ B′ ≠ C ∧
line B C′ ≠ *line* B′ C ∧
A ≠ C′ ∧ A′ ≠ C ∧ *line* A C′ ≠ *line* A′ C› ‹*is-a-intersec P A B′ A′ B*›
‹*is-a-intersec Q B C′ B′ C*›
‹*is-a-intersec R A C′ A′ C*› *ax-uniqueness col-def incidA-lAB incidB-lAB*
is-a-intersec-def that)
have *col P Q R* **if** B′ = C′
by (*smt* ‹A ≠ B′ ∧ A′ ≠ B ∧ *line* A B′ ≠ *line* A′ B ∧ B ≠ C′ ∧ B′ ≠ C ∧
line B C′ ≠ *line* B′ C ∧
A ≠ C′ ∧ A′ ≠ C ∧ *line* A C′ ≠ *line* A′ C› ‹*is-a-intersec P A B′ A′ B*›
‹*is-a-intersec Q B C′ B′ C*›
‹*is-a-intersec R A C′ A′ C*› *ax-uniqueness col-def incidA-lAB incidB-lAB*
is-a-intersec-def that)
have *col P Q R* **if** A ≠ B ∧ A ≠ C ∧ A ≠ A′ ∧ B ≠ C ∧ B ≠ B′ ∧ C ≠ C′ ∧
A′≠ B′
∧ A′ ≠ C′ ∧ B′ ≠ C′
proof −
have *a1*:*distinct* [A,B,C,A′,B′,C′]
using ‹A ≠ B′ ∧ A′ ≠ B ∧ *line* A B′ ≠ *line* A′ B ∧ B ≠ C′ ∧ B′ ≠ C ∧
line B C′ ≠ *line* B′ C ∧
A ≠ C′ ∧ A′ ≠ C ∧ *line* A C′ ≠ *line* A′ C› *distinct6-def that*
by *auto*
have *is-a-proper-intersec P A B′ A′ B*
using ‹A ≠ B′ ∧ A′ ≠ B ∧ *line* A B′ ≠ *line* A′ B ∧ B ≠ C′ ∧ B′ ≠ C ∧
line B C′ ≠ *line* B′ C ∧
A ≠ C′ ∧ A′ ≠ C ∧ *line* A C′ ≠ *line* A′ C› ‹*is-a-intersec P A B′ A′ B*›
is-a-intersec-def
is-a-proper-intersec-def
by *auto*
have *is-a-proper-intersec Q B C′ B′ C*
using ‹A ≠ B′ ∧ A′ ≠ B ∧ *line* A B′ ≠ *line* A′ B ∧ B ≠ C′ ∧ B′ ≠ C ∧
line B C′ ≠ *line* B′ C ∧
A ≠ C′ ∧ A′ ≠ C ∧ *line* A C′ ≠ *line* A′ C› ‹*is-a-intersec Q B C′ B′ C*›
is-a-intersec-def
is-a-proper-intersec-def
by *auto*
have *is-a-proper-intersec R A C′ A′ C*
using ‹A ≠ B′ ∧ A′ ≠ B ∧ *line* A B′ ≠ *line* A′ B ∧ B ≠ C′ ∧ B′ ≠ C ∧
line B C′ ≠ *line* B′ C ∧
A ≠ C′ ∧ A′ ≠ C ∧ *line* A C′ ≠ *line* A′ C› ‹*is-a-intersec R A C′ A′ C*›
is-a-intersec-def
is-a-proper-intersec-def
by *auto*
show *col P Q R*
using ‹*is-a-proper-intersec P A B′ A′ B*› ‹*is-a-proper-intersec Q B C′ B′ C*›

```
      ‹is-a-proper-intersec R A C′ A′ C› a1 assms h1 h2 is-pappus1-def
    by blast
  qed
    show col P Q R
      using ‹A = A′ ⟹ col P Q R› ‹A = B ⟹ col P Q R› ‹A = C ⟹ col P Q
R›
        ‹A ≠ B ∧ A ≠ C ∧ A ≠ A′ ∧ B ≠ C ∧ B ≠ B′ ∧ C ≠ C′ ∧ A′ ≠ B′ ∧
A′ ≠ C′ ∧ B′ ≠ C′ ⟹ col P Q R›
        ‹A′ = B′ ⟹ col P Q R› ‹A′ = C′ ⟹ col P Q R› ‹B = B′ ⟹ col P Q
R› ‹B = C ⟹ col P Q R›
        ‹B′ = C′ ⟹ col P Q R› ‹C = C′ ⟹ col P Q R›
      by blast
  qed
  have col P Q R if distinct [A,B,C,A′,B′,C′] and col A B C and col A′ B′ C′
    and is-a-intersec P A B′ A′ B and is-a-intersec Q B C′ B′ C and is-a-intersec
R A C′ A′ C
  proof –
    have col P Q R if line A B′ = line A′ B
      using ‹col A B C› ‹col A′ B′ C′› ‹distinct [A,B,C,A′,B′,C′]› ‹is-a-intersec P
A B′ A′ B›
      ‹is-a-intersec Q B C′ B′ C› ‹is-a-intersec R A C′ A′ C› degenerate-hexagon-is-pappus
that
      by blast
    have col P Q R if line B C′ = line B′ C
      using ‹col A B C› ‹col A′ B′ C′› ‹distinct [A,B,C,A′,B′,C′]› ‹is-a-intersec P
A B′ A′ B›
      ‹is-a-intersec Q B C′ B′ C› ‹is-a-intersec R A C′ A′ C› degenerate-hexagon-is-pappus
that
      by blast
    have col P Q R if line A′ C = line A C′
      using ‹col A B C› ‹col A′ B′ C′› ‹distinct [A,B,C,A′,B′,C′]› ‹is-a-intersec P
A B′ A′ B›
      ‹is-a-intersec Q B C′ B′ C› ‹is-a-intersec R A C′ A′ C› degenerate-hexagon-is-pappus
that
      by auto
    have col P Q R if line A B′ ≠ line A′ B and line B C′ ≠ line B′ C and
      line A C′ ≠ line A′ C
    proof –
      have is-a-proper-intersec P A B′ A′ B
        using ‹distinct [A,B,C,A′,B′,C′]› ‹is-a-intersec P A B′ A′ B› distinct6-def
is-a-intersec-def
          is-a-proper-intersec-def that(1)
        by auto
      have is-a-proper-intersec Q B C′ B′ C
        using ‹distinct [A,B,C,A′,B′,C′]› ‹is-a-intersec Q B C′ B′ C› distinct6-def
is-a-intersec-def
          is-a-proper-intersec-def that(2)
        by auto
      have is-a-proper-intersec R A C′ A′ C
```

using ‹*distinct* $[A,B,C,A',B',C']$› ‹*is-a-intersec* R A C' A' C› *distinct6-def* *is-a-intersec-def*
is-a-proper-intersec-def *that*(*3*)
by *auto*
show *col* P Q R
using ‹*col* A B C› ‹*col* A' B' C'› ‹*distinct* $[A,B,C,A',B',C']$› ‹*is-a-proper-intersec* P A B' A' B›
‹*is-a-proper-intersec* Q B C' B' C› ‹*is-a-proper-intersec* R A C' A' C› *assms* *is-pappus1-def*
by *blast*
qed
show *col* P Q R
using ‹$[\![$*line* A B' $\neq$ *line* A' B; *line* B C' $\neq$ *line* B' C; *line* A C' $\neq$ *line* A' $C]\!] \Longrightarrow$ *col* P Q R›
‹*line* A B' $=$ *line* A' B $\Longrightarrow$ *col* P Q R› ‹*line* A' C $=$ *line* A C' $\Longrightarrow$ *col* P Q R›
‹*line* B C' $=$ *line* B' C $\Longrightarrow$ *col* P Q R›
by *fastforce*
qed
show *is-pappus2* A B C A' B' C' P Q R
by (*simp add*: ‹$[\![A \neq B' \wedge A' \neq B \wedge$ *line* A B' $\neq$ *line* A' B $\wedge$ $B \neq C' \wedge B' \neq C \wedge$ *line* B C' $\neq$ *line* B' C
$\wedge$ $A \neq C' \wedge A' \neq C \wedge$ *line* A C' $\neq$ *line* A' C; *col* A B C; *col* A' B' C'; *is-a-intersec* P A B' A' B; *is-a-intersec* Q B C' B' C; *is-a-intersec* R A C' A' $C]\!]$ $\Longrightarrow$ *col* P Q R›
‹$[\![$*distinct* $[A,B,C,A',B',C']$; *col* A B C; *col* A' B' C'; *is-a-intersec* P A B' A' B; *is-a-intersec* Q B C' B' C; *is-a-intersec* R A C' A' $C]\!] \Longrightarrow$ *col* P Q R›
is-pappus2-def)
qed

lemma *pappus-equiv*: *is-pappus1* A B C A' B' C' P Q R $=$ *is-pappus2* A B C A' B' C' P Q R
using *pappus12* *pappus21*
by *blast*

definition *is-pappus* :: *bool* **where**
is-pappus $\equiv \forall$ A B C D E F P Q R. *is-pappus2* A B C D E F P Q R

end

end
theory *Pascal-Property*
imports *Main* *Projective-Plane-Axioms* *Pappus-Property*
begin

Contents:

- A hexagon is pascal if its three opposite sides meet in collinear points *is-pascal*.

- A plane is pascal, or has Pascal's property, if for every hexagon of that plane Pascal property is stable under any permutation of that hexagon.

3 Pascal's Property

context *projective-plane*
begin

definition *inters* :: *'line* ⇒ *'line* ⇒ *'point set* **where**
inters l m ≡ {*P*. *incid P l* ∧ *incid P m*}

lemma *inters-is-singleton*:
  assumes *l* ≠ *m* **and** *P* ∈ *inters l m* **and** *Q* ∈ *inters l m*
  shows *P* = *Q*
  using *assms ax-uniqueness inters-def*
  by *blast*

definition *inter* :: *'line* ⇒ *'line* ⇒ *'point* **where**
inter l m ≡ @*P*. *P* ∈ *inters l m*

lemma *uniq-inter*:
  assumes *l* ≠ *m* **and** *incid P l* **and** *incid P m*
  shows *inter l m* = *P*
proof −
  have *P* ∈ *inters l m*
    by (*simp add*: *assms*(*2*) *assms*(*3*) *inters-def*)
  have ∀ *Q*. *Q* ∈ *inters l m* ⟶ *Q* = *P*
    using ‹*P* ∈ *inters l m*› *assms*(*1*) *inters-is-singleton*
    by *blast*
  show *inter l m* = *P*
    using ‹*P* ∈ *inters l m*› *assms*(*1*) *inter-def inters-is-singleton*
    by *auto*
qed

definition *is-pascal* :: [*'point*, *'point*, *'point*, *'point*, *'point*, *'point*] ⇒ *bool* **where**
is-pascal A B C D E F ≡ *distinct* [*A*,*B*,*C*,*D*,*E*,*F*] ⟶ *line B C* ≠ *line E F* ⟶ *line C D* ≠ *line A F*
⟶ *line A B* ≠ *line D E* ⟶
(*let P* = *inter* (*line B C*) (*line E F*) *in*
let Q = *inter* (*line C D*) (*line A F*) *in*
let R = *inter* (*line A B*) (*line D E*) *in*
col P Q R)

lemma *col-rot-CW*:
assumes *col P Q R*
shows *col R P Q*
using *assms col-def*
by *auto*

lemma *col-2cycle*:
assumes *col P Q R*
shows *col P R Q*
using *assms col-def*
by *auto*

lemma *distinct6-rot-CW*:
assumes *distinct* [*A*,*B*,*C*,*D*,*E*,*F*]
shows *distinct* [*F*,*A*,*B*,*C*,*D*,*E*]
using *assms distinct6-def*
by *auto*

lemma *lines-comm*: *lines P Q* = *lines Q P*
using *lines-def*
by *auto*

lemma *line-comm*:
assumes $P \neq Q$
shows *line P Q* = *line Q P*
by (*metis ax-uniqueness incidA-lAB incidB-lAB*)

lemma *inters-comm*: *inters l m* = *inters m l*
using *inters-def*
by *auto*

lemma *inter-comm*: *inter l m* = *inter m l*
by (*simp add*: *inter-def inters-comm*)

lemma *inter-line-line-comm*:
assumes $C \neq D$
shows *inter* (*line A B*) (*line C D*) = *inter* (*line A B*) (*line D C*)
using *assms line-comm*
by *auto*

lemma *inter-line-comm-line*:
assumes $A \neq B$
shows *inter* (*line A B*) (*line C D*) = *inter* (*line B A*) (*line C D*)
using *assms line-comm*
by *auto*

lemma *inter-comm-line-line-comm*:
assumes $C \neq D$ **and** *line A B* $\neq$ *line C D*
shows *inter* (*line A B*) (*line C D*) = *inter* (*line D C*) (*line A B*)

by (*metis inter-comm line-comm*)

lemma *is-pascal-rot-CW*:
assumes *is-pascal A B C D E F*
shows *is-pascal F A B C D E*
proof −
define *P Q R* **where** *P* = *inter* (*line A B*) (*line D E*) **and** *Q* = *inter* (*line B C*) (*line E F*) **and**
R = *inter* (*line F A*) (*line C D*)
have *col P Q R* **if** *distinct* [*F*,*A*,*B*,*C*,*D*,*E*] **and** *line A B* ≠ *line D E* **and** *line B C* ≠ *line E F*
and *line F A* ≠ *line C D*
using *P-def Q-def R-def assms col-rot-CW distinct6-def inter-comm is-pascal-def line-comm*
that(*1*) *that*(*2*) *that*(*3*) *that*(*4*)
by *auto*
then show *is-pascal F A B C D E*
by (*metis P-def Q-def R-def is-pascal-def line-comm*)
qed

lemma *incid-C-AB*:
assumes *A* ≠ *B* **and** *incid A l* **and** *incid B l* **and** *incid C l*
shows *incid C* (*line A B*)
using *assms ax-uniqueness incidA-lAB incidB-lAB*
by *blast*

lemma *incid-inters-left*:
assumes *P* ∈ *inters l m*
shows *incid P l*
using *assms inters-def*
by *auto*

lemma *incid-inters-right*:
assumes *P* ∈ *inters l m*
shows *incid P m*
using *assms incid-inters-left inters-comm*
by *blast*

lemma *inter-in-inters*: *inter l m* ∈ *inters l m*
proof −
have ∃ *P*. *P* ∈ *inters l m*
using *inters-def ax2*
by *auto*
show *inter l m* ∈ *inters l m*

by (*metis* ‹∃ *P*. *P* ∈ *inters l m*› *inter-def some-eq-ex*)
qed

lemma *incid-inter-left*: *incid* (*inter l m*) *l*
using *incid-inters-left inter-in-inters*
by *blast*

lemma *incid-inter-right*: *incid* (*inter l m*) *m*
using *incid-inter-left inter-comm*
by *fastforce*

lemma *col-A-B-ABl*: *col A B* (*inter* (*line A B*) *l*)
using *col-def incidA-lAB incidB-lAB incid-inter-left*
by *blast*

lemma *col-A-B-lAB*: *col A B* (*inter l* (*line A B*))
using *col-A-B-ABl inter-comm*
by *auto*

lemma *inter-is-a-intersec*: *is-a-intersec* (*inter* (*line A B*) (*line C D*)) *A B C D*
by (*simp add*: *col-A-B-ABl col-A-B-lAB col-rot-CW is-a-intersec-def*)

definition *line-ext* :: *'line* ⇒ *'point set* **where**
line-ext l ≡ {*P*. *incid P l*}

lemma *line-left-inter-1*:
assumes *P* ∈ *line-ext l* **and** *P* ∉ *line-ext m*
shows *line* (*inter l m*) *P* = *l*
by (*metis CollectD CollectI assms*(*1*) *assms*(*2*) *incidA-lAB incidB-lAB incid-inter-left*

incid-inter-right line-ext-def uniq-inter)

lemma *line-left-inter-2*:
assumes *P* ∈ *line-ext m* **and** *P* ∉ *line-ext l*
shows *line* (*inter l m*) *P* = *m*
using *assms inter-comm line-left-inter-1*
by *fastforce*

lemma *line-right-inter-1*:
assumes *P* ∈ *line-ext l* **and** *P* ∉ *line-ext m*
shows *line P* (*inter l m*) = *l*
by (*metis assms line-comm line-left-inter-1*)

lemma *line-right-inter-2*:
assumes *P* ∈ *line-ext m* **and** *P* ∉ *line-ext l*
shows *line P* (*inter l m*) = *m*
by (*metis assms inter-comm line-comm line-left-inter-1*)

lemma *inter-ABC-1*:

assumes *line* A $B \neq$ *line* C A
shows *inter* (*line* A B) (*line* C A) $= A$
using *assms ax-uniqueness incidA-lAB incidB-lAB incid-inter-left incid-inter-right*

by *blast*

lemma *line-inter-2*:
assumes *inter* l $m \neq$ *inter* l' m
shows *line* (*inter* l m) (*inter* l' m) $= m$
using *assms ax-uniqueness incidA-lAB incidB-lAB incid-inter-right*
by *blast*

lemma *col-line-ext-1*:
assumes *col* A B C **and** $A \neq C$
shows $B \in$ *line-ext* (*line* A C)
by (*metis CollectI assms ax-uniqueness col-def incidA-lAB incidB-lAB line-ext-def*)

lemma *inter-line-ext-1*:
assumes *inter* l $m \in$ *line-ext* n **and** $l \neq m$ **and** $l \neq n$
shows *inter* l $m =$ *inter* l n
using *assms*(*1*) *assms*(*3*) *ax-uniqueness incid-inter-left incid-inter-right line-ext-def*

by *blast*

lemma *inter-line-ext-2*:
assumes *inter* l $m \in$ *line-ext* n **and** $l \neq m$ **and** $m \neq n$
shows *inter* l $m =$ *inter* m n
by (*metis assms inter-comm inter-line-ext-1*)

definition *pascal-prop* :: *bool* **where**
pascal-prop $\equiv \forall A\ B\ C\ D\ E\ F.$ *is-pascal* A B C D E $F \longrightarrow$ *is-pascal* B A C D E F

lemma *pappus-pascal*:
assumes *is-pappus*
shows *pascal-prop*
proof−
have *is-pascal* B A C D E F **if** *is-pascal* A B C D E F **for** A B C D E F
proof−
define X Y Z **where** $X =$ *inter* (*line* A C) (*line* E F) **and** $Y =$ *inter* (*line* C D) (*line* B F)
and $Z =$ *inter* (*line* B A) (*line* D E)
have *col* X Y Z **if** *distinct* [B,A,C,D,E,F] **and** *line* A $C \neq$ *line* E F **and** *line* C $D \neq$ *line* B F
and *line* B $A \neq$ *line* D E **and** *line* B $C =$ *line* E F
by (*smt X-def Y-def ax-uniqueness col-ABA col-rot-CW distinct6-def incidB-lAB incid-inter-left*
incid-inter-right line-comm that(*1*) *that*(*2*) *that*(*3*) *that*(*5*))
have *col* X Y Z **if** *distinct* [B,A,C,D,E,F] **and** *line* A $C \neq$ *line* E F **and** *line* C $D \neq$ *line* B F

and *line* B $A \neq$ *line* D E **and** *line* C $D =$ *line* A F
by (*metis X-def Y-def col-ABA col-rot-CW distinct6-def inter-ABC-1 line-comm that*(*1*) *that*(*2*)
that(*3*) *that*(*5*))
have *col X Y Z* **if** *distinct* [B,A,C,D,E,F] **and** *line* A $C \neq$ *line* E F **and** *line* C $D \neq$ *line* B F
and *line* B $A \neq$ *line* D E **and** *line* B $C \neq$ *line* E F **and** *line* C $D \neq$ *line* A F
proof−
define W **where** $W =$ *inter* (*line* A C) (*line* E F)
have *col A C W*
by (*simp add*: *col-A-B-ABl W-def*)
define P Q R **where** $P =$ *inter* (*line* B C) (*line* E F)
and $Q =$ *inter* (*line* A B) (*line* D E)
and $R =$ *inter* (*line* C D) (*line* A F)
have *col P Q R*
using *P-def Q-def R-def* ‹*is-pascal A B C D E F*› *col-2cycle distinct6-def is-pascal-def*
line-comm that(*1*) *that*(*4*) *that*(*5*) *that*(*6*)
by *auto*

have *col X Y Z* **if** $P = Q$
by (*smt P-def Q-def X-def Y-def Z-def* ‹*distinct* [B,A,C,D,E,F]› *ax-uniqueness col-ABA col-def*
distinct6-def incidA-lAB incidB-lAB incid-inter-left inter-comm that)
have *col X Y Z* **if** $P = R$
by (*smt P-def R-def X-def Y-def Z-def* ‹*distinct* [B,A,C,D,E,F]› ‹*line* A $C \neq$ *line* E F›
‹*line* C $D \neq$ *line* B F› *col-2cycle col-A-B-ABl col-rot-CW distinct6-def incidA-lAB*
incidB-lAB incid-inter-left incid-inter-right that uniq-inter)
have *col X Y Z* **if** $P = A$
by (*smt P-def Q-def R-def X-def Y-def Z-def* ‹$P = Q \Longrightarrow$ *col X Y Z*› ‹$P = R \Longrightarrow$ *col X Y Z*›
‹*col P Q R*› ‹*line* B $C \neq$ *line* E F› *ax-uniqueness col-def incidA-lAB incid-inter-left*
incid-inter-right line-comm that)
have *col X Y Z* **if** $P = C$
by (*smt P-def Q-def R-def X-def Y-def Z-def* ‹$P = R \Longrightarrow$ *col X Y Z*› ‹*col P Q R*›
‹*line* A $C \neq$ *line* E F› *ax-uniqueness col-def incidA-lAB incid-inter-left*
incid-inter-right line-comm that)
have *col X Y Z* **if** $P = W$
by (*smt P-def Q-def R-def W-def X-def Y-def Z-def* ‹$P = C \Longrightarrow$ *col X Y Z*› ‹$P = Q \Longrightarrow$ *col X Y Z*›
‹*col P Q R*› ‹*distinct* [B,A,C,D,E,F]› *ax-uniqueness col-def distinct6-def incidB-lAB*
incid-inter-left incid-inter-right line-comm that)
have *col X Y Z* **if** $Q = R$
by (*smt Q-def R-def X-def Y-def Z-def* ‹*distinct* [B,A,C,D,E,F]› *ax-uniqueness*

col-A-B-lAB
col-rot-CW distinct6-def incidB-lAB incid-inter-right inter-comm line-comm that)
have *col X Y Z* **if** $Q = A$
by (*smt P-def Q-def R-def X-def Y-def Z-def* ‹*col P Q R*› ‹*distinct* [*B*,*A*,*C*,*D*,*E*,*F*]›
‹*line C D* ≠ *line B F*› *ax-uniqueness col-ABA col-def distinct6-def incidA-lAB incidB-lAB*
incid-inter-left incid-inter-right that)
have *col X Y Z* **if** $Q = C$
by (*metis P-def Q-def W-def* ‹*P* = *W* ⟹ *col X Y Z*› ‹*distinct* [*B*,*A*,*C*,*D*,*E*,*F*]› *ax-uniqueness*
distinct6-def incidA-lAB incid-inter-left line-comm that)
have *col X Y Z* **if** $Q = W$
by (*metis Q-def W-def X-def Z-def col-ABA line-comm that*)
have *col X Y Z* **if** $R = A$
by (*smt P-def Q-def R-def W-def X-def Y-def* ‹*P* = *W* ⟹ *col X Y Z*› ‹*Q* = *A* ⟹ *col X Y Z*›
‹*col P Q R*› ‹*distinct* [*B*,*A*,*C*,*D*,*E*,*F*]› *ax-uniqueness col-ABA col-def col-rot-CW distinct6-def*
incidA-lAB incidB-lAB incid-inter-right inter-comm that)
have *col X Y Z* **if** $R = C$
by (*smt P-def Q-def R-def X-def Y-def Z-def* ‹*col P Q R*› ‹*distinct* [*B*,*A*,*C*,*D*,*E*,*F*]›
‹*line A C* ≠ *line E F*› *ax-uniqueness col-def distinct6-def incidA-lAB incidB-lAB*
incid-inter-left inter-comm that)
have *col X Y Z* **if** $R = W$
by (*metis R-def W-def* ‹*R* = *A* ⟹ *col X Y Z*› ‹*R* = *C* ⟹ *col X Y Z*› ‹*line C D* ≠ *line A F*›
ax-uniqueness incidA-lAB incidB-lAB incid-inter-left incid-inter-right that)
have *col X Y Z* **if** $A = W$
by (*smt P-def Q-def R-def W-def X-def Y-def Z-def* ‹*P* = *R* ⟹ *col X Y Z*› ‹*Q* = *A* ⟹ *col X Y Z*›
‹*col P Q R*› ‹*distinct* [*B*,*A*,*C*,*D*,*E*,*F*]› *ax-uniqueness col-def distinct6-def incidA-lAB*
incidB-lAB incid-inter-left incid-inter-right that)
have *col X Y Z* **if** $C = W$
by (*metis P-def W-def* ‹*P* = *C* ⟹ *col X Y Z*› ‹*line B C* ≠ *line E F*› *ax-uniqueness incidB-lAB*
incid-inter-left incid-inter-right that)
have *f1*:*col* (*inter* (*line P C*) (*line A Q*)) (*inter* (*line Q W*) (*line C R*))
(*inter* (*line P W*) (*line A R*)) **if** *distinct* [*P*,*Q*,*R*,*A*,*C*,*W*]
using *assms*(*1*) *is-pappus-def is-pappus2-def* ‹*distinct* [*P*,*Q*,*R*,*A*,*C*,*W*]› ‹*col P Q R*›
‹*col A C W*› *inter-is-a-intersec inter-line-line-comm*
by *presburger*
have *col X Y Z* **if** *C* ∈ *line-ext* (*line E F*)

using *P-def* ‹$P = C \Longrightarrow$ *col X Y Z*› ‹*line B C* $\neq$ *line E F*› *incidB-lAB line-ext-def that uniq-inter*
by *auto*
have *col X Y Z* **if** $A \in$ *line-ext* (*line D E*)
by (*metis Q-def* ‹$Q = A \Longrightarrow$ *col X Y Z*› ‹*line B A* $\neq$ *line D E*› *ax-uniqueness incidA-lAB*
incid-inter-left incid-inter-right line-comm line-ext-def mem-Collect-eq that)
have *col X Y Z* **if** *line B C* = *line A B*
by (*metis P-def W-def* ‹$P = W \Longrightarrow$ *col X Y Z*› ‹*distinct* [*B*,*A*,*C*,*D*,*E*,*F*]› *ax-uniqueness*
distinct6-def incidA-lAB incidB-lAB that)

have *f2*:*inter* (*line P C*) (*line A Q*) = *B* **if**
C $\notin$ *line-ext* (*line E F*) **and** *A* $\notin$ *line-ext* (*line D E*) **and** *line B C* $\neq$ *line A B*
by (*smt CollectI P-def Q-def ax-uniqueness incidA-lAB incidB-lAB incid-inter-left*
incid-inter-right line-ext-def that(*1*) *that*(*2*) *that*(*3*))

have *col X Y Z* **if** *line E F* = *line A F*
by (*metis W-def* ‹$A = W \Longrightarrow$ *col X Y Z*› ‹*line A C* $\neq$ *line E F*› *inter-ABC-1 inter-comm that*)
have *col X Y Z* **if** $A \in$ *line-ext* (*line C D*)
using *R-def* ‹$R = A \Longrightarrow$ *col X Y Z*› ‹*line C D* $\neq$ *line A F*› *ax-uniqueness incidA-lAB*
incid-inter-left incid-inter-right line-ext-def that
by *blast*
have *col X Y Z* **if** *inter* (*line B C*) (*line E F*) = *inter* (*line A C*) (*line E F*)
by (*simp add*: *P-def W-def* ‹$P = W \Longrightarrow$ *col X Y Z*› *that*)

have *f3*:*inter* (*line P W*) (*line A R*) = *F* **if** *line E F* $\neq$ *line A F* **and** *A* $\notin$ *line-ext* (*line C D*)
and *inter* (*line B C*) (*line E F*) $\neq$ *inter* (*line A C*) (*line E F*)
by (*smt CollectI P-def R-def W-def ax-uniqueness incidA-lAB incidB-lAB incid-inter-left*
incid-inter-right line-ext-def that(*1*) *that*(*2*) *that*(*3*))

have *col X Y Z* **if** $C \in$ *line-ext* (*line A F*)
using *R-def* ‹$R = C \Longrightarrow$ *col X Y Z*› ‹*line C D* $\neq$ *line A F*› *ax-uniqueness incidA-lAB*
incid-inter-left incid-inter-right line-ext-def that
by *blast*
have *f4*:*inter* (*line Q W*) (*line C R*) = *inter* (*line Q W*) (*line C D*) **if** *C* $\notin$ *line-ext* (*line A F*)
using *R-def incidA-lAB line-ext-def line-right-inter-1 that*
by *auto*
then have *inter* (*line Q W*) (*line C D*) $\in$ *line-ext* (*line B F*) **if** *distinct* [*P*,*Q*,*R*,*A*,*C*,*W*]

and $C \notin$ *line-ext* (*line E F*) **and** $A \notin$ *line-ext* (*line D E*) **and** *line B C* $\neq$ *line A B*
and *line E F* $\neq$ *line A F* **and** $A \notin$ *line-ext* (*line C D*)
and *inter* (*line B C*) (*line E F*) $\neq$ *inter* (*line A C*) (*line E F*)
by (*smt R-def* ‹*distinct* [*B*,*A*,*C*,*D*,*E*,*F*]› *ax-uniqueness col-line-ext-1 distinct6-def f1 f2 f3*
incidA-lAB incidB-lAB incid-inter-left that(*1*) *that*(*2*) *that*(*3*) *that*(*5*) *that*(*6*) *that*(*7*))
then have *inter* (*line Q W*) (*line C D*) = *inter* (*line C D*) (*line B F*) **if** *distinct* [*P*,*Q*,*R*,*A*,*C*,*W*]
and $C \notin$ *line-ext* (*line E F*) **and** $A \notin$ *line-ext* (*line D E*) **and** *line B C* $\neq$ *line A B*
and *line E F* $\neq$ *line A F* **and** $A \notin$ *line-ext* (*line C D*)
and *inter* (*line B C*) (*line E F*) $\neq$ *inter* (*line A C*) (*line E F*)
by (*smt W-def* ‹*distinct* [*B*,*A*,*C*,*D*,*E*,*F*]› ‹*line C D* $\neq$ *line B F*› *ax-uniqueness distinct6-def f2*
incidA-lAB incidB-lAB incid-inter-left incid-inter-right inter-line-ext-2 that(*1*) *that*(*2*)
that(*3*) *that*(*5*) *that*(*6*) *that*(*7*))
moreover have *inter* (*line C D*) (*line B F*) $\in$ *line-ext* (*line Q W*) **if** *distinct* [*P*,*Q*,*R*,*A*,*C*,*W*]
and $C \notin$ *line-ext* (*line E F*) **and** $A \notin$ *line-ext* (*line D E*) **and** *line B C* $\neq$ *line A B*
and *line E F* $\neq$ *line A F* **and** $A \notin$ *line-ext* (*line C D*)
and *inter* (*line B C*) (*line E F*) $\neq$ *inter* (*line A C*) (*line E F*)
by (*metis calculation col-2cycle col-A-B-ABl col-line-ext-1 distinct6-def that*(*1*) *that*(*2*)
that(*3*) *that*(*4*) *that*(*5*) *that*(*6*) *that*(*7*))
ultimately have *col* (*inter* (*line A C*) (*line E F*)) (*inter* (*line C D*) (*line B F*))
(*inter* (*line A B*) (*line D E*)) **if** *distinct* [*P*,*Q*,*R*,*A*,*C*,*W*]
and $C \notin$ *line-ext* (*line E F*) **and** $A \notin$ *line-ext* (*line D E*) **and** *line B C* $\neq$ *line A B*
and *line E F* $\neq$ *line A F* **and** $A \notin$ *line-ext* (*line C D*)
and *inter* (*line B C*) (*line E F*) $\neq$ *inter* (*line A C*) (*line E F*)
by (*metis Q-def W-def col-A-B-ABl col-rot-CW that*(*1*) *that*(*2*) *that*(*3*) *that*(*4*) *that*(*5*) *that*(*6*)
that(*7*))
show *col X Y Z*
by (*metis P-def W-def X-def Y-def Z-def* ‹$A = W \Longrightarrow$ *col X Y Z*› ‹$A \in$ *line-ext* (*line C D*) $\Longrightarrow$ *col X Y Z*›
‹$A \in$ *line-ext* (*line D E*) $\Longrightarrow$ *col X Y Z*› ‹$C = W \Longrightarrow$ *col X Y Z*› ‹$C \in$ *line-ext* (*line E F*) $\Longrightarrow$ *col X Y Z*›
‹$P = A \Longrightarrow$ *col X Y Z*› ‹$P = C \Longrightarrow$ *col X Y Z*› ‹$P = Q \Longrightarrow$ *col X Y Z*› ‹$P = R \Longrightarrow$ *col X Y Z*›
‹*inter* (*line B C*) (*line E F*) = *inter* (*line A C*) (*line E F*) $\Longrightarrow$ *col X Y Z*›
‹$Q = A \Longrightarrow$ *col X Y Z*› ‹$Q = C \Longrightarrow$ *col X Y Z*› ‹$Q = R \Longrightarrow$ *col X Y Z*› ‹$Q = W \Longrightarrow$ *col X Y Z*› ‹$R = A \Longrightarrow$ *col X Y Z*›

‹$R = C \Longrightarrow$ *col X Y Z*› ‹$R = W \Longrightarrow$ *col X Y Z*› ‹⟦*distinct* [P,Q,R,A,C,W]; $C \notin$ *line-ext* (*line E F*); $A \notin$ *line-ext* (*line D E*); *line B C* $\neq$ *line A B*; *line E F* $\neq$ *line A F*; $A \notin$ *line-ext* (*line C D*); *inter* (*line B C*) (*line E F*) $\neq$ *inter* (*line A C*) (*line E F*)⟧ $\Longrightarrow$ *col* (*inter* (*line A C*) (*line E F*)) (*inter* (*line C D*) (*line B F*)) (*inter* (*line A B*) (*line D E*))›
‹*line B C* = *line A B* $\Longrightarrow$ *col X Y Z*› ‹*line E F* = *line A F* $\Longrightarrow$ *col X Y Z*› *distinct6-def line-comm*)
qed
show *is-pascal B A C D E F*
using *X-def Y-def Z-def* ‹⟦*distinct* [B,A,C,D,E,F]; *line A C* $\neq$ *line E F*; *line C D* $\neq$ *line B F*; *line B A* $\neq$ *line D E*; *line B C* = *line E F*⟧ $\Longrightarrow$ *col X Y Z*›
‹⟦*distinct* [B,A,C,D,E,F]; *line A C* $\neq$ *line E F*; *line C D* $\neq$ *line B F*; *line B A* $\neq$ *line D E*; *line B C* $\neq$ *line E F*; *line C D* $\neq$ *line A F*⟧ $\Longrightarrow$ *col X Y Z*›
‹⟦*distinct* [B,A,C,D,E,F]; *line A C* $\neq$ *line E F*; *line C D* $\neq$ *line B F*; *line B A* $\neq$ *line D E*; *line C D* = *line A F*⟧ $\Longrightarrow$ *col X Y Z*›
is-pascal-def
by *force*
qed
thus *pascal-prop* **using** *pascal-prop-def*
by *auto*
qed

lemma *is-pascal-under-alternate-vertices*:
assumes *pascal-prop* **and** *is-pascal* A B C A' B' C'
shows *is-pascal* A B' C A' B C'
using *assms pascal-prop-def is-pascal-rot-CW*
by *presburger*

lemma *col-inter*:
assumes *distinct* [A,B,C,D,E,F] **and** *col A B C* **and** *col D E F*
shows *inter* (*line B C*) (*line E F*) = *inter* (*line A B*) (*line D E*)
by (*smt assms ax-uniqueness col-def distinct6-def incidA-lAB incidB-lAB*)

lemma *pascal-pappus1*:
assumes *pascal-prop*
shows *is-pappus1* A B C A' B' C' P Q R
proof−
define *a1 a2 a3 a4 a5 a6* **where** *a1* = *distinct* [A,B,C,A',B',C'] **and** *a2* = *col A B C* **and**
a3 = *col* A' B' C' **and** *a4* = *is-a-proper-intersec* P A B' A' B **and** *a5* = *is-a-proper-intersec* Q B C' B' C
and *a6* = *is-a-proper-intersec* R A C' A' C

have *inter* (*line B C*) (*line* B' C') = *inter* (*line A B*) (*line* A' B') **if** *a1 a2 a3 a4 a5 a6*
using *a1-def a2-def a3-def col-inter that*(*1*) *that*(*2*) *that*(*3*)
by *blast*
then have *is-pascal* A B C A' B' C' **if** *a1 a2 a3 a4 a5 a6*
using *a1-def col-ABA is-pascal-def that*(*1*) *that*(*2*) *that*(*3*) *that*(*4*) *that*(*5*)

that(*6*)
  by *auto*
 then have *is-pascal A B′ C A′ B C′* **if** *a1 a2 a3 a4 a5 a6*
  using *assms is-pascal-under-alternate-vertices that*(*1*) *that*(*2*) *that*(*3*) *that*(*4*) *that*(*5*) *that*(*6*)
  by *blast*
 then have *col P Q R* **if** *a1 a2 a3 a4 a5 a6*
  by (*smt a1-def a4-def a5-def a6-def ax-uniqueness col-def distinct6-def incidB-lAB incid-inter-left*
   incid-inter-right is-a-proper-intersec-def is-pascal-def line-comm that(*1*) *that*(*2*) *that*(*3*)
   that(*4*) *that*(*5*) *that*(*6*))
 show *is-pappus1 A B C A′ B′ C′ P Q R*
  by (*simp add*: ‹⟦*a1*; *a2*; *a3*; *a4*; *a5*; *a6*⟧ ⟹ *col P Q R*› *a1-def a2-def a3-def a4-def a5-def a6-def*
   is-pappus1-def)
qed

lemma *pascal-pappus*:
 assumes *pascal-prop*
 shows *is-pappus*
 by (*simp add*: *assms is-pappus-def pappus12 pascal-pappus1*)

theorem *pappus-iff-pascal*: *is-pappus* = *pascal-prop*
 using *pappus-pascal pascal-pappus*
 by *blast*

end

end
theory *Desargues-Property*
 imports *Main Projective-Plane-Axioms Pappus-Property Pascal-Property*
begin

Contents:

- We formalize Desargues's property, *desargues-prop*, that states that if two triangles are perspective from a point, then they are perspective from a line. Note that some planes satisfy that property and some others don't, hence Desargues's property is not a theorem though it is a theorem in projective space geometry.

4 Desargues's Property

context *projective-plane*
begin

lemma *distinct3-def*:
distinct [*A*, *B*, *C*] = (*A* ≠ *B* ∧ *A* ≠ *C* ∧ *B* ≠ *C*)
by *auto*

definition *triangle* :: [*'point*, *'point*, *'point*] ⇒ *bool* **where**
triangle A B C ≡ *distinct* [*A*,*B*,*C*] ∧ (*line A B* ≠ *line A C*)

definition *meet-in* :: *'line* ⇒ *'line* => *'point* => *bool* **where**
meet-in l m P ≡ *incid P l* ∧ *incid P m*

lemma *meet-col-1*:
assumes *meet-in* (*line A B*) (*line C D*) *P*
shows *col A B P*
using *assms col-def incidA-lAB incidB-lAB meet-in-def*
by *blast*

lemma *meet-col-2*:
assumes *meet-in* (*line A B*) (*line C D*) *P*
shows *col C D P*
using *assms meet-col-1 meet-in-def*
by *auto*

definition *meet-3-in* :: [*'line*, *'line*, *'line*, *'point*] ⇒ *bool* **where**
meet-3-in l m n P ≡ *meet-in l m P* ∧ *meet-in l n P*

lemma *meet-all-3*:
assumes *meet-3-in l m n P*
shows *meet-in m n P*
using *assms meet-3-in-def meet-in-def*
by *auto*

lemma *meet-comm*:
assumes *meet-in l m P*
shows *meet-in m l P*
using *assms meet-in-def*
by *auto*

lemma *meet-3-col-1*:
assumes *meet-3-in* (*line A B*) *m n P*
shows *col A B P*
using *assms meet-3-in-def meet-col-2 meet-in-def*
by *auto*

lemma *meet-3-col-2*:
assumes *meet-3-in l* (*line A B*) *n P*
shows *col A B P*
using *assms col-def incidA-lAB incidB-lAB meet-3-in-def meet-in-def*
by *blast*

lemma *meet-3-col-3*:
  assumes *meet-3-in l m* (*line A B*) *P*
  shows *col A B P*
  using *assms meet-3-col-2 meet-3-in-def*
  by *auto*

lemma *distinct7-def*: *distinct* $[A,B,C,D,E,F,G] = ((A \neq B) \wedge (A \neq C) \wedge (A \neq D) \wedge (A \neq E) \wedge (A \neq F) \wedge (A \neq G) \wedge$
$(B \neq C) \wedge (B \neq D) \wedge (B \neq E) \wedge (B \neq F) \wedge (B \neq G) \wedge$
$(C \neq D) \wedge (C \neq E) \wedge (C \neq F) \wedge (C \neq G) \wedge$
$(D \neq E) \wedge (D \neq F) \wedge (D \neq G) \wedge$
$(E \neq F) \wedge (E \neq G) \wedge$
$(F \neq G))$
  by *auto*

definition *desargues-config* ::
  [*'point*, *'point*, *'point*, *'point*, *'point*, *'point*, *'point*, *'point*, *'point*, *'point*] => *bool*
where
desargues-config $A\ B\ C\ A'\ B'\ C'\ M\ N\ P\ R \equiv$ *distinct* $[A,B,C,A',B',C',R] \wedge \neg$ *col* $A\ B\ C$
$\wedge \neg$ *col* $A'\ B'\ C' \wedge$ *distinct* [(*line* $A\ A'$),(*line* $B\ B'$),(*line* $C\ C'$)] $\wedge$
meet-3-in (*line* $A\ A'$) (*line* $B\ B'$) (*line* $C\ C'$) $R \wedge$ (*line* $A\ B$) $\neq$ (*line* $A'\ B'$) $\wedge$
(*line* $B\ C$) $\neq$ (*line* $B'\ C'$) $\wedge$ (*line* $A\ C$) $\neq$ (*line* $A'\ C'$) $\wedge$ *meet-in* (*line* $B\ C$) (*line* $B'\ C'$) $M \wedge$
meet-in (*line* $A\ C$) (*line* $A'\ C'$) $N \wedge$ *meet-in* (*line* $A\ B$) (*line* $A'\ B'$) P

lemma *distinct7-rot-CW*:
  assumes *distinct* $[A,B,C,D,E,F,G]$
  shows *distinct* $[C,A,B,F,D,E,G]$
  using *assms distinct7-def*
  by *auto*

lemma *desargues-config-rot-CW*:
  assumes *desargues-config* $A\ B\ C\ A'\ B'\ C'\ M\ N\ P\ R$
  shows *desargues-config* $C\ A\ B\ C'\ A'\ B'\ P\ M\ N\ R$
  by (*smt assms col-rot-CW desargues-config-def distinct3-def distinct7-rot-CW line-comm*
      meet-3-in-def meet-all-3 meet-comm)

lemma *desargues-config-rot-CCW*:
  assumes *desargues-config* $A\ B\ C\ A'\ B'\ C'\ M\ N\ P\ R$
  shows *desargues-config* $B\ C\ A\ B'\ C'\ A'\ N\ P\ M\ R$
  by (*simp add*: *assms desargues-config-rot-CW*)

definition *are-perspective-from-point* ::
  [*'point*, *'point*, *'point*, *'point*, *'point*, *'point*, *'point*] ⇒ *bool* **where**
are-perspective-from-point A B C A′ B′ C′ R ≡ *distinct* [*A*,*B*,*C*,*A′*,*B′*,*C′*,*R*] ∧
triangle A B C ∧
triangle A′ B′ C′ ∧ *distinct* [(*line A A′*),(*line B B′*),(*line C C′*)] ∧
meet-3-in (*line A A′*) (*line B B′*) (*line C C′*) *R*

definition *are-perspective-from-line* ::
  [*'point*, *'point*, *'point*, *'point*, *'point*, *'point*] ⇒ *bool* **where**
are-perspective-from-line A B C A′ B′ C′ ≡ *distinct* [*A*,*B*,*C*,*A′*,*B′*,*C′*] ⟶ *triangle A B C* ⟶
triangle A′ B′ C′ ⟶ *line A B* ≠ *line A′ B′* ⟶ *line A C* ≠ *line A′ C′* ⟶ *line B C* ≠ *line B′ C′* ⟶
col (*inter* (*line A B*) (*line A′ B′*)) (*inter* (*line A C*) (*line A′ C′*)) (*inter* (*line B C*) (*line B′ C′*))

lemma *meet-in-inter*:
  assumes *l* ≠ *m*
  shows *meet-in l m* (*inter l m*)
  by (*simp add*: *incid-inter-left incid-inter-right meet-in-def*)

lemma *perspective-from-point-desargues-config*:
  assumes *are-perspective-from-point A B C A′ B′ C′ R* **and** *line A B* ≠ *line A′ B′* **and**
    line A C ≠ *line A′ C′* **and** *line B C* ≠ *line B′ C′*
  shows *desargues-config A B C A′ B′ C′* (*inter* (*line B C*) (*line B′ C′*)) (*inter* (*line A C*) (*line A′ C′*))
    (*inter* (*line A B*) (*line A′ B′*)) *R*
  unfolding *desargues-config-def distinct7-def distinct3-def*

  using *assms are-perspective-from-point-def* **apply** *auto*
    apply (*smt* (*z3*) *ax-uniqueness col-2cycle col-line-ext-1 incidB-lAB line-ext-def mem-Collect-eq triangle-def*)
  apply (*smt* (*z3*) *ax-uniqueness col-def incidA-lAB line-comm triangle-def*)
  using *meet-in-inter* **apply** *presburger+*
  done

definition *desargues-prop* :: *bool* **where**
desargues-prop ≡
∀ *A B C A′ B′ C′ P*.
  are-perspective-from-point A B C A′ B′ C′ P ⟶ *are-perspective-from-line A B C A′ B′ C′*

end

end
theory *Pappus-Desargues*

imports *Main Projective-Plane-Axioms Pappus-Property Pascal-Property Desargues-Property*
begin

Contents:

- We prove Hessenberg's theorem *hessenberg-theorem*: Pappus's property implies Desargues's property in a projective plane.

5 Hessenberg's Theorem

```
context projective-plane
begin

lemma col-ABC-ABD-1:
  assumes A ≠ B and col A B C and col A B D
  shows col B C D
  by (metis assms ax-uniqueness col-def)

lemma col-ABC-ABD-2:
  assumes A ≠ B and col A B C and col A B D
  shows col A C D
  by (metis assms ax-uniqueness col-def)

lemma col-line-eq-1:
  assumes A ≠ B and B ≠ Cand col A B C
  shows line A B = line B C
  by (metis assms ax-uniqueness col-def incidA-lAB incidB-lAB)

lemma col-line-eq-2:
  assumes A ≠ B and A ≠ C and col A B C
  shows line A B = line A C
  by (metis assms col-line-eq-1 col-rot-CW line-comm)

lemma desargues-config-not-col-1:
  assumes desargues-config A B C A′ B′ C′ M N P R
  shows ¬ col A A′ B′
proof
  assume a1:col A A′ B′
  have f1:A ≠ A′
    using assms desargues-config-def distinct7-def
    by force
  have f2:col A A′ R
    using assms desargues-config-def meet-3-col-1
    by blast
  from f1 and f2 and a1 have f3:col A′ B′ R
    using col-ABC-ABD-1
    by blast
```

from *a1* **have** *f4*:*line A A′ = line A′ B′*
by (*metis assms ax-uniqueness col-def desargues-config-def f1 incidA-lAB incidB-lAB*)
have *f5*:$A' \neq B'$
using *assms desargues-config-def distinct7-def*
by *force*
have *f6*:$B' \neq R$
using *assms desargues-config-def distinct7-def*
by *force*
from *f3* **and** *f5* **and** *f6* **have** *f7*:*line A′ B′ = line B′ R*
using *col-line-eq-1*
by *blast*
have *line B′ R = line B B′*
by (*metis a1 assms col-2cycle col-AAB col-line-eq-1 desargues-config-def f6 meet-3-col-2*)
have *line A A′ = line B B′*
by (*simp add*: ‹*line B′ R = line B B′*› *f4 f7*)
then have *False*
using *assms desargues-config-def distinct3-def*
by *auto*
then show *f8*:*col A A′ B′* $\Longrightarrow$ *False*
by *simp*
qed

lemma *desargues-config-not-col-2*:
assumes *desargues-config A B C A′ B′ C′ M N P R*
shows ¬ *col B B′ C′*
using *assms desargues-config-not-col-1 desargues-config-rot-CCW*
by *blast*

lemma *desargues-config-not-col-3*:
assumes *desargues-config A B C A′ B′ C′ M N P R*
shows ¬ *col C C′ B′*
by (*smt assms col-line-eq-1 col-rot-CW desargues-config-def desargues-config-not-col-2*

desargues-config-rot-CW distinct3-def meet-3-in-def meet-col-1 meet-col-2)

lemma *desargues-config-not-col-4*:
assumes *desargues-config A B C A′ B′ C′ M N P R*
shows ¬ *col A A′ C′*
using *assms desargues-config-not-col-3 desargues-config-rot-CCW*
by *blast*

lemma *desargues-config-not-col-5*:
assumes *desargues-config A B C A′ B′ C′ M N P R*
shows ¬ *col B B′ A′*
using *assms desargues-config-not-col-3 desargues-config-rot-CW*
by *blast*

```
lemma desargues-config-not-col-6:
  assumes desargues-config A B C A' B' C' M N P R
  shows ¬ col C C' A'
  using assms desargues-config-not-col-1 desargues-config-rot-CW
  by blast

lemma desargues-config-not-col-7:
  assumes desargues-config A B C A' B' C' M N P R
  shows ¬ col A B B'
proof
  assume a1:col A B B'
  have f1:col A B R
     by (metis a1 assms col-ABB col-ABC-ABD-2 col-rot-CW desargues-config-def
desargues-config-not-col-5
        meet-3-col-2)
  have f2:col A A' R
    using assms desargues-config-def meet-3-col-1
    by blast
  have f3:A ≠ A'
    using assms col-AAB desargues-config-not-col-4
    by blast
  have f4:A ≠ R using assms desargues-config-def distinct7-def
    by auto
  from f2 and f3 and f4 have f5:line A A' = line A R
    using col-line-eq-2
    by blast
  from f1 have f6:line A R = line B R
   by (metis a1 assms col-2cycle col-ABC-ABD-2 desargues-config-not-col-1 f2 f4)
  have f7:line B R = line B B'
      by (metis ‹line A R = line B R› a1 assms col-AAB col-line-eq-1 desar-
gues-config-def
        desargues-config-not-col-2 f1)
  from f5 and f6 and f7 have line A A' = line B B'
    by simp
  then have False
    using assms desargues-config-def distinct3-def
    by auto
  thus col A B B' ⟹ False
    by simp
qed

lemma desargues-config-not-col-8:
  assumes desargues-config A B C A' B' C' M N P R
  shows ¬ col A C C'

  by (smt (z3) assms col-ABA col-line-eq-1 col-rot-CW desargues-config-def desar-
gues-config-not-col-6 desargues-config-not-col-7 distinct3-def meet-3-col-1 meet-3-col-3
projective-plane.meet-all-3 projective-plane.meet-col-1 projective-plane-axioms)
```

```
lemma desargues-config-not-col-9:
  assumes desargues-config A B C A′ B′ C′ M N P R
  shows ¬ col B A A′
  using assms desargues-config-not-col-8 desargues-config-rot-CCW
  by blast

lemma desargues-config-not-col-10:
  assumes desargues-config A B C A′ B′ C′ M N P R
  shows ¬ col B C C′
  using assms desargues-config-not-col-7 desargues-config-rot-CCW
  by blast

lemma desargues-config-not-col-11:
  assumes desargues-config A B C A′ B′ C′ M N P R
  shows ¬ col C A A′
  using assms desargues-config-not-col-7 desargues-config-rot-CW
  by blast

lemma desargues-config-not-col-12:
  assumes desargues-config A B C A′ B′ C′ M N P R
  shows ¬ col C B B′
  using assms desargues-config-not-col-8 desargues-config-rot-CW
  by blast

lemma col-inter:
  assumes A ≠ C and B ≠ C and col A B C
  shows inter l (line B C) = inter l (line A C)
  by (metis assms col-line-eq-1 col-line-eq-2)

lemma lemma-1:
  assumes desargues-config A B C A′ B′ C′ M N P R and is-pappus
  shows col M N P ∨ incid A (line B′ C′) ∨ incid C′ (line A B)
proof−
  have ?thesis if incid A (line B′ C′) ∨ incid C′ (line A B)
    by (simp add: that)

  define Q E X F where Q = inter (line A B) (line B′ C′) and E = inter (line
A C) (line R Q)
    and X = inter (line A C′) (line R B) and F = inter (line A′ C′) (line R Q)
  have col X E M if ¬ incid A (line B′ C′) and ¬ incid C′ (line A B)
  proof−
    have f1:distinct [C,C′,R,Q,B,A]
      by (smt Q-def ‹¬ incid A (line B′ C′)› ‹¬ incid C′ (line A B)› assms(1)
col-ABB col-A-B-ABl
        col-A-B-lAB col-line-eq-2 col-rot-CW desargues-config-def desargues-config-not-col-12

        desargues-config-not-col-2 desargues-config-not-col-3 desargues-config-not-col-7
```

desargues-config-not-col-9 distinct6-def incidA-lAB line-comm meet-3-col-1 meet-3-col-2)
have *f2*: *col C C′ R*
using *assms*(*1*) *desargues-config-def meet-3-col-3*
by *blast*
have *f3*: *col Q B A*
using *Q-def col-2cycle col-A-B-ABl col-rot-CW*
by *blast*
have *f4*: *is-a-intersec E C A Q R*
using *E-def col-2cycle inter-is-a-intersec is-a-intersec-def*
by *auto*
have *f5*:*is-a-intersec M C B Q C′*
by (*metis Q-def assms*(*1*) *col-2cycle col-ABB col-ABC-ABD-1 col-A-B-lAB col-rot-CW*
desargues-config-def is-a-intersec-def meet-col-1 meet-col-2)
have *f6*:*is-a-intersec X C′ A B R*
using *X-def col-2cycle inter-is-a-intersec is-a-intersec-def*
by *auto*
from *f1* **and** *f2* **and** *f3* **and** *f4* **and** *f5* **and** *f6* **have** *col M X E*
using *assms*(*2*) *is-pappus2-def is-pappus-def*
by *blast*
then show *col X E M*
using *col-def*
by *auto*
qed
have *col P X F* **if** ¬ *incid A* (*line B′ C′*) **and** ¬ *incid C′* (*line A B*)
proof−
have *f1*:*distinct* [*A′*,*A*,*R*,*Q*,*B′*,*C′*]
by (*smt Q-def* ‹¬ *incid A* (*line B′ C′*)› ‹¬ *incid C′* (*line A B*)› *assms*(*1*) *col-AAB col-A-B-ABl*
col-A-B-lAB col-line-eq-1 col-rot-CW desargues-config-def desargues-config-not-col-2
desargues-config-not-col-3 desargues-config-not-col-4 desargues-config-not-col-6
desargues-config-not-col-7 distinct6-def incidB-lAB meet-3-col-2 meet-3-col-3)
have *f2*:*col A′ A R*
by (*metis assms*(*1*) *col-ABA col-line-eq-1 desargues-config-def meet-3-col-1*)
have *f3*:*col Q B′ C′*
by (*simp add*: *Q-def col-A-B-lAB col-rot-CW*)
have *is-a-intersec* (*inter* (*line A B*) (*line A′ B′*)) *A′ B′ Q A*
by (*metis Q-def col-def incidA-lAB incid-inter-left inter-is-a-intersec is-a-intersec-def*)
then have *f4*:*is-a-intersec P A′ B′ Q A*
using *assms*(*1*) *desargues-config-def meet-in-def uniq-inter*
by *auto*
have *f5*:*is-a-intersec X A C′ B′ R*
by (*metis X-def assms*(*1*) *col-def col-line-eq-2 desargues-config-def desargues-config-not-col-9*
f2 inter-is-a-intersec is-a-intersec-def line-comm meet-3-col-2)
have *f6*:*is-a-intersec F A′ C′ Q R*

by (*metis F-def inter-is-a-intersec inter-line-line-comm*)
from *f1* **and** *f2* **and** *f3* **and** *f4* **and** *f5* **and** *f6* **and** *assms*(*2*)
show *col P X F*
using *is-pappus2-def is-pappus-def*
by *blast*
qed
have *col M N P* **if** ¬ *incid A* (*line B′ C′*) **and** ¬ *incid C′* (*line A B*)
proof−
have *f1*:*Q* ≠ *C′* ∧ *X* ≠ *E* ∧ *line Q C′* ≠ *line X E*
by (*smt E-def Q-def X-def* ‹¬ *incid A* (*line B′ C′*)› ‹¬ *incid C′* (*line A B*)›
assms(*1*) *col-ABB*
col-A-B-ABl col-A-B-lAB col-line-eq-2 col-rot-CW desargues-config-def
desargues-config-not-col-10 desargues-config-not-col-2 desargues-config-not-col-8

incidB-lAB incid-C-AB line-comm meet-3-col-1 meet-3-col-2 meet-3-col-3)

have *f2*:*E* ≠ *A* ∧ *C′* ≠ *F* ∧ *line E A* ≠ *line C′ F*
by (*smt E-def F-def Q-def X-def* ‹⟦¬ *incid A* (*line B′ C′*); ¬ *incid C′* (*line*
A B)⟧ ⟹ *col X E M*›
assms(*1*) *ax-uniqueness col-def desargues-config-def desargues-config-not-col-10*

desargues-config-not-col-3 f1 incidA-lAB incidB-lAB incid-inter-left
incid-inter-right
meet-in-def that(*1*))
have *f3*:*Q* ≠ *A* ∧ *X* ≠ *F* ∧ *line Q A* ≠ *line X F*
by (*smt F-def Q-def X-def* ‹¬ *incid A* (*line B′ C′*)› ‹¬ *incid C′* (*line A B*)›
assms(*1*)
ax-uniqueness col-def desargues-config-def desargues-config-not-col-10
desargues-config-not-col-2 desargues-config-not-col-7 incidA-lAB in-
cidB-lAB incid-inter-left
incid-inter-right meet-3-col-2 meet-3-col-3)
have *f4*:*col Q E F*
using *E-def F-def col-def incidB-lAB incid-inter-right*
by *blast*
have *f5*:*col X C′ A*
using *X-def col-2cycle col-A-B-ABl col-rot-CW*
by *blast*
have *f6*:*is-a-intersec M Q C′ X E*
by (*metis Q-def* ‹⟦¬ *incid A* (*line B′ C′*); ¬ *incid C′* (*line A B*)⟧ ⟹ *col X*
E M› *assms*(*1*)
col-ABB col-A-B-lAB col-def col-line-eq-1 desargues-config-def incidB-lAB
is-a-intersec-def
meet-in-def that(*1*) *that*(*2*))
have *f7*:*is-a-intersec N E A C′ F*
by (*metis E-def F-def assms*(*1*) *ax-uniqueness col-rot-CW desargues-config-def*
f2 incidA-lAB
incidB-lAB incid-inter-left is-a-intersec-def meet-col-1 meet-col-2)
have *f8*:*is-a-intersec P Q A X F*
by (*metis Q-def* ‹⟦¬ *incid A* (*line B′ C′*); ¬ *incid C′* (*line A B*)⟧ ⟹ *col P*

X F› *assms*(*1*) *col-AAB col-A-B-ABl col-line-eq-2 col-rot-CW desargues-config-def is-a-intersec-def meet-col-1 that*(*1*) *that*(*2*))
  from *f1* **and** *f2* **and** *f3* **and** *f4* **and** *f5* **and** *f6* **and** *f7* **and** *f8* **and** *assms*(*2*)
show *col M N P*
  using *is-pappus2-def is-pappus-def*
  by *blast*
 qed
 show $col\ M\ N\ P \lor incid\ A\ (line\ B'\ C') \lor incid\ C'\ (line\ A\ B)$
  using ‹$[\![\neg\ incid\ A\ (line\ B'\ C');\ \neg\ incid\ C'\ (line\ A\ B)]\!] \Longrightarrow col\ M\ N\ P$›
  by *auto*
qed

corollary *corollary-1*:
 assumes *desargues-config* $A\ B\ C\ A'\ B'\ C'\ M\ N\ P\ R$ **and** *is-pappus*
 shows $col\ M\ N\ P \lor ((incid\ A\ (line\ B'\ C') \lor incid\ C'\ (line\ A\ B)) \land$
 $(incid\ C\ (line\ A'\ B') \lor incid\ B'\ (line\ A\ C)) \land (incid\ B\ (line\ A'\ C') \lor incid\ A'$
$(line\ B\ C)))$
 by (*metis assms*(*1*) *assms*(*2*) *col-rot-CW desargues-config-rot-CCW lemma-1 line-comm*)

definition *triangle-circumscribes-triangle* ::
 [*'point*, *'point*, *'point*, *'point*, *'point*, *'point*] ⇒ *bool* **where**
triangle-circumscribes-triangle $A'\ B'\ C'\ A\ B\ C \equiv incid\ A\ (line\ B'\ C') \land incid\ C$
$(line\ A'\ B') \land$
$incid\ B\ (line\ A'\ C')$

lemma *lemma-2*:
 assumes *desargues-config* $A\ B\ C\ A'\ B'\ C'\ M\ N\ P\ R$ **and** $incid\ A\ (line\ B'\ C')$
$\lor incid\ C'\ (line\ A\ B)$
  and $incid\ C\ (line\ A'\ B') \lor incid\ B'\ (line\ A\ C)$ **and** $incid\ B\ (line\ A'\ C') \lor$
$incid\ A'\ (line\ B\ C)$
 shows $col\ M\ N\ P \lor$ *triangle-circumscribes-triangle* $A\ B\ C\ A'\ B'\ C' \lor$ *triangle-circumscribes-triangle* $A'\ B'\ C'\ A\ B\ C$
 by (*smt assms ax-uniqueness col-def desargues-config-not-col-1 desargues-config-not-col-11 desargues-config-not-col-12 desargues-config-not-col-2*

  desargues-config-not-col-3 desargues-config-not-col-9 incidA-lAB incidB-lAB triangle-circumscribes-triangle-def)

lemma *lemma-3*:
 assumes *is-pappus* **and** *desargues-config* $A\ B\ C\ A'\ B'\ C'\ M\ N\ P\ R$ **and**
  triangle-circumscribes-triangle $A'\ B'\ C'\ A\ B\ C$
 shows *col M N P*
proof−
 define *S T* **where** $S = inter\ (line\ C'\ P)\ (line\ R\ A)$ **and** $T = inter\ (line\ C'\ P)$
$(line\ R\ B)$

 have *col N S B'*
 proof−

have *f1*:*distinct* $[R,C,C',P,B,A]$
by (*smt assms*(*2*) *col-AAB col-line-eq-2 col-rot-CW desargues-config-def desargues-config-not-col-1 desargues-config-not-col-12 desargues-config-not-col-2*

desargues-config-not-col-5 desargues-config-not-col-7 desargues-config-not-col-8

desargues-config-not-col-9 distinct6-def line-comm meet-3-col-1 meet-3-col-2 meet-col-1
meet-col-2)
have *f2*:*col* R C C'
using *assms*(*2*) *col-rot-CW desargues-config-def meet-3-col-3*
by *blast*
have *f3*:*col* P B A
by (*metis assms*(*2*) *col-rot-CW desargues-config-def line-comm meet-col-1*)
have *f4*:*is-a-intersec* B' R B P C
by (*metis assms*(*2*) *assms*(*3*) *col-def desargues-config-def incidB-lAB is-a-intersec-def*

meet-3-col-2 meet-in-def triangle-circumscribes-triangle-def)
have *f5*:*is-a-intersec* S R A P C'
using *S-def col-2cycle inter-is-a-intersec is-a-intersec-def*
by *auto*
have *line* B C' = *line* A' C'
by (*metis* ‹*distinct* $[R,C,C',P,B,A]$› *assms*(*3*) *ax-uniqueness distinct6-def incidA-lAB incidB-lAB*
triangle-circumscribes-triangle-def)
then have *f6*:*is-a-intersec* N C A B C'
by (*metis assms*(*2*) *desargues-config-def inter-is-a-intersec line-comm meet-in-def uniq-inter*)
from *f1* **and** *f2* **and** *f3* **and** *f4* **and** *f5* **and** *f6* **and** *assms*(*1*) **have** *col* B' N S
using *is-pappus2-def is-pappus-def*
by *blast*
then show *col* N S B'
by (*simp add*: *col-rot-CW*)
qed
have *col* M T A'
proof−
have *f1*:*distinct* $[R,C,C',P,A,B]$
by (*smt assms*(*2*) *col-ABA col-line-eq-2 col-rot-CW desargues-config-def desargues-config-not-col-1*
desargues-config-not-col-12 desargues-config-not-col-2 desargues-config-not-col-5

desargues-config-not-col-7 desargues-config-not-col-8 desargues-config-not-col-9 distinct6-def
line-comm meet-3-col-1 meet-3-col-2 meet-col-1 meet-col-2)
have *f2*:*col* R C C'
using *assms*(*2*) *col-rot-CW desargues-config-def meet-3-col-3*
by *blast*
have *f3*:*col* P A B
using *assms*(*2*) *col-rot-CW desargues-config-def meet-col-1*

by *blast*
have *f4*:*line P C* = *line A′ B′*
by (*metis* ‹*distinct* [*R*,*C*,*C′*,*P*,*A*,*B*]› *assms*(*2*) *assms*(*3*) *ax-uniqueness desargues-config-def*
distinct6-def incidA-lAB incidB-lAB meet-in-def triangle-circumscribes-triangle-def)
have *f5*:*line R A* = *line A A′*
by (*metis* ‹*distinct* [*R*,*C*,*C′*,*P*,*A*,*B*]› *assms*(*2*) *col-AAB col-line-eq-2 desargues-config-def*
desargues-config-not-col-1 distinct6-def line-comm meet-3-col-1)
from *f4* **and** *f5* **have** *f6*:*is-a-intersec A′ R A P C*
by (*metis col-def incidA-lAB incidB-lAB is-a-intersec-def*)
have *line A C′* = *line B′ C′*
by (*metis assms*(*3*) *ax-uniqueness distinct6-def f1 incidA-lAB incidB-lAB*
triangle-circumscribes-triangle-def)
then have *f7*:*is-a-intersec M C B A C′*
by (*metis assms*(*2*) *col-rot-CW desargues-config-def is-a-intersec-def line-comm meet-col-1*
meet-col-2)
have *f8*:*is-a-intersec T R B P C′*
by (*metis T-def distinct6-def f1 inter-comm-line-line-comm inter-is-a-intersec line-comm*)
from *f1* **and** *f2* **and** *f3* **and** *f6* **and** *f7* **and** *f8* **and** *assms*(*1*) **have** *col A′ M T*
using *is-pappus2-def is-pappus-def*
by *blast*
thus *col M T A′*
by (*simp add*: *col-rot-CW*)
qed
then show *col M N P*
proof−
have *f1*:*S* ≠ *T* ∧ *B* ≠ *A* ∧ *line S T* ≠ *line B A*
by (*smt T-def* ‹*col N S B′*› *assms*(*2*) *assms*(*3*) *ax-uniqueness col-AAB col-line-eq-2 col-rot-CW*
desargues-config-def desargues-config-not-col-10 desargues-config-not-col-7
desargues-config-not-col-9 incidB-lAB incid-inter-left incid-inter-right
line-comm meet-3-col-2 meet-3-col-3 meet-col-1 meet-col-2 triangle-circumscribes-triangle-def)
have *f2*:*A* ≠ *B′* ∧ *T* ≠ *A′* ∧ *line A B′* ≠ *line T A′*
by (*smt* (*verit*) *S-def T-def assms*(*2*) *assms*(*3*) *col-A-B-ABl col-line-eq-2 desargues-config-def desargues-config-not-col-1 desargues-config-not-col-9 f1 incidA-lAB inter-comm line-comm meet-3-col-1 meet-col-2 projective-plane.col-def projective-plane-axioms triangle-circumscribes-triangle-def*)
have *f3*:*S* ≠ *B′* ∧ *B* ≠ *A′*
by (*smt S-def assms*(*2*) *assms*(*3*) *ax-uniqueness col-A-B-ABl col-line-eq-2 col-rot-CW*
desargues-config-def desargues-config-not-col-2 desargues-config-not-col-5
desargues-config-not-col-7 incidA-lAB incidB-lAB incid-inter-right inter-comm line-comm
meet-3-col-2 meet-in-def triangle-circumscribes-triangle-def)
then have *f4*:*line S B′* ≠ *line B A′*

by (*metis assms*(*2*) *col-def desargues-config-not-col-5 incidA-lAB incidB-lAB*)
have *f5*:*col S A A′*
by (*metis S-def assms*(*2*) *col-ABC-ABD-1 col-A-B-lAB col-rot-CW desargues-config-def*
desargues-config-not-col-8 meet-3-col-1 meet-3-col-3)
have *f6*:*col B T B′*
by (*metis T-def assms*(*2*) *col-def col-line-eq-2 desargues-config-def desargues-config-not-col-10*
incidB-lAB incid-inter-right line-comm meet-3-col-2 meet-3-col-3)
have *f7*:*is-a-intersec P S T B A*
by (*metis S-def T-def assms*(*2*) *col-ABC-ABD-1 col-A-B-ABl col-def desargues-config-def incidA-lAB*
incidB-lAB is-a-intersec-def meet-in-def)
have *f8*:*is-a-intersec M A B′ T A′*
by (*smt* (*verit*, *del-insts*) ‹*col M T A′*› *assms*(*2*) *assms*(*3*) *col-rot-CW desargues-config-def f2 incidA-lAB incidB-lAB is-a-intersec-def meet-col-2 projective-plane.ax-uniqueness projective-plane-axioms triangle-circumscribes-triangle-def*)
have *f9*:*is-a-intersec N S B′ B A′*
using ‹*col N S B′*› *assms*(*2*) *assms*(*3*) *col-def desargues-config-def incidA-lAB is-a-intersec-def*
meet-in-def triangle-circumscribes-triangle-def
by *auto*
from *f1* **and** *f2* **and** *f3* **and** *f4* **and** *f5* **and** *f6* **and** *f7* **and** *f8* **and** *f9* **and** *assms*(*1*) **have** *col P M N*
using *is-pappus2-def is-pappus-def*
by *blast*
thus *col M N P*
by (*simp add*: *col-rot-CW*)
qed
qed

theorem *pappus-desargues*:
assumes *is-pappus* **and** *desargues-config A B C A′ B′ C′ M N P R*
shows *col M N P*
proof−
have *f1*:*col M N P* ∨ ((*incid A* (*line B′ C′*) ∨ *incid C′* (*line A B*)) ∧
(*incid C* (*line A′ B′*) ∨ *incid B′* (*line A C*)) ∧ (*incid B* (*line A′ C′*) ∨ *incid A′* (*line B C*)))
using *assms corollary-1*
by *auto*
have *f2*:*col M N P* ∨ *triangle-circumscribes-triangle A B C A′ B′ C′* ∨ *triangle-circumscribes-triangle A′ B′ C′ A B C*
if (*incid A* (*line B′ C′*) ∨ *incid C′* (*line A B*)) ∧ (*incid C* (*line A′ B′*) ∨ *incid B′* (*line A C*))
∧ (*incid B* (*line A′ C′*) ∨ *incid A′* (*line B C*))
using *assms*(*2*) *lemma-2 that*
by *auto*
have *f3*:*col M N P* **if** *triangle-circumscribes-triangle A′ B′ C′ A B C*
using *assms lemma-3 that*

```
    by auto
  have f4:col M N P if triangle-circumscribes-triangle A B C A′ B′ C′
  proof−
    have desargues-config A′ B′ C′ A B C M N P R
    proof−
      have f1:distinct [A′,B′,C′,A,B,C,R]
        using assms(2) desargues-config-def distinct7-def
        by auto
      have f2:¬ col A′ B′ C′
        using assms(2) desargues-config-def
        by blast
      have f3:¬ col A B C
        using assms(2) desargues-config-def
        by blast
      have f4:distinct [(line A′ A),(line B′ B),(line C′ C)]
        by (metis assms(2) desargues-config-def line-comm)
      have f5:meet-3-in (line A′ A) (line B′ B) (line C′ C) R
        by (metis assms(2) desargues-config-def line-comm)
      have f6:(line A′ B′) ≠ (line A B) ∧ (line B′ C′) ≠ (line B C) ∧ (line A′ C′)
≠ (line A C)
        using assms(2) desargues-config-def
        by auto
      have f7:meet-in (line B′ C′) (line B C) M ∧ meet-in (line A′ C′) (line A C)
N ∧
      meet-in (line A′ B′) (line A B) P
        using assms(2) desargues-config-def meet-comm
        by blast
      from f1 and f2 and f3 and f4 and f5 and f6 and f7 show desargues-config
A′ B′ C′ A B C M N P R
        by (simp add: desargues-config-def)
    qed
    then show col M N P
      using assms(1) lemma-3 that
      by blast
  qed
  from f1 and f2 and f3 and f4 show col M N P
    by blast
qed

theorem hessenberg-thereom:
  assumes is-pappus
  shows desargues-prop
  by (smt are-perspective-from-line-def assms col-def desargues-prop-def pappus-desargues

      perspective-from-point-desargues-config)

corollary pascal-desargues:
  assumes pascal-prop
  shows desargues-prop
```

by (*simp add*: *assms hessenberg-thereom pascal-pappus*)

end

end
theory *Higher-Projective-Space-Rank-Axioms*
imports *Main*
begin

Contents:

- Following [2] we introduce a set of axioms for projective space geometry based on the notions of matroid and rank.

6 A Based-rank Set of Axioms for Projective Space Geometry

locale *higher-projective-space-rank* =

fixes *rk* :: *'point set* ⇒ *nat*

assumes
matroid-ax-1a: $rk\ X \geq 0$ **and**
matroid-ax-1b: $rk\ X \leq card\ X$ **and**
matroid-ax-2: $X \subseteq Y \longrightarrow rk\ X \leq rk\ Y$ **and**
matroid-ax-3: $rk\ (X \cup Y) + rk\ (X \cap Y) \leq rk\ X + rk\ Y$

assumes
rk-ax-singleton: $rk\ \{P\} \geq 1$ **and**
rk-ax-couple: $P \neq Q \longrightarrow rk\ \{P,Q\} \geq 2$ **and**
rk-ax-pasch: $rk\ \{A,B,C,D\} \leq 3 \longrightarrow (\exists J.\ rk\ \{A,B,J\} = 2 \wedge rk\ \{C,D,J\} = 2)$
and
rk-ax-3-pts: $\exists C.\ rk\ \{A,B,C\} = 2 \wedge rk\ \{B,C\} = 2 \wedge rk\ \{A,C\} = 2$ **and**
rk-ax-dim: $\exists A\ B\ C\ D.\ rk\ \{A,B,C,D\} \geq 4$

end
theory *Matroid-Rank-Properties*
imports *Main Higher-Projective-Space-Rank-Axioms*
begin

Contents:

- In this file we introduce the basic lemmas and properties derived from our based-rank axioms that will allow us to simplify our future proofs.

7 Proof Techniques Using Ranks

context *higher-projective-space-rank*
begin

lemma *matroid-ax-3-alt*:
assumes $I \subseteq X \cap Y$
shows $rk\ (X \cup Y) + rk\ I \leq rk\ X + rk\ Y$
by (*metis add.commute add-le-cancel-right assms matroid-ax-2 matroid-ax-3 order-trans*)

lemma *rk-uniqueness*:
assumes $rk\ \{A,B\} = 2$ **and** $rk\ \{C,D\} = 2$ **and** $rk\ \{A,B,M\} \leq 2$ **and** $rk\ \{C,D,M\} \leq 2$ **and**
$rk\ \{A,B,P\} \leq 2$ **and** $rk\ \{C,D,P\} \leq 2$ **and** $rk\ \{A,B,C,D\} \geq 3$
shows $rk\ \{M,P\} = 1$
proof−
have $\{A,B,M\} \cup \{A,B,P\} = \{A,B,M,P\}$
by *auto*
have $rk\ \{A,B,M,P\} + rk\ \{A,B\} \leq rk\ \{A,B,M\} + rk\ \{A,B,P\}$
by (*metis* (*full-types*) ‹$\{A, B, M\} \cup \{A, B, P\} = \{A, B, M, P\}$› *insert-commute le-inf-iff*
matroid-ax-3-alt subset-insertI)
then have $rk\ \{A,B,M,P\} = 2$
by (*metis add-diff-cancel-right′ antisym assms*(*1*) *assms*(*3*) *assms*(*5*) *insert-commute le-diff-conv matroid-ax-2 subset-insertI*)
have $\{C,D,M\} \cup \{C,D,P\} = \{C,D,M,P\}$
by *auto*
have $rk\ \{C,D,M,P\} + rk\ \{C,D\} \leq rk\ \{C,D,M\} + rk\ \{C,D,P\}$
by (*metis Un-insert-left Un-upper1* ‹$\{C, D, M\} \cup \{C, D, P\} = \{C, D, M, P\}$› *insert-is-Un le-inf-iff*
matroid-ax-3-alt)
then have *i1*:$rk\ \{C,D,M,P\} + 2 \leq 4$
using *assms*(*2*) *assms*(*4*) *assms*(*6*)
by *linarith*
moreover have *i2*:$rk\ \{C,D,M,P\} \geq 2$
by (*metis assms*(*2*) *insertI1 insert-subset matroid-ax-2 subset-insertI*)
from *i1* **and** *i2* **have** $rk\ \{C,D,M,P\} = 2$
by *linarith*
have $rk\ \{A,B,C,D,M,P\} \geq 3$
by (*metis Un-insert-right Un-upper2 assms*(*7*) *matroid-ax-2 order-trans sup-bot.right-neutral*)
have $\{A,B,M,P\} \cup \{C,D,M,P\} = \{A,B,C,D,M,P\}$
by *auto*
then have $rk\ \{A,B,C,D,M,P\} + rk\ \{M,P\} \leq rk\ \{A,B,M,P\} + rk\ \{C,D,M,P\}$

by (*smt le-inf-iff matroid-ax-3-alt order-trans subset-insertI*)
then have *i3*:*rk* {*A*,*B*,*C*,*D*,*M*,*P*} + *rk* {*M*,*P*} ≤ *4*
using ‹*rk* {*A*, *B*, *M*, *P*} = *2*› ‹*rk* {*C*, *D*, *M*, *P*} = *2*›
by *linarith*
have *i4*:*rk* {*A*,*B*,*C*,*D*,*M*,*P*} + *rk* {*M*,*P*} ≥ *3* + *rk*{*M*,*P*}
by (*simp add*: ‹*3* ≤ *rk* {*A*, *B*, *C*, *D*, *M*, *P*}›)
from *i3* **and** *i4* **show** *rk* {*M*,*P*} = *1*
by (*metis* (*no-types*, *lifting*) ‹*rk* {*A*, *B*, *C*, *D*, *M*, *P*} + *rk* {*M*, *P*} ≤ *rk* {*A*, *B*, *M*, *P*} + *rk* {*C*, *D*, *M*, *P*}›
‹*rk* {*A*, *B*, *M*, *P*} = *2*› ‹*rk* {*C*, *D*, *M*, *P*} = *2*› *add-le-cancel-left add-numeral-left antisym*
insert-absorb2 numeral-Bit1 numeral-One numeral-plus-one one-add-one one-le-numeral
order-trans rk-ax-couple rk-ax-singleton)
qed

lemma *rk-ax-dim-alt*: ∃ *A B C D*. ∀ *M*. *rk* {*A*,*B*,*M*} ≠ *2* ∨ *rk* {*C*,*D*,*M*} ≠ *2*
proof−
obtain *A B C D* **where** *f1*:*rk* {*A*,*B*,*C*,*D*} ≥ *4*
using *rk-ax-dim*
by *auto*
have ∀ *M*. *rk* {*A*,*B*,*M*} ≠ *2* ∨ *rk* {*C*,*D*,*M*} ≠ *2*
proof
fix *M*
have {*A*,*B*,*C*,*D*,*M*} = {*A*,*B*,*M*} ∪ {*C*,*D*,*M*}
by *auto*
then have *rk* {*A*,*B*,*C*,*D*,*M*} + *rk* {*M*} ≤ *rk* {*A*,*B*,*M*} + *rk* {*C*,*D*,*M*}
by (*smt le-inf-iff matroid-ax-3-alt order-trans subset-insertI*)
then have *rk* {*A*,*B*,*C*,*D*,*M*} ≤ *3* **if** *rk* {*A*,*B*,*M*} = *2* **and** *rk* {*C*,*D*,*M*} = *2*
by (*smt* (*z3*) *One-nat-def Suc-le-eq Suc-numeral add-Suc-right add-le-same-cancel1 nat-1-add-1 not-less numeral-Bit1 numerals*(*1*) *order-trans rk-ax-singleton semiring-norm*(*2*) *that*(*1*) *that*(*2*))
then have *rk* {*A*,*B*,*C*,*D*} ≤ *3* **if** *rk* {*A*,*B*,*M*} = *2* **and** *rk* {*C*,*D*,*M*} = *2*
by (*smt insert-commute matroid-ax-2 order-trans subset-insertI that*(*1*) *that*(*2*))
thus *rk* {*A*, *B*, *M*} ≠ *2* ∨ *rk* {*C*, *D*, *M*} ≠ *2*
using ‹*4* ≤ *rk* {*A*, *B*, *C*, *D*}›
by *linarith*
qed
thus ∃ *A B C D*. ∀ *M*. *rk* {*A*, *B*, *M*} ≠ *2* ∨ *rk* {*C*, *D*, *M*} ≠ *2*
by *blast*
qed

lemma *rk-empty*: *rk* {} = *0*
proof−
have *rk* {} ≥ *0*
by *simp*
have *rk* {} ≤ *0*
by (*metis card.empty matroid-ax-1b*)

thus *rk* {} = *0*
by *blast*
qed

lemma *matroid-ax-2-alt*: *rk X* ≤ *rk* (*X* ∪ {*x*}) ∧ *rk* (*X* ∪ {*x*}) ≤ *rk X* + *1*
proof
have *X* ⊆ *X* ∪ {*x*}
by *auto*
thus *rk X* ≤ *rk* (*X* ∪ {*x*})
by (*simp add*: *matroid-ax-2*)
have *rk* {*x*} ≤ *1*
by (*metis One-nat-def card.empty card-Suc-eq insert-absorb insert-not-empty matroid-ax-1b*)
thus *rk* (*X* ∪ {*x*}) ≤ *rk X* + *1*
by (*metis add-leD1 le-antisym matroid-ax-3 rk-ax-singleton*)
qed

lemma *matroid-ax-3-alt'*: *rk* (*X* ∪ {*y*}) = *rk* (*X* ∪ {*z*}) ⟶ *rk* (*X* ∪ {*z*}) = *rk X* ⟶ *rk X* = *rk* (*X* ∪ {*y*,*z*})
proof−
have *i1*:*rk X* ≤ *rk* (*X* ∪ {*y*,*z*})
using *matroid-ax-2*
by *blast*
have *i2*:*rk X* ≥ *rk* (*X* ∪ {*y*,*z*}) **if** *rk* (*X* ∪ {*y*}) = *rk* (*X* ∪ {*z*}) **and** *rk* (*X* ∪ {*z*}) = *rk X*
proof−
have (*X* ∪ {*y*}) ∪ (*X* ∪ {*z*}) = *X* ∪ {*y*,*z*}
by *blast*
then have *rk* (*X* ∪ {*y*,*z*}) + *rk X* ≤ *rk X* + *rk X*
by (*metis* ‹*rk* (*X* ∪ {*y*}) = *rk* (*X* ∪ {*z*})› ‹*rk* (*X* ∪ {*z*}) = *rk X*› *inf-sup-ord*(*3*) *le-inf-iff*
matroid-ax-3-alt)
thus *rk* (*X* ∪ {*y*,*z*}) ≤ *rk X*
by *simp*
qed
thus *rk* (*X* ∪ {*y*}) = *rk* (*X* ∪ {*z*}) ⟶ *rk* (*X* ∪ {*z*}) = *rk X* ⟶ *rk X* = *rk* (*X* ∪ {*y*, *z*})
using *antisym i1*
by *blast*
qed

lemma *rk-ext*:
assumes *rk X* ≤ *3*
shows ∃ *P*. *rk*(*X* ∪ {*P*}) = *rk X* + *1*
proof−
obtain *A B C D* **where** *rk* {*A*,*B*,*C*,*D*} ≥ *4*
using *rk-ax-dim*
by *auto*
have *f1*:*rk* (*X* ∪ {*A*, *B*, *C*, *D*}) ≥ *4*

by (*metis Un-upper2* ‹*4* ≤ *rk* {*A*, *B*, *C*, *D*}› *matroid-ax-2 sup.coboundedI2 sup.orderE*)
have *rk* (*X* ∪ {*A*, *B*, *C*, *D*}) = *rk X* **if** *rk*(*X* ∪ {*A*}) = *rk*(*X* ∪ {*B*}) **and** *rk*(*X* ∪ {*B*}) = *rk*(*X* ∪ {*C*})
and *rk*(*X* ∪ {*C*}) = *rk*(*X* ∪ {*D*}) **and** *rk*(*X* ∪ {*D*}) = *rk X*
using *matroid-ax-3-alt′ that*(*1*) *that*(*2*) *that*(*3*) *that*(*4*)
by *auto*
then have *f2*:*rk* (*X* ∪ {*A*, *B*, *C*, *D*}) ≤ *3* **if** *rk*(*X* ∪ {*A*}) = *rk*(*X* ∪ {*B*}) **and** *rk*(*X* ∪ {*B*}) = *rk*(*X* ∪ {*C*})
and *rk*(*X* ∪ {*C*}) = *rk*(*X* ∪ {*D*}) **and** *rk*(*X* ∪ {*D*}) = *rk X*
using *assms that*(*1*) *that*(*2*) *that*(*3*) *that*(*4*)
by *linarith*
from *f1* **and** *f2* **have** *False* **if** *rk*(*X* ∪ {*A*}) = *rk*(*X* ∪ {*B*}) **and** *rk*(*X* ∪ {*B*}) = *rk*(*X* ∪ {*C*})
and *rk*(*X* ∪ {*C*}) = *rk*(*X* ∪ {*D*}) **and** *rk*(*X* ∪ {*D*}) = *rk X*
using *that*(*1*) *that*(*2*) *that*(*3*) *that*(*4*)
by *linarith*
then have *rk* (*X* ∪ {*A*}) = *rk X* + *1* ∨ *rk* (*X* ∪ {*B*}) = *rk X* + *1* ∨ *rk* (*X* ∪ {*C*}) = *rk X* + *1* ∨
rk (*X* ∪ {*D*}) = *rk X* + *1*
by (*smt One-nat-def Suc-le-eq Suc-numeral Un-upper2* ‹*4* ≤ *rk* {*A*, *B*, *C*, *D*}›
‹⟦*rk* (*X* ∪ {*A*}) = *rk* (*X* ∪ {*B*}); *rk* (*X* ∪ {*B*}) = *rk* (*X* ∪ {*C*}); *rk* (*X* ∪ {*C*}) = *rk* (*X* ∪ {*D*}); *rk* (*X* ∪ {*D*}) = *rk X*⟧ ⟹ *rk* (*X* ∪ {*A*, *B*, *C*, *D*}) = *rk X*›
add.right-neutral add-Suc-right assms antisym-conv1 matroid-ax-2 matroid-ax-2-alt
not-less semiring-norm(*2*) *semiring-norm*(*8*) *sup.coboundedI2 sup.orderE*)
thus ∃ *P* . *rk* (*X* ∪ {*P*}) = *rk X* + *1*
by *blast*
qed

lemma *rk-singleton* : ∀ *P*. *rk* {*P*} = *1*
proof
fix *P*
have *f1*:*rk* {*P*} ≤ *1*
by (*metis One-nat-def card.empty card-Suc-eq insert-absorb insert-not-empty matroid-ax-1b*)
have *f2*:*rk* {*P*} ≥ *1*
using *rk-ax-singleton*
by *auto*
from *f1* **and** *f2* **show** *rk* {*P*} = *1*
using *antisym*
by *blast*
qed

lemma *rk-singleton-bis* :
assumes *A* = *B*
shows *rk* {*A*, *B*} = *1*
by (*simp add*: *assms rk-singleton*)

```
lemma rk-couple :
  assumes A ≠ B
  shows rk {A, B} = 2
proof−
  have f1:rk {A, B} ≤ 2
    by (metis insert-is-Un matroid-ax-2-alt one-add-one rk-singleton)
  have f2:rk {A, B} ≥ 2
    by (simp add: assms rk-ax-couple)
  from f1 and f2 show ?thesis
    by (simp add: f1 le-antisym)
qed

lemma rk-triple-le : rk {A, B, C} ≤ 3
 by (metis Suc-numeral Un-commute insert-absorb2 insert-is-Un linear matroid-ax-2-alt
numeral-2-eq-2
         numeral-3-eq-3 numeral-le-one-iff numeral-plus-one rk-couple rk-singleton
semiring-norm(70))

lemma rk-couple-to-singleton :
  assumes rk {A, B} = 1
  shows A = B
proof−
  have rk {A, B} = 2 if A ≠ B
    using rk-couple
    by (simp add: that)
  thus A = B
    using assms
    by auto
qed

lemma rk-triple-to-rk-couple :
  assumes rk {A, B, C} = 3
  shows rk {A, B} = 2
proof−
  have rk {A, B} ≤ 2
    using matroid-ax-1b
    by (metis one-le-numeral rk-ax-couple rk-couple rk-singleton-bis)
  have rk {A, B, C} ≤ 2 if rk {A, B} = 1
    using matroid-ax-2-alt[of {A, B} C]
    by (simp add: insert-commute that)
  then have rk {A, B} ≥ 2
    using assms rk-ax-couple rk-singleton-bis
    by force
  thus rk {A, B} = 2
    by (simp add: ‹rk {A, B} ≤ 2› le-antisym)
qed
```

end

end
theory *Desargues-2D*
imports *Main Higher-Projective-Space-Rank-Axioms Matroid-Rank-Properties*
begin

Contents:

- We prove Desargues's theorem: if two triangles ABC and A'B'C' are perspective from a point P (ie. the lines AA', BB' and CC' are concurrent in P), then they are perspective from a line (ie. the points $\alpha = BC \cap B'C'$, $\beta = AC \cap A'C'$ and $\gamma = AB \cap A'B'$ are collinear). In this file we restrict ourself to the case where the two triangles ABC and A'B'C' are coplanar.

8 Desargues's Theorem: The Coplanar Case

context *higher-projective-space-rank*
begin

definition *desargues-config-2D* ::
$[{}'point, {}'point, {}'point, {}'point, {}'point, {}'point, {}'point, {}'point, {}'point, {}'point] \Rightarrow bool$
where *desargues-config-2D* $A\ B\ C\ A'\ B'\ C'\ P\ \alpha\ \beta\ \gamma \equiv rk\ \{A, B, C\} = 3 \wedge rk\ \{A', B', C'\} = 3 \wedge$
$rk\ \{A, A', P\} = 2 \wedge rk\ \{B, B', P\} = 2 \wedge rk\ \{C, C', P\} = 2 \wedge rk\ \{A, B, \gamma\} = 2 \wedge rk\ \{A', B', \gamma\} = 2 \wedge$
$rk\ \{A, C, \beta\} = 2 \wedge rk\ \{A', C', \beta\} = 2 \wedge rk\ \{B, C, \alpha\} = 2 \wedge rk\ \{B', C', \alpha\} = 2 \wedge$
$rk\ \{A, B, C, A', B', C'\} = 3 \wedge$
— We add the following non-degeneracy conditions
$rk\ \{A, B, P\} = 3 \wedge rk\ \{A, C, P\} = 3 \wedge rk\ \{B, C, P\} = 3 \wedge$
$rk\ \{A, A'\} = 2 \wedge rk\ \{B, B'\} = 2 \wedge rk\ \{C, C'\} = 2$

lemma *coplanar-ABCA'B'C'P* :
assumes $rk\ \{A, A'\} = 2$ **and** $rk\ \{A, B, C, A', B', C'\} = 3$ **and** $rk\ \{A, A', P\} = 2$
shows $rk\ \{A, B, C, A', B', C', P\} = 3$
proof−
have $rk\ \{A, B, C, A', B', C', P\} + rk\ \{A, A'\} \le rk\ \{A, B, C, A', B', C'\} + rk\ \{A, A', P\}$
using *matroid-ax-3-alt*[*of* $\{A, A'\}$ $\{A, B, C, A', B', C'\}$ $\{A, A', P\}$]
by (*simp add*: *insert-commute*)
then have $rk\ \{A, B, C, A', B', C', P\} \le 3$
using *assms*(*1*) *assms*(*2*) *assms*(*3*)
by *linarith*
then show $rk\ \{A, B, C, A', B', C', P\} = 3$

using *assms*(*2*) *matroid-ax-2*
by (*metis Un-insert-right Un-upper2 le-antisym sup-bot.right-neutral*)
qed

lemma *non-colinear-*$A'B'P$:
assumes *rk* $\{A, B, P\} = 3$ **and** *rk* $\{A, A', P\} = 2$ **and** *rk* $\{B, B', P\} = 2$ **and** *rk* $\{A', P\} = 2$
and *rk* $\{B', P\} = 2$
shows *rk* $\{A', B', P\} = 3$
proof−
have *f1*:*rk* $\{A', B', P\} \leq 3$
using *rk-triple-le* **by** *auto*
have *rk* $\{A, B, A', B', P\} \geq 3$
using *assms*(*1*) *matroid-ax-2*
by (*metis insert-mono insert-subset subset-insertI*)
then have *f2*:*rk* $\{A, B, A', B', P\} = 3$
using *matroid-ax-3-alt*[*of* $\{P\}$ $\{A, A', P\}$ $\{B, B', P\}$] *assms*(*2*) *assms*(*3*)
by (*simp add*: *insert-commute rk-singleton*)
have *rk* $\{A, B, A', B', P\}$ + *rk* $\{B', P\} \leq$ *rk* $\{A, A', B', P\}$ + *rk* $\{B, B', P\}$
using *matroid-ax-3-alt*[*of* $\{B', P\}$ $\{A, A', B', P\}$ $\{B, B', P\}$]
by (*simp add*: *insert-commute*)
then have *rk* $\{A, A', B', P\} \geq 3$
using *f2 assms*(*3*) *assms*(*5*) **by** *linarith*
then have *f3*:*rk* $\{A, A', B', P\} = 3$
using *f2 matroid-ax-2*
by (*metis eq-iff insert-commute subset-insertI*)
have *rk* $\{A, A', B', P\}$ + *rk* $\{A', P\} \leq$ *rk* $\{A', B', P\}$ + *rk* $\{A, A', P\}$
using *matroid-ax-3-alt*[*of* $\{A', P\}$ $\{A', B', P\}$ $\{A, A', P\}$]
by (*simp add*: *insert-commute*)
then have *rk* $\{A', B', P\} \geq 3$
using *f3 assms*(*2*) *assms*(*4*) **by** *linarith*
thus *rk* $\{A', B', P\} = 3$
using *f1* **by** *auto*
qed

lemma *desargues-config-2D-non-collinear-P* :
assumes *desargues-config-2D A B C A′ B′ C′ P* α β γ **and** *rk* $\{A', P\} = 2$ **and** *rk* $\{B', P\} = 2$
and *rk* $\{C', P\} = 2$
shows *rk* $\{A', B', P\} = 3$ **and** *rk* $\{A', C', P\} = 3$ **and** *rk* $\{B', C', P\} = 3$
proof−
show *rk* $\{A', B', P\} = 3$
using *non-colinear-*$A'B'P$ *assms*(*1*) *desargues-config-2D-def*[*of A B C A′ B′ C′ P* α β γ] *assms*(*2*)
assms(*3*)
by *blast*
show *rk* $\{A', C', P\} = 3$
using *non-colinear-*$A'B'P$ *assms*(*1*) *desargues-config-2D-def*[*of A B C A′ B′ C′ P* α β γ] *assms*(*2*)

assms(*4*)
by *blast*
show *rk* {*B*′, *C*′, *P*} = *3*
using *non-colinear-A*′*B*′*P* *assms*(*1*) *desargues-config-2D-def*[*of A B C A*′ *B*′ *C*′ *P* α β γ] *assms*(*3*)
assms(*4*)
by *blast*
qed

lemma *rk-A*′*B*′*PQ* :
assumes *rk* {*A*, *A*′} = *2* **and** *rk* {*A*, *B*, *C*, *A*′, *B*′, *C*′} = *3* **and** *rk* {*A*, *A*′, *P*} = *2* **and**
rk {*A*, *B*, *P*} = *3* **and** *rk* {*B*, *B*′, *P*} = *2* **and** *rk* {*A*′, *P*} = *2* **and** *rk* {*B*′, *P*} = *2* **and**
rk {*A*, *B*, *C*, *A*′, *B*′, *C*′, *P*, *Q*} ≥ *4*
shows *rk* {*A*′, *B*′, *P*, *Q*} = *4*
proof−
have *card* {*A*′, *B*′, *P*, *Q*} ≤ *4*
by (*smt One-nat-def Suc-numeral card.insert card.empty finite.emptyI finite-insert insert-absorb*
insert-not-empty linear nat-add-left-cancel-le numeral-3-eq-3 numeral-Bit0 numeral-code(*3*)
numeral-le-one-iff numerals(*1*) *one-plus-numeral semiring-norm*(*4*) *semiring-norm*(*69*)
semiring-norm(*70*) *semiring-norm*(*8*))
then have *f1*:*rk* {*A*′, *B*′, *P*, *Q*} ≤ *4*
using *matroid-ax-1b dual-order.trans* **by** *blast*
have *rk* {*A*, *B*, *C*, *A*′, *B*′, *C*′, *P*, *Q*} + *rk* {*A*′, *B*′, *P*} ≤ *rk* {*A*′, *B*′, *P*, *Q*} + *rk* {*A*, *B*, *C*, *A*′, *B*′, *C*′, *P*}
using *matroid-ax-3-alt*[*of* {*A*′, *B*′, *P*} {*A*′, *B*′, *P*, *Q*} {*A*, *B*, *C*, *A*′, *B*′, *C*′, *P*}]
by (*simp add*: *insert-commute*)
then have *rk* {*A*′, *B*′, *P*, *Q*} ≥ *rk* {*A*, *B*, *C*, *A*′, *B*′, *C*′, *P*, *Q*} + *rk* {*A*′, *B*′, *P*} − *rk* {*A*, *B*, *C*, *A*′, *B*′, *C*′, *P*}
using *le-diff-conv*
by *blast*
then have *f2*:*rk* {*A*′, *B*′, *P*, *Q*} ≥ *4*
using *assms non-colinear-A*′*B*′*P coplanar-ABCA*′*B*′*C*′*P*
by (*smt diff-add-inverse2 le-trans*)
from *f1* **and** *f2* **show** *rk* {*A*′, *B*′, *P*, *Q*} = *4*
by (*simp add*: *f1 eq-iff*)
qed

lemma *desargues-config-2D-rkA*′*B*′*PQ-rkA*′*C*′*PQ-rkB*′*C*′*PQ* :
assumes *desargues-config-2D A B C A*′ *B*′ *C*′ *P* α β γ **and** *rk* {*A*′, *P*} = *2* **and** *rk* {*B*′, *P*} = *2*
and *rk* {*C*′, *P*} = *2* **and** *rk* {*A*, *B*, *C*, *A*′, *B*′, *C*′, *P*, *Q*} ≥ *4*
shows *rk* {*A*′, *B*′, *P*, *Q*} = *4* **and** *rk* {*A*′, *C*′, *P*, *Q*} = *4* **and** *rk* {*B*′, *C*′, *P*, *Q*} = *4*

proof −
  show *rk* {A', B', P, Q} = *4*
    using *rk-A′B′PQ*[*of A A′ B C B′ C′ P Q*] *assms*(*1*) *desargues-config-2D-def*[*of A B C A′ B′ C′ P α β γ*]
      assms(*2*) *assms*(*3*) *assms*(*5*)
    by *blast*
  show *rk* {A', C', P, Q} = *4*
    using *rk-A′B′PQ*[*of A A′ C B C′ B′ P Q*] *assms*(*1*) *desargues-config-2D-def*[*of A B C A′ B′ C′ P α β γ*]
      assms(*2*) *assms*(*4*) *assms*(*5*)
    by (*metis insert-commute*)
  show *rk* {B', C', P, Q} = *4*
    using *rk-A′B′PQ*[*of B B′ C A C′ A′ P Q*] *assms*(*1*) *desargues-config-2D-def*[*of A B C A′ B′ C′ P α β γ*]
      assms(*3*) *assms*(*4*) *assms*(*5*)
    by (*metis insert-commute*)
qed

lemma *rk-A′B′PR* :
  assumes *rk* {P, Q, R} = *2* **and** *rk* {P, R} = *2* **and** *rk* {A', B', P, Q} = *4*
shows *rk* {A', B', P, R} = *4*
proof −
  have *card* {A', B', P, R} ≤ *4*
    by (*smt Suc-numeral assms*(*2*) *card.empty card-insert-disjoint dual-order.trans finite.emptyI*
        finite-insert insert-absorb linear nat-add-left-cancel-le numeral-2-eq-2 numeral-3-eq-3
        numeral-Bit0 numeral-code(*3*) *numeral-le-one-iff rk-singleton rk-triple-le semiring-norm*(*2*)
      semiring-norm(*69*) *semiring-norm*(*8*))
  then have *f1*:*rk* {A', B', P, R} ≤ *4*
    using *dual-order.trans matroid-ax-1b*
    by *auto*
  have *f2*:*rk* {A', B', P, Q, R} + *rk* {P, R} ≤ *rk* {A', B', P, R} + *rk* {P, Q, R}
    using *matroid-ax-3-alt*[*of* {P, R} {A', B', P, R} {P, Q, R}]
    by (*simp add*: *insert-commute*)
  have *f3*:*rk* {A', B', P, Q, R} ≥ *4*
    using *matroid-ax-2 assms*(*3*)
    by (*metis insert-mono subset-insertI*)
  from *f2* **and** *f3* **have** *f4*:*rk* {A', B', P, R} ≥ *4*
    using *assms*(*1*) *assms*(*2*)
    by *linarith*
  thus *rk* {A', B', P, R} = *4*
    using *f1 f4*
    by (*simp add*: *f1 le-antisym*)
qed

lemma *rk-A′C′PR* :
  assumes *rk* {P, Q, R} = *2* **and** *rk* {P, R} = *2* **and** *rk* {A', C', P, Q} = *4*

shows *rk* $\{A', C', P, R\} = 4$
using *assms(1) assms(2) assms(3) rk-A′B′PR*
by *blast*

lemma *rk-B′C′PR* :
assumes *rk* $\{P, Q, R\} = 2$ **and** *rk* $\{P, R\} = 2$ **and** *rk* $\{B', C', P, Q\} = 4$
shows *rk* $\{B', C', P, R\} = 4$
using *assms(1) assms(2) assms(3) rk-A′C′PR*
by *blast*

lemma *rk-ABA′* :
assumes *rk* $\{A, B, P\} = 3$ **and** *rk* $\{A, A'\} = 2$ **and** *rk* $\{A, A', P\} = 2$
shows *rk* $\{A, B, A'\} = 3$
proof−
have *rk* $\{A, B, A', P\}$ + *rk* $\{A, A'\} \leq$ *rk* $\{A, B, A'\}$ + *rk* $\{A, A', P\}$
using *matroid-ax-3-alt[of* $\{A, A'\}$ $\{A, B, A'\}$ $\{A, A', P\}$*]*
by (*simp add*: *insert-commute*)
then have *rk* $\{A, B, A'\} \geq 3$
using *assms matroid-ax-2*
by (*smt eq-iff insert-absorb2 insert-commute non-colinear-A′B′P rk-couple*)
thus *rk* $\{A, B, A'\} = 3$
by (*simp add*: *le-antisym rk-triple-le*)
qed

lemma *desargues-config-2D-non-collinear* :
assumes *desargues-config-2D A B C A′ B′ C′ P* α β γ
shows *rk* $\{A, B, A'\} = 3$ **and** *rk* $\{A, B, B'\} = 3$ **and** *rk* $\{A, C, C'\} = 3$
proof−
show *rk* $\{A, B, A'\} = 3$
using *assms desargues-config-2D-def[of A B C A′ B′ C′ P* α β γ*] rk-ABA′*
by *auto*
show *rk* $\{A, B, B'\} = 3$
using *assms desargues-config-2D-def[of A B C A′ B′ C′ P* α β γ*] rk-ABA′*
by (*smt insert-commute*)
show *rk* $\{A, C, C'\} = 3$
using *assms desargues-config-2D-def[of A B C A′ B′ C′ P* α β γ*] rk-ABA′*
by (*smt insert-commute*)
qed

lemma *rk-Aa* :
assumes *rk* $\{A, B, P\} = 3$ **and** *rk* $\{A, A'\} = 2$ **and** *rk* $\{A, A', P\} = 2$ **and**
rk $\{Q, A', a\} = 2$
and *rk* $\{A, B, C, A', B', C', P, Q\} \geq 4$ **and** *rk* $\{A, B, C, A', B', C'\} \leq 3$
shows *rk* $\{A, a\} = 2$
proof−
have *rk* $\{Q, A', A, a\}$ + *rk* $\{a\} \leq$ *rk* $\{Q, A', a\}$ + *rk* $\{A, a\}$
using *matroid-ax-3-alt[of* $\{a\}$ $\{Q, A', a\}$ $\{A, a\}$*]*
by (*simp add*: *insert-commute*)
then have *rk* $\{Q, A', A, a\} \leq$ *rk* $\{Q, A', a\}$ + *rk* $\{A, a\}$ − *rk* $\{a\}$

using *add-le-imp-le-diff*
by *blast*
then have *rk* $\{Q, A', A, a\} \leq 2$ **if** *rk* $\{A, a\} = 1$
using *assms*(*4*)
by (*simp add*: *rk-singleton that*)
then have *rk* $\{Q, A', A\} \leq 2$ **if** *rk* $\{A, a\} = 1$
using *matroid-ax-2*
by (*metis One-nat-def assms*(*4*) *le-numeral-extra*(*4*) *nat-add-left-cancel-le numeral-2-eq-2*
numeral-3-eq-3 one-add-one rk-couple rk-triple-le that)
then have *rk* $\{Q, A', A\} = 2$ **if** *rk* $\{A, a\} = 1$
using *assms*(*2*) *matroid-ax-2*
by (*metis assms*(*4*) *numeral-eq-one-iff rk-couple semiring-norm*(*85*) *that*)
then have *rk* $\{A, A', P, Q\} = 2$ **if** *rk* $\{A, a\} = 1$
using *assms*(*3*) *matroid-ax-3-alt'*[*of* $\{A, A'\}$ *P Q*]
by (*simp add*: *assms*(*2*) *insert-commute that*)
then have *f1*:*rk* $\{A, A', B, P, Q\} \leq 3$ **if** *rk* $\{A, a\} = 1$
by (*metis One-nat-def Un-insert-right add.right-neutral add-Suc-right insert-commute matroid-ax-2-alt*
numeral-2-eq-2 numeral-3-eq-3 sup-bot.right-neutral that)
have *rk* $\{A, B, C, A', B', C', P, Q\}$ + *rk* $\{A, B, A'\} \leq$ *rk* $\{A, A', B, P, Q\}$ + *rk* $\{A, B, C, A', B', C'\}$
using *matroid-ax-3-alt*[*of* $\{A, B, A'\}$ $\{A, A', B, P, Q\}$ $\{A, B, C, A', B', C'\}$]
by (*simp add*: *insert-commute*)
then have *rk* $\{A, B, C, A', B', C', P, Q\} \leq$ *rk* $\{A, A', B, P, Q\}$ + *rk* $\{A, B, C, A', B', C'\}$ − *rk* $\{A, B, A'\}$
by *linarith*
then have *rk* $\{A, B, C, A', B', C', P, Q\} \leq 3$ **if** *rk* $\{A, a\} = 1$
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*6*) *f1 rk-ABA'*
by (*smt* ‹*rk* $\{A, B, C, A', B', C', P, Q\}$ + *rk* $\{A, B, A'\} \leq$ *rk* $\{A, A', B, P, Q\}$ + *rk* $\{A, B, C, A', B', C'\}$›
add-diff-cancel-right' add-leD2 le-less-trans not-le
ordered-cancel-comm-monoid-diff-class.add-diff-inverse
ordered-cancel-comm-monoid-diff-class.le-add-diff that)
then have $\neg$ (*rk* $\{A, a\} = 1$)
using *assms*(*5*)
by *linarith*
thus *rk* $\{A, a\} = 2$
using *rk-couple rk-singleton-bis*
by *blast*
qed

lemma *desargues-config-2D-rkAa-rkBb-rkCc* :
assumes *desargues-config-2D* $A\ B\ C\ A'\ B'\ C'\ P\ \alpha\ \beta\ \gamma$ **and** *rk* $\{A, B, C, A', B', C', P, Q\} \geq 4$
and *rk* $\{Q, A', a\} = 2$ **and** *rk* $\{Q, B', b\} = 2$ **and** *rk* $\{Q, C', c\} = 2$
shows *rk* $\{A, a\} = 2$ **and** *rk* $\{B, b\} = 2$ **and** *rk* $\{C, c\} = 2$
proof−
show *rk* $\{A, a\} = 2$

using *rk-Aa assms*(*1*) *desargues-config-2D-def*[*of A B C A′ B′ C′ P α β γ*] *assms*(*2*) *assms*(*3*)
by (*metis rk-triple-le*)
show *rk* {*B*, *b*} = *2*
using *rk-Aa assms*(*1*) *desargues-config-2D-def*[*of A B C A′ B′ C′ P α β γ*] *assms*(*2*) *assms*(*4*)
by (*smt insert-commute rk-triple-le*)
show *rk* {*C*, *c*} = *2*
using *rk-Aa*[*of C A P C′ Q c B B′ A′*] *assms*(*1*)
desargues-config-2D-def[*of A B C A′ B′ C′ P α β γ*] *assms*(*2*) *assms*(*5*)
by (*metis insert-commute rk-triple-le*)
qed

lemma *rk-ABPRa* :
assumes *rk* {*A*, *B*, *P*} = *3* **and** *rk* {*A*, *B*, *C*, *A′*, *B′*, *C′*, *P*} = *3* **and** *rk* {*P*, *Q*, *R*} = *2*
and *rk* {*P*, *R*} = *2* **and** *rk* {*A′*, *B′*, *P*, *Q*} = *4*
shows *rk* {*A*, *B*, *P*, *R*, *a*} ≥ *4*
proof−
have *rk* {*A′*, *B′*, *P*, *R*, *a*, *A*, *B*} ≥ *rk* {*A′*, *B′*, *P*, *R*}
using *matroid-ax-2*
by *auto*
then have *f1*:*rk* {*A′*, *B′*, *P*, *R*, *a*, *A*, *B*} ≥ *4*
using *rk-A′B′PR assms*(*3*) *assms*(*4*) *assms*(*5*)
by *auto*
have *f2*:*rk* {*A′*, *B′*, *A*, *B*, *P*} ≤ *3*
using *matroid-ax-2 assms*(*2*)
by (*smt insertI1 insert-subset subset-insertI*)
have *rk* {*A′*, *B′*, *P*, *R*, *a*, *A*, *B*} + *rk* {*A*, *B*, *P*} ≤ *rk* {*A*, *B*, *P*, *R*, *a*} + *rk* {*A′*, *B′*, *A*, *B*, *P*}
using *matroid-ax-3-alt*[*of* {*A*, *B*, *P*} {*A*, *B*, *P*, *R*, *a*} {*A′*, *B′*, *A*, *B*, *P*}]
by (*simp add*: *insert-commute*)
then have *rk* {*A*, *B*, *P*, *R*, *a*} ≥ *rk* {*A′*, *B′*, *P*, *R*, *a*, *A*, *B*} + *rk* {*A*, *B*, *P*} − *rk* {*A′*, *B′*, *A*, *B*, *P*}
by *linarith*
thus *rk* {*A*, *B*, *P*, *R*, *a*} ≥ *4*
using *f1 assms*(*1*) *f2*
by *linarith*
qed

lemma *rk-ABPa* :
assumes *rk* {*A*, *B*, *P*} = *3* **and** *rk* {*A*, *A′*} = *2* **and** *rk* {*A*, *A′*, *P*} = *2* **and** *rk* {*Q*, *A′*, *a*} = *2*
and *rk* {*A*, *B*, *C*, *A′*, *B′*, *C′*, *P*, *Q*} ≥ *4* **and** *rk* {*A*, *B*, *C*, *A′*, *B′*, *C′*, *P*} = *3*
and *rk* {*P*, *Q*, *R*} = *2*
and *rk* {*P*, *R*} = *2* **and** *rk* {*A′*, *B′*, *P*, *Q*} = *4* **and** *rk* {*R*, *A*, *a*} = *2*
shows *rk* {*A*, *B*, *P*, *a*} ≥ *4*
proof−
have *rk* {*A*, *B*, *C*, *A′*, *B′*, *C′*} ≤ *3*

using *matroid-ax-2 assms*(*6*)
by (*smt insert-iff subsetI*)
then have *f1*:*rk* {*A*, *a*} = *2*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *rk-Aa*
by *blast*
have *f2*:*rk* {*A*, *B*, *P*, *R*, *a*} ≥ *4*
using *assms*(*1*) *assms*(*6*) *assms*(*7*) *assms*(*8*) *assms*(*9*) *rk-ABPRa*
by *blast*
have *rk* {*A*, *B*, *P*, *R*, *a*} + *rk* {*A*, *a*} ≤ *rk* {*A*, *B*, *P*, *a*} + *rk* {*R*, *A*, *a*}
using *matroid-ax-3-alt*[*of* {*A*, *a*} {*A*, *B*, *P*, *a*} {*R*, *A*, *a*}]
by (*simp add*: *insert-commute*)
thus *rk* {*A*, *B*, *P*, *a*} ≥ *4*
using *f1 f2 assms*(*10*)
by (*smt add-le-imp-le-diff diff-add-inverse2 order-trans*)
qed

lemma *desargues-config-2D-rkABPa-rkABPb-rkABPc* :
assumes *desargues-config-2D A B C A′ B′ C′ P α β γ* **and** *rk* {*A*, *B*, *C*, *A′*, *B′*, *C′*, *P*, *Q*} ≥ *4*
and *rk* {*P*, *Q*, *R*} = *2* **and** *rk* {*P*, *R*} = *2* **and** *rk* {*A′*, *P*} = *2* **and** *rk* {*B′*, *P*} = *2* **and**
rk {*Q*, *A′*, *a*} = *2* **and** *rk* {*R*, *A*, *a*} = *2* **and** *rk* {*Q*, *B′*, *b*} = *2* **and** *rk* {*R*, *B*, *b*} = *2* **and**
rk {*Q*, *C′*, *c*} = *2* **and** *rk* {*R*, *C*, *c*} = *2*
shows *rk* {*A*, *B*, *P*, *a*} ≥ *4* **and** *rk* {*A*, *B*, *P*, *b*} ≥ *4* **and** *rk* {*A*, *B*, *P*, *c*} ≥ *4*
proof−
have *f1*:*rk* {*A*, *B*, *C*, *A′*, *B′*, *C′*, *P*} = *3*
using *assms*(*1*) *desargues-config-2D-def*[*of A B C A′ B′ C′ P α β γ*] *coplanar-ABCA′B′C′P*
by *auto*
have *f2*:*rk* {*A′*, *B′*, *P*, *Q*} = *4*
using *assms*(*1*) *desargues-config-2D-def*[*of A B C A′ B′ C′ P α β γ*] *assms*(*2*) *assms*(*5*) *assms*(*6*)
rk-A′B′PQ[*of A A′ B C B′ C′ P Q*]
by *auto*
show *rk* {*A*, *B*, *P*, *a*} ≥ *4*
using *f1 f2 assms*(*1*) *desargues-config-2D-def*[*of A B C A′ B′ C′ P α β γ*] *assms*(*7*) *assms*(*2*) *assms*(*3*)
assms(*4*) *assms*(*8*) *rk-ABPa*
by *auto*
show *rk* {*A*, *B*, *P*, *b*} ≥ *4*
using *f1 f2 assms*(*1*) *desargues-config-2D-def*[*of A B C A′ B′ C′ P α β γ*] *assms*(*9*) *assms*(*2*) *assms*(*3*)
assms(*4*) *assms*(*10*) *rk-ABPa*[*of B A P B′ Q b C A′ C′ R*]
by (*metis insert-commute*)
show *rk* {*A*, *B*, *P*, *c*} ≥ *4*
proof−
have *f3*:*rk* {*A*, *B*, *P*, *R*, *c*} ≥ *4*
using *rk-ABPRa*[*of A B P C A′ B′ C′ Q R c*] *assms*(*1*) *desargues-config-2D-def*[*of*

A B C A′ B′ C′ P α β γ]
f1 assms(*3*) *assms*(*4*) *f2*
by *auto*
have {*A*, *B*, *P*, *R*, *c*} ⊆ {*A*, *B*, *C*, *P*, *R*, *c*}
by *auto*
then have *f4*:*rk* {*A*, *B*, *C*, *P*, *R*, *c*} ≥ *4*
using *matroid-ax-2 f3*
by (*meson dual-order.trans*)
have *rk* {*A*, *B*, *C*, *P*, *R*, *c*} + *rk* {*C*, *c*} ≤ *rk* {*A*, *B*, *C*, *P*, *c*} + *rk* {*R*, *C*, *c*}
using *matroid-ax-3-alt*[*of* {*C*,*c*} {*A*, *B*, *C*, *P*, *c*} {*R*, *C*, *c*}]
by (*simp add*: *insert-commute*)
then have *f5*:*rk* {*A*, *B*, *C*, *P*, *c*} ≥ *4*
using *f4 assms*(*12*) *desargues-config-2D-rkAa-rkBb-rkCc assms*(*1*) *assms*(*9*) *assms*(*11*) *assms*(*2*) *assms*(*7*)
by *auto*
have *f6*:*rk* {*A*, *B*, *C*, *P*} ≤ *3*
using *matroid-ax-2 f1*
by (*smt insert-iff subsetI*)
have *rk* {*A*, *B*, *C*, *P*, *c*} + *rk* {*A*, *B*, *P*} ≤ *rk* {*A*, *B*, *P*, *c*} + *rk* {*A*, *B*, *C*, *P*}
using *matroid-ax-3-alt*[*of* {*A*, *B*, *P*} {*A*, *B*, *P*, *c*} {*A*, *B*, *C*, *P*}]
by (*simp add*: *insert-commute*)
then have *rk* {*A*, *B*, *P*, *c*} ≥ *rk* {*A*, *B*, *C*, *P*, *c*}
using *assms*(*1*) *desargues-config-2D-def*[*of A B C A′ B′ C′ P α β γ*] *f6*
by *linarith*
thus *rk* {*A*, *B*, *P*, *c*} ≥ *4*
using *f5*
by *linarith*
qed
qed

lemma *rk-AA′C* :
assumes *rk* {*A*, *C*, *P*} = *3* **and** *rk* {*A*, *A′*} = *2* **and** *rk* {*A*, *A′*, *P*} = *2*
shows *rk* {*A*, *A′*, *C*} ≥ *3*
proof−
have *f1*:*rk* {*A*, *C*, *A′*, *P*} ≥ *3*
using *assms*(*1*) *matroid-ax-2*
by (*metis insert-commute subset-insertI*)
have *rk* {*A*, *C*, *A′*, *P*} + *rk* {*A*, *A′*} ≤ *rk* {*A*, *A′*, *C*} + *rk* {*A*, *A′*, *P*}
using *matroid-ax-3-alt*[*of* {*A*, *A′*} {*A*, *A′*, *C*} {*A*, *A′*, *P*}]
by (*simp add*: *insert-commute*)
thus *rk* {*A*, *A′*, *C*} ≥ *3*
using *f1 assms*(*2*) *assms*(*3*)
by *linarith*
qed

lemma *rk-AA′C′* :
assumes *rk* {*A′*, *C′*, *P*} = *3* **and** *rk* {*A*, *A′*} = *2* **and** *rk* {*A*, *A′*, *P*} = *2*

shows *rk* {*A*, *A*′, *C*′} ≥ *3*
by (*smt assms*(*1*) *assms*(*2*) *assms*(*3*) *insert-commute rk-AA*′*C*)

lemma *rk-AA*′*Ca* :
assumes *rk* {*A*, *A*′, *C*} ≥ *3* **and** *rk* {*A*, *B*, *P*, *a*} ≥ *4* **and** *rk* {*A*, *B*, *C*, *A*′, *B*′, *C*′, *P*} = *3*
shows *rk* {*A*, *A*′, *C*, *a*} ≥ *4*
proof−
have *f1*:*rk* {*A*, *A*′, *C*, *a*, *B*, *P*} ≥ *4*
using *assms*(*2*) *matroid-ax-2*
by (*smt dual-order.trans insert-commute insert-mono insert-subset subset-insertI*)
have *f2*:*rk* {*A*, *B*, *C*, *P*, *A*′} ≤ *3*
using *assms*(*3*) *matroid-ax-2*
by (*smt empty-subsetI insert-commute insert-mono semiring-norm*(*3*))
have *rk* {*A*, *A*′, *C*, *a*, *B*, *P*} + *rk* {*A*, *A*′, *C*} ≤ *rk* {*A*, *A*′, *C*, *a*} + *rk* {*A*, *B*, *C*, *P*, *A*′}
using *matroid-ax-3-alt*[*of* {*A*, *A*′, *C*} {*A*, *A*′, *C*, *a*} {*A*, *B*, *C*, *P*, *A*′}]
by (*simp add*: *insert-commute*)
then have *rk* {*A*, *A*′, *C*, *a*} ≥ *rk* {*A*, *A*′, *C*, *a*, *B*, *P*} + *rk* {*A*, *A*′, *C*} − *rk* {*A*, *B*, *C*, *P*, *A*′}
using *le-diff-conv*
by *blast*
thus *rk* {*A*, *A*′, *C*, *a*} ≥ *4*
using *f1 assms*(*1*) *f2*
by *linarith*
qed

lemma *rk-AA*′*C*′*a* :
assumes *rk* {*A*, *A*′, *C*′} ≥ *3* **and** *rk* {*A*, *B*, *P*, *a*} ≥ *4* **and** *rk* {*A*, *B*, *C*, *A*′, *B*′, *C*′, *P*} = *3*
shows *rk* {*A*, *A*′, *C*′, *a*} ≥ *4*
by (*smt assms*(*1*) *assms*(*2*) *assms*(*3*) *insert-commute rk-AA*′*Ca*)

lemma *rk-Ra* :
assumes *rk* {*Q*, *A*′, *a*} = *2* **and** *rk* {*P*, *Q*, *R*} = *2* **and** *rk* {*R*, *Q*} = *2* **and** *rk* {*A*, *A*′, *P*} = *2*
and *rk* {*A*′, *P*} = *2* **and** *rk* {*A*, *B*, *C*, *A*′, *B*′, *C*′, *P*} = *3* **and** *rk* {*A*, *B*, *A*′} = *3* **and**
rk {*A*, *B*, *C*, *A*′, *B*′, *C*′, *P*, *Q*} ≥ *4*
shows *rk* {*R*, *a*} = *2*
proof−
have *R* = *a* **if** *rk* {*R*, *a*} = *1*
using *rk-couple-to-singleton*
by (*simp add*: *that*)
then have *rk* {*R*, *Q*, *A*′} = *2* **if** *rk* {*R*, *a*} = *1*
using *assms*(*1*)
by (*simp add*: *insert-commute that*)
then have *f1*:*rk* {*P*, *Q*, *R*, *A*′} = *2* **if** *rk* {*R*, *a*} = *1*
using *assms*(*2*) *assms*(*3*) *matroid-ax-3-alt*′

by (*metis Un-empty-right Un-insert-right insert-commute that*)
have *rk* $\{P, Q, R, A', A\}$ + *rk* $\{A', P\} \leq$ *rk* $\{A, A', P\}$ + *rk* $\{P, Q, R, A'\}$
using *matroid-ax-3-alt*[*of* $\{A', P\}$ $\{A, A', P\}$ $\{P, Q, R, A'\}$]
by (*simp add*: *insert-commute*)
then have *rk* $\{P, Q, R, A', A\} = 2$ **if** *rk* $\{R, a\} = 1$
using *assms*(*4*) *f1* *assms*(*5*)
by (*metis Un-insert-right add-le-cancel-right insert-is-Un le-antisym matroid-ax-2 subset-insertI that*)
then have *f2*:*rk* $\{P, Q, R, A', A, B\} \leq 3$ **if** *rk* $\{R, a\} = 1$
using *matroid-ax-2-alt*[*of* $\{P, Q, R, A', A\}$ *B*]
by (*simp add*: *insert-commute that*)
have *f3*:*rk* $\{A, B, A', P\} \geq 3$
using *assms*(*7*) *matroid-ax-2*
by (*metis insert-commute subset-insertI*)
have *rk* $\{P, Q, R, A', A, B, C, B', C'\}$ + *rk* $\{A, B, A', P\} \leq$ *rk* $\{P, Q, R, A', A, B\}$ + *rk* $\{A, B, C, A', B', C', P\}$
using *matroid-ax-3-alt*[*of* $\{A, B, A', P\}$ $\{P, Q, R, A', A, B\}$ $\{A, B, C, A', B', C', P\}$]
by (*simp add*: *insert-commute*)
then have *f4*:*rk* $\{P, Q, R, A', A, B, C, B', C'\} \leq 3$ **if** *rk* $\{R, a\} = 1$
using *f2 f3 assms*(*6*) *that*
by *linarith*
have *f5*:*rk* $\{P, Q, R, A', A, B, C, B', C'\} \geq$ *rk* $\{A, B, C, A', B', C', P, Q\}$
using *matroid-ax-2*
by *auto*
thus *rk* $\{R, a\} = 2$ **using** *f4 f5 assms*(*8*)
by (*smt Suc-1 Suc-le-eq add-Suc add-Suc-right nat-1-add-1 not-le numeral-2-eq-2 numeral-3-eq-3*
numeral-Bit0 order.trans rk-couple rk-singleton-bis)
qed

lemma *desargues-config-2D-rkRa-rkRb-rkRc* :
assumes *desargues-config-2D* $A\ B\ C\ A'\ B'\ C'\ P\ \alpha\ \beta\ \gamma$ **and** *rk* $\{A, B, C, A', B', C', P, Q\} \geq 4$
and *rk* $\{P, Q, R\} = 2$ **and** *rk* $\{Q, R\} = 2$ **and** *rk* $\{Q, A', a\} = 2$ **and** *rk* $\{Q, B', b\} = 2$ **and**
rk $\{Q, C', c\} = 2$ **and** *rk* $\{A', P\} = 2$ **and** *rk* $\{B', P\} = 2$ **and** *rk* $\{C', P\} = 2$
shows *rk* $\{R, a\} = 2$ **and** *rk* $\{R, b\} = 2$ **and** *rk* $\{R, c\} = 2$
proof−
have *f1*:*rk* $\{A, B, C, A', B', C', P\} = 3$
using *coplanar-ABCA'B'C'P* *assms*(*1*) *desargues-config-2D-def*[*of* $A\ B\ C\ A'\ B'\ C'\ P\ \alpha\ \beta\ \gamma$]
by *blast*
have *f2*:*rk* $\{A, B, A'\} = 3$
using *desargues-config-2D-non-collinear assms*(*1*)
by *auto*
have *f3*:*rk* $\{A, B, B'\} = 3$
using *desargues-config-2D-non-collinear assms*(*1*)
by *auto*

have *f4*:*rk* {*A*, *C*, *C'*} = *3*
using *desargues-config-2D-non-collinear assms*(*1*)
by *auto*
show *rk* {*R*, *a*} = *2*
using *f1 f2 rk-Ra*[*of Q A' a P R A B C B' C'*] *assms*(*1*) *desargues-config-2D-def*[*of A B C A' B' C' P α β γ*]
assms(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *assms*(*8*)
by (*metis insert-commute*)
show *rk* {*R*, *b*} = *2*
using *f1 f3 rk-Ra*[*of Q B' b P R B A C A' C'*] *assms*(*1*) *desargues-config-2D-def*[*of A B C A' B' C' P α β γ*]
assms(*2*) *assms*(*3*) *assms*(*4*) *assms*(*6*) *assms*(*9*)
by (*metis insert-commute*)
show *rk* {*R*, *c*} = *2*
using *f1 f4 rk-Ra*[*of Q C' c P R C A B A' B'*] *assms*(*1*) *desargues-config-2D-def*[*of A B C A' B' C' P α β γ*]
assms(*2*) *assms*(*3*) *assms*(*4*) *assms*(*7*) *assms*(*10*)
by (*metis insert-commute*)
qed

lemma *rk-acACβ* :
assumes *rk* {*R*, *A*, *a*} = *2* **and** *rk* {*R*, *C*, *c*} = *2* **and** *rk* {*A*, *C*} = *2* **and** *rk* {*A*, *C*, *β*} = *2*
and *rk* {*Q*, *A'*, *a*} = *2* **and** *rk* {*A*, *A'*, *C*, *a*} ≥ *4*
shows *rk* {*a*, *c*, *A*, *C*, *β*} = *3*
proof−
have *rk* {*a*, *c*, *A*, *C*, *R*} + *rk* {*R*} ≤ *rk* {*R*, *A*, *a*} + *rk* {*R*, *C*, *c*}
using *matroid-ax-3-alt*[*of* {*R*} {*R*, *A*, *a*} {*R*, *C*, *c*}]
by (*simp add*: *insert-commute*)
then have *f1*:*rk* {*a*, *c*, *A*, *C*, *R*} ≤ *3*
using *assms*(*1*) *assms*(*2*)
by (*metis Suc-1 Suc-eq-plus1 Suc-le-mono add-Suc-right numeral-3-eq-3 numeral-nat*(*1*) *numerals*(*1*)
rk-singleton)
have *rk* {*a*, *c*, *A*, *C*, *R*, *β*} + *rk* {*A*, *C*} ≤ *rk* {*a*, *c*, *A*, *C*, *R*} + *rk* {*A*, *C*, *β*}
using *matroid-ax-3-alt*[*of* {*A*, *C*} {*a*, *c*, *A*, *C*, *R*} {*A*, *C*, *β*}]
by (*simp add*: *insert-commute*)
then have *f2*:*rk* {*a*, *c*, *A*, *C*, *R*, *β*} ≤ *3*
using *f1 assms*(*3*) *assms*(*4*)
by *linarith*
have {*a*, *c*, *A*, *C*, *β*} ⊆ {*a*, *c*, *A*, *C*, *R*, *β*}
by *auto*
then have *f3*:*rk* {*a*, *c*, *A*, *C*, *β*} ≤ *3*
using *matroid-ax-2 f2*
by (*meson order-trans*)
have *f4*:*rk* {*A*, *A'*, *C*, *a*, *c*, *Q*} ≥ *4*
using *matroid-ax-2 assms*(*6*)
by (*smt dual-order.trans insert-commute insert-mono insert-subset subset-insertI*)
have *rk* {*A*, *A'*, *C*, *a*, *c*, *Q*} + *rk* {*a*} ≤ *rk* {*a*, *c*, *A*, *C*} + *rk* {*Q*, *A'*, *a*}

using *matroid-ax-3-alt*[*of* $\{a\}$ $\{a, c, A, C\}$ $\{Q, A', a\}$]
by (*simp add*: *insert-commute*)
then have *rk* $\{a, c, A, C\} \geq$ *rk* $\{A, A', C, a, c, Q\}$ + *rk* $\{a\}$ − *rk* $\{Q, A', a\}$
using *le-diff-conv*
by *blast*
then have *rk* $\{a, c, A, C\} \geq 3$
using *f4 assms*(*5*)
by (*smt One-nat-def* ‹*rk* $\{A, A', C, a, c, Q\}$ + *rk* $\{a\} \leq$ *rk* $\{a, c, A, C\}$ + *rk* $\{Q, A', a\}$›
ab-semigroup-add-class.add-ac(*1*) *add-2-eq-Suc′ add-diff-cancel-right′ add-le-imp-le-diff*

card.empty card.insert dual-order.antisym finite.intros(*1*) *insert-absorb insert-not-empty*
matroid-ax-1b nat-1-add-1 numeral-3-eq-3 numeral-Bit0 order.trans rk-ax-singleton)
then have *rk* $\{a, c, A, C, \beta\} \geq 3$
using *matroid-ax-2*
by (*metis Un-insert-right Un-upper2 dual-order.trans sup-bot.comm-neutral*)
thus *rk* $\{a, c, A, C, \beta\} = 3$
using *f3 le-antisym*
by *blast*
qed

lemma *rk-acA′C′β* :
assumes *rk* $\{Q, A', a\} = 2$ **and** *rk* $\{Q, C', c\} = 2$ **and** *rk* $\{A', C'\} = 2$ **and** *rk* $\{A', C', \beta\} = 2$
and *rk* $\{R, A, a\} = 2$ **and** *rk* $\{A', A, C', a\} \geq 4$
shows *rk* $\{a, c, A', C', \beta\} = 3$
using *assms rk-acACβ*
by *blast*

lemma *plane-representation-change* :
assumes *rk* $\{A, B, C, P\} = 3$ **and** *rk* $\{B, C, P\} = 3$ **and** *rk* $\{A, B, C, Q\} = 4$
shows *rk* $\{P, B, C, Q\} = 4$
proof−
have *rk* $\{P, B, C, Q\} \leq 4$ **using** *assms*(*2*) *matroid-ax-2-alt*[*of* $\{B, C, P\}$ *Q*]
by (*simp add*: *insert-commute*)
have *rk* $\{A, B, C, Q, P\}$ + *rk* $\{B, C, P\} \leq$ *rk* $\{P, B, C, Q\}$ + *rk* $\{A, B, C, P\}$
using *matroid-ax-3-alt*[*of* $\{B, C, P\}$ $\{P, B, C, Q\}$ $\{A, B, C, P\}$]
by (*simp add*: *insert-commute*)
then have *rk* $\{P, B, C, Q\} \geq 4$
using *assms*
by (*smt add.commute dual-order.trans insert-commute matroid-ax-2 nat-add-left-cancel-le*

subset-insertI)
thus *rk* $\{P, B, C, Q\} = 4$
by (*simp add*: ‹*rk* $\{P, B, C, Q\} \leq 4$› *antisym*)
qed

lemma *desargues-config-2D-rkABCP* :
assumes *desargues-config-2D A B C A′ B′ C′ P α β γ*
shows *rk* {*A*, *B*, *C*, *P*} = *3*
proof−
have *rk* {*A*, *B*, *C*} = *3*
using *assms desargues-config-2D-def*[*of A B C A′ B′ C′ P α β γ*]
by *auto*
then have *f1*:*rk* {*A*, *B*, *C*, *P*} ≥ *3*
using *matroid-ax-2*
by (*metis empty-subsetI insert-mono*)
have *f2*:*rk* {*A*, *B*, *C*, *A′*, *B′*, *C′*, *P*} = *3*
using *assms desargues-config-2D-def*[*of A B C A′ B′ C′ P*] *coplanar-ABCA′B′C′P*

by *auto*
have {*A*, *B*, *C*, *P*} ⊆ {*A*, *B*, *C*, *A′*, *B′*, *C′*, *P*}
by *auto*
then have *rk* {*A*, *B*, *C*, *P*} ≤ *3*
using *matroid-ax-2 f2*
by *metis*
thus *rk* {*A*, *B*, *C*, *P*} = *3*
using *f1 antisym*
by *blast*
qed

lemma *desargues-config-2D-rkABCabc* :
assumes *desargues-config-2D A B C A′ B′ C′ P α β γ* **and** *rk* {*A*, *B*, *C*, *A′*, *B′*, *C′*, *P*, *Q*} ≥ *4*
and *rk* {*Q*, *A′*, *a*} = *2* **and** *rk* {*P*, *Q*, *R*} = *2* **and** *rk* {*P*, *R*} = *2* **and** *rk* {*R*, *A*, *a*} = *2* **and**
rk {*A′*, *P*} = *2* **and** *rk* {*B′*, *P*} = *2*
shows *rk* {*A*, *B*, *C*, *a*, *b*, *c*} ≥ *4*
proof−
have *f1*:*rk* {*A*, *B*, *C*, *A′*, *B′*, *C′*, *P*} = *3*
using *assms*(*1*) *desargues-config-2D-def*[*of A B C A′ B′ C′ P α β γ*] *coplanar-ABCA′B′C′P*
by *auto*
have *f2*:*rk* {*A′*, *B′*, *P*, *Q*} = *4*
using *rk-A′B′PQ*[*of A A′ B C B′ C′ P Q*] *assms*(*1*) *desargues-config-2D-def*[*of A B C A′ B′ C′ P α β γ*]
assms(*2*) *assms*(*7*) *assms*(*8*)
by *auto*
from *f1* **and** *f2* **have** *f3*:*rk* {*A*, *B*, *P*, *a*} ≥ *4*
using *assms*(*1*) *desargues-config-2D-def*[*of A B C A′ B′ C′ P α β γ*] *assms*(*2*) *assms*(*3*) *assms*(*4*)
assms(*5*) *assms*(*6*) *rk-ABPa*
by *auto*
have {*A*, *B*, *P*, *a*} ⊆ {*A*, *B*, *C*, *a*, *b*, *c*, *P*}
by *auto*

then have *f4*:*rk* {*A*, *B*, *C*, *a*, *b*, *c*, *P*} ≥ *4*
using *matroid-ax-2 f3*
by (*meson dual-order.trans*)
have *f5*:*rk* {*A*, *B*, *C*, *P*} = *3*
using *assms*(*1*) *desargues-config-2D-rkABCP*
by *auto*
have *rk* {*A*, *B*, *C*, *a*, *b*, *c*, *P*} + *rk* {*A*, *B*, *C*} ≤ *rk* {*A*, *B*, *C*, *a*, *b*, *c*} + *rk* {*A*, *B*, *C*, *P*}
using *matroid-ax-3-alt*[*of* {*A*, *B*, *C*} {*A*, *B*, *C*, *a*, *b*, *c*} {*A*, *B*, *C*, *P*}]
by (*simp add*: *insert-commute*)
thus *rk* {*A*, *B*, *C*, *a*, *b*, *c*} ≥ *4*
using *f4 assms*(*1*) *desargues-config-2D-def*[*of A B C A′ B′ C′ P α β γ*] *f5*
by *linarith*
qed

lemma *rk-abc* :
assumes *desargues-config-2D A B C A′ B′ C′ P α β γ* **and** *rk* {*A*, *B*, *C*, *A′*, *B′*, *C′*, *P*, *Q*} ≥ *4*
and *rk* {*Q*, *A′*, *a*} = *2* **and** *rk* {*Q*, *B′*, *b*} = *2* **and** *rk* {*Q*, *C′*, *c*} = *2* **and** *rk* {*P*, *Q*, *R*} = *2* **and**
rk {*P*, *R*} = *2* **and** *rk* {*Q*, *R*} = *2* **and** *rk* {*R*, *A*, *a*} = *2* **and** *rk* {*R*, *B*, *b*} = *2* **and**
rk {*R*, *C*, *c*} = *2* **and** *rk* {*A′*, *P*} = *2* **and** *rk* {*B′*, *P*} = *2* **and** *rk* {*C′*, *P*} = *2*
shows *rk* {*a*, *b*, *c*} = *3*
proof−
have *rk* {*a*, *b*, *c*} ≤ *3*
by (*simp add*: *rk-triple-le*)
have *rk* {*A*, *B*, *C*, *a*, *b*, *c*} ≥ *4*
using *desargues-config-2D-rkABCabc assms*(*1*) *assms*(*13*) *assms*(*2*) *assms*(*3*) *assms*(*6*) *assms*(*7*) *assms*(*9*)
assms(*12*)
by *auto*
then have *f1*:*rk* {*A*, *B*, *C*, *R*, *a*, *b*, *c*} ≥ *4*
using *matroid-ax-2*
by (*smt dual-order.trans insert-commute subset-insertI*)
have *f2*:*rk* {*A*, *B*, *C*, *A′*, *B′*, *C′*, *P*} = *3*
using *coplanar-ABCA′B′C′P assms*(*1*) *desargues-config-2D-def*[*of A B C A′ B′ C′ P α β γ*]
by *auto*
have *f3*:*rk* {*A*, *C*, *C′*} = *3*
using *assms*(*1*) *desargues-config-2D-non-collinear*(*3*)
by *auto*
from *f2* **and** *f3* **have** *f4*:*rk* {*R*, *c*} = *2*
using *assms*(*1*) *desargues-config-2D-def*[*of A B C A′ B′ C′ P α β γ*] *assms*(*2*) *assms*(*5*) *assms*(*6*)
assms(*8*) *assms*(*14*) *rk-Ra*[*of Q C′ c P R C A B A′ B′*]
by (*metis insert-commute*)
have *rk* {*A*, *B*, *C*, *R*, *a*, *b*, *c*} + *rk* {*R*, *c*} ≤ *rk* {*a*, *b*, *c*, *R*, *A*, *B*} + *rk* {*R*, *C*, *c*}

using *matroid-ax-3-alt*[*of* $\{R, c\}$ $\{a, b, c, R, A, B\}$ $\{R, C, c\}$]
by (*simp add*: *insert-commute*)
then have *f5*:*rk* $\{a, b, c, R, A, B\} \geq 4$
using *f1 f4 assms*(*11*)
by *linarith*
have *rk* $\{A, B, B'\} = 3$
using *assms*(*1*) *desargues-config-2D-non-collinear*(*2*)
by *auto*
then have *f6*:*rk* $\{R, b\} = 2$
using *f2 assms*(*1*) *desargues-config-2D-def*[*of* $A\ B\ C\ A'\ B'\ C'\ P\ \alpha\ \beta\ \gamma$]
rk-Ra[*of* $Q\ B'\ b\ P\ R\ B\ A\ C\ A'\ C'$] *assms*(*2*) *assms*(*4*) *assms*(*6*) *assms*(*8*) *assms*(*13*)
by (*metis insert-commute*)
have *rk* $\{a, b, c, R, A, B\}$ + *rk* $\{R, b\} \leq$ *rk* $\{a, b, c, R, A\}$ + *rk* $\{R, B, b\}$
using *matroid-ax-3-alt*[*of* $\{R, b\}$ $\{a, b, c, R, A\}$ $\{R, B, b\}$]
by (*simp add*: *insert-commute*)
then have *f7*:*rk* $\{a, b, c, R, A\} \geq 4$
using *f5 f6 assms*(*10*)
by *linarith*
have *rk* $\{a, b, c, R, A\}$ + *rk* $\{a\} \leq$ *rk* $\{a, b, c\}$ + *rk* $\{R, A, a\}$
using *matroid-ax-3-alt*[*of* $\{a\}$ $\{a, b, c\}$ $\{R, A, a\}$]
by (*simp add*: *insert-commute*)
then have *rk* $\{a, b, c\} \geq 3$
using *f7 assms*(*9*)
by (*smt One-nat-def Suc-eq-plus1 Suc-le-mono Suc-numeral add.assoc card.empty card.insert*
dual-order.trans finite.intros(*1*) *insert-absorb insert-not-empty le-antisym matroid-ax-1b*
one-add-one rk-ax-singleton semiring-norm(*2*) *semiring-norm*(*8*))
thus *rk* $\{a, b, c\} = 3$
by (*simp add*: ‹*rk* $\{a, b, c\} \leq 3$› *le-antisym*)
qed

lemma *rk-acβ* :
assumes *desargues-config-2D* $A\ B\ C\ A'\ B'\ C'\ P\ \alpha\ \beta\ \gamma$ **and** *rk* $\{A, B, C, A', B', C', P, Q\} \geq 4$
and *rk* $\{Q, A', a\} = 2$ **and** *rk* $\{Q, B', b\} = 2$ **and** *rk* $\{Q, C', c\} = 2$ **and** *rk* $\{P, Q, R\} = 2$ **and**
rk $\{P, R\} = 2$ **and** *rk* $\{Q, R\} = 2$ **and** *rk* $\{R, A, a\} = 2$ **and** *rk* $\{R, B, b\} = 2$ **and**
rk $\{R, C, c\} = 2$ **and** *rk* $\{A', P\} = 2$ **and** *rk* $\{B', P\} = 2$ **and** *rk* $\{C', P\} = 2$
shows *rk* $\{a, c, \beta\} = 2$
proof−
have *rk* $\{a, b, c\} = 3$
using *rk-abc assms*
by *auto*
then have *rk* $\{a, c\} = 2$
by (*metis insert-commute rk-triple-to-rk-couple*)
then have *rk* $\{a, c, \beta\} \geq 2$

using *matroid-ax-2*
by (*metis empty-subsetI insert-mono*)
have *f1*:*rk* {*a*, *c*, *A*, *C*, *A*′, *C*′, *β*} + *rk* {*a*, *c*, *β*} ≤ *rk* {*a*, *c*, *A*, *C*, *β*} + *rk* {*a*, *c*, *A*′, *C*′, *β*}
using *matroid-ax-3-alt*[*of* {*a*, *c*, *β*} {*a*, *c*, *A*, *C*, *β*} {*a*, *c*, *A*′, *C*′, *β*}]
by (*simp add*: *insert-commute*)
have *f2*:*rk* {*A*, *A*′, *C*} ≥ *3*
using *rk-AA*′*C assms*(*1*) *desargues-config-2D-def*[*of A B C A*′ *B*′ *C*′ *P α β γ*]
by *auto*
have *f3*:*rk* {*A*, *B*, *C*, *A*′, *B*′, *C*′, *P*} = *3*
using *coplanar-ABCA*′*B*′*C*′*P assms*(*1*) *desargues-config-2D-def*[*of A B C A*′ *B*′ *C*′ *P α β γ*]
by *auto*
then have *f4*:*rk* {*A*, *B*, *P*, *a*} ≥ *4*
using *rk-ABPa assms*(*1*) *desargues-config-2D-def*[*of A B C A*′ *B*′ *C*′ *P α β γ*]
by (*meson assms*(*12*) *assms*(*13*) *assms*(*14*) *assms*(*2*) *assms*(*3*) *assms*(*6*) *assms*(*7*) *assms*(*9*)
desargues-config-2D-rkA′*B*′*PQ-rkA*′*C*′*PQ-rkB*′*C*′*PQ*(*1*))
have *rk* {*A*, *A*′, *C*, *a*} ≥ *4*
using *f2 f3 f4 rk-AA*′*Ca*[*of A A*′ *C B P a B*′ *C*′]
by *auto*
then have *f5*:*rk* {*a*, *c*, *A*, *C*, *β*} = *3*
using *rk-acACβ*[*of R A a C c β Q A*′] *assms*(*1*) *desargues-config-2D-def*[*of A B C A*′ *B*′ *C*′ *P α β γ*]
assms(*9*) *assms*(*11*) *assms*(*3*) *rk-triple-to-rk-couple*
by *blast*
have *rk* {*A*′, *A*, *C*′, *a*} ≥ *4*
using *rk-AA*′*C*′*a*[*of A A*′ *C*′ *B P a C B*′] *assms*(*1*) *desargues-config-2D-def*[*of A B C A*′ *B*′ *C*′ *P α β γ*]
by (*smt assms*(*12*) *assms*(*13*) *assms*(*14*) *desargues-config-2D-non-collinear-P*(*2*) *f3 f4 insert-commute*
rk-AA′*C*)
then have *f6*:*rk* {*a*, *c*, *A*′, *C*′, *β*} = *3*
using *rk-acA*′*C*′*β*[*of Q A*′ *a C*′ *c β R A*] *assms*(*1*) *desargues-config-2D-def*[*of A B C A*′ *B*′ *C*′ *P α β γ*]
assms(*3*) *assms*(*5*) *assms*(*9*)
by (*metis* (*full-types*) *insert-commute rk-triple-to-rk-couple*)
have {*A*, *A*′, *C*, *a*} ⊆ {*a*, *c*, *A*, *C*, *A*′, *C*′, *β*}
by *auto*
then have *f7*:*rk* {*a*, *c*, *A*, *C*, *A*′, *C*′, *β*} ≥ *4*
using *matroid-ax-2*
by (*meson* ‹*4* ≤ *rk* {*A*, *A*′, *C*, *a*}› *dual-order.trans*)
then have *rk* {*a*, *c*, *β*} ≤ *2*
using *f1 f5 f6*
by *linarith*
thus *rk* {*a*, *c*, *β*} = *2*
using ‹*2* ≤ *rk* {*a*, *c*, *β*}› *le-antisym*
by *blast*
qed

lemma *rk-abγ* :
assumes *desargues-config-2D A B C A′ B′ C′ P α β γ* **and** *rk* {*A*, *B*, *C*, *A′*, *B′*, *C′*, *P*, *Q*} ≥ *4*
and *rk* {*Q*, *A′*, *a*} = *2* **and** *rk* {*Q*, *B′*, *b*} = *2* **and** *rk* {*Q*, *C′*, *c*} = *2* **and** *rk* {*P*, *Q*, *R*} = *2* **and**
rk {*P*, *R*} = *2* **and** *rk* {*Q*, *R*} = *2* **and** *rk* {*R*, *A*, *a*} = *2* **and** *rk* {*R*, *B*, *b*} = *2* **and**
rk {*R*, *C*, *c*} = *2* **and** *rk* {*A′*, *P*} = *2* **and** *rk* {*B′*, *P*} = *2* **and** *rk* {*C′*, *P*} = *2*
shows *rk* {*a*, *b*, *γ*} = *2*
proof−
have *desargues-config-2D A C B A′ C′ B′ P α γ β*
using *assms*(*1*) *desargues-config-2D-def*[*of A B C A′ B′ C′ P α β γ*]
desargues-config-2D-def[*of A C B A′ C′ B′ P α γ β*]
by (*metis insert-commute*)
thus *rk* {*a*, *b*, *γ*} = *2*
using *rk-acβ*[*of A C B A′ C′ B′ P α γ β Q a c b R*]
by (*metis assms*(*10*) *assms*(*11*) *assms*(*12*) *assms*(*13*) *assms*(*14*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*)
assms(*6*) *assms*(*7*) *assms*(*8*) *assms*(*9*) *insert-commute*)
qed

lemma *rk-bcα* :
assumes *desargues-config-2D A B C A′ B′ C′ P α β γ* **and** *rk* {*A*, *B*, *C*, *A′*, *B′*, *C′*, *P*, *Q*} ≥ *4*
and *rk* {*Q*, *A′*, *a*} = *2* **and** *rk* {*Q*, *B′*, *b*} = *2* **and** *rk* {*Q*, *C′*, *c*} = *2* **and** *rk* {*P*, *Q*, *R*} = *2* **and**
rk {*P*, *R*} = *2* **and** *rk* {*Q*, *R*} = *2* **and** *rk* {*R*, *A*, *a*} = *2* **and** *rk* {*R*, *B*, *b*} = *2* **and**
rk {*R*, *C*, *c*} = *2* **and** *rk* {*A′*, *P*} = *2* **and** *rk* {*B′*, *P*} = *2* **and** *rk* {*C′*, *P*} = *2*
shows *rk* {*b*, *c*, *α*} = *2*
proof−
have *desargues-config-2D B A C B′ A′ C′ P β α γ*
using *assms*(*1*) *desargues-config-2D-def*[*of A B C A′ B′ C′ P α β γ*]
desargues-config-2D-def[*of B A C B′ A′ C′ P β α γ*]
by (*metis insert-commute*)
thus *rk* {*b*, *c*, *α*} = *2*
using *rk-acβ*[*of B A C B′ A′ C′ P β α γ Q b a c R*]
by (*metis assms*(*10*) *assms*(*11*) *assms*(*12*) *assms*(*13*) *assms*(*14*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*)
assms(*6*) *assms*(*7*) *assms*(*8*) *assms*(*9*) *insert-commute*)
qed

lemma *rk-abcαβγ* :
assumes *desargues-config-2D A B C A′ B′ C′ P α β γ* **and** *rk* {*A*, *B*, *C*, *A′*, *B′*, *C′*, *P*, *Q*} ≥ *4*
and *rk* {*Q*, *A′*, *a*} = *2* **and** *rk* {*Q*, *B′*, *b*} = *2* **and** *rk* {*Q*, *C′*, *c*} = *2* **and** *rk* {*P*, *Q*, *R*} = *2* **and**
rk {*P*, *R*} = *2* **and** *rk* {*Q*, *R*} = *2* **and** *rk* {*R*, *A*, *a*} = *2* **and** *rk* {*R*, *B*, *b*} = *2*

and
rk {*R*, *C*, *c*} = *2* **and** *rk* {*A*′, *P*} = *2* **and** *rk* {*B*′, *P*} = *2* **and** *rk* {*C*′, *P*} = *2*
shows *rk* {*a*, *b*, *c*, *α*, *β*, *γ*} = *3*
proof−
have *f1*:*rk* {*a*, *b*, *γ*} = *2*
using *rk-abγ*[*of A B C A*′ *B*′ *C*′ *P α β γ Q a b c R*] *assms*
by *auto*
have *f2*:*rk* {*a*, *c*, *β*} = *2*
using *rk-acβ*[*of A B C A*′ *B*′ *C*′ *P α β γ Q a b c R*] *assms*
by *auto*
have *f3*:*rk* {*b*, *c*, *α*} = *2*
using *rk-bcα*[*of A B C A*′ *B*′ *C*′ *P α β γ Q a b c R*] *assms*
by *auto*
have *rk* {*a*, *b*, *c*, *β*, *γ*} + *rk* {*a*} ≤ *rk* {*a*, *b*, *γ*} + *rk* {*a*, *c*, *β*}
using *matroid-ax-3-alt*[*of* {*a*} {*a*, *b*, *γ*} {*a*, *c*, *β*}]
by (*simp add*: *insert-commute*)
then have *rk* {*a*, *b*, *c*, *β*, *γ*} ≤ *3*
using *f1 f2*
by (*metis Suc-1 Suc-eq-plus1 Suc-le-mono add-Suc-right numeral-3-eq-3 numeral-nat*(*1*) *numerals*(*1*)
rk-singleton)
then have *f4*:*rk* {*a*, *b*, *c*, *β*, *γ*} = *3*
using *matroid-ax-2 rk-abc*[*of A B C A*′ *B*′ *C*′ *P α β γ Q a b c R*]
by (*metis Un-upper2 assms*(*1*) *assms*(*10*) *assms*(*11*) *assms*(*12*) *assms*(*13*) *assms*(*14*) *assms*(*2*) *assms*(*3*)
assms(*4*) *assms*(*5*) *assms*(*6*) *assms*(*7*) *assms*(*8*) *assms*(*9*) *insert-mono le-antisym sup-bot.comm-neutral*)
have *rk* {*a*, *b*, *c*} = *3*
using *rk-abc*[*of A B C A*′ *B*′ *C*′ *P α β γ Q a b c R*] *assms*(*1*) *assms*(*10*) *assms*(*11*) *assms*(*12*) *assms*(*13*)
assms(*14*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *assms*(*6*) *assms*(*7*) *assms*(*8*) *assms*(*9*)
by *blast*
then have *f5*:*rk* {*b*, *c*} = *2*
using *rk-triple-to-rk-couple rk-couple*
by *force*
have *rk* {*a*, *b*, *c*, *α*, *β*, *γ*} + *rk* {*b*, *c*} ≤ *rk* {*a*, *b*, *c*, *β*, *γ*} + *rk* {*b*, *c*, *α*}
using *matroid-ax-3-alt*[*of* {*b*, *c*} {*a*, *b*, *c*, *β*, *γ*} {*b*, *c*, *α*}]
by (*simp add*: *insert-commute*)
then have *rk* {*a*, *b*, *c*, *α*, *β*, *γ*} ≤ *3*
using *f3 f4 f5*
by *linarith*
thus *rk* {*a*, *b*, *c*, *α*, *β*, *γ*} = *3*
using *matroid-ax-2*
by (*metis* ‹*rk* {*a*, *b*, *c*} = *3*› *empty-subsetI insert-mono le-antisym*)
qed

lemma *rk-ABCαβγ* :
assumes *desargues-config-2D A B C A*′ *B*′ *C*′ *P α β γ* **and** *rk* {*A*, *B*, *C*, *A*′,

B', C', P, $Q\} \geq 4$
and *rk* $\{Q, A', a\} = 2$ **and** *rk* $\{Q, B', b\} = 2$ **and** *rk* $\{Q, C', c\} = 2$ **and** *rk* $\{P, Q, R\} = 2$ **and**
rk $\{P, R\} = 2$ **and** *rk* $\{Q, R\} = 2$ **and** *rk* $\{R, A, a\} = 2$ **and** *rk* $\{R, B, b\} = 2$ **and**
rk $\{R, C, c\} = 2$ **and** *rk* $\{A', P\} = 2$ **and** *rk* $\{B', P\} = 2$ **and** *rk* $\{C', P\} = 2$
shows *rk* $\{A, B, C, \alpha, \beta, \gamma\} = 3$
proof−
have *f1*:*rk* $\{A, B, \gamma\} = 2$
using *assms*(*1*) *desargues-config-2D-def*[*of A B C A′ B′ C′ P α β γ*]
by *auto*
have *f2*:*rk* $\{A, C, \beta\} = 2$
using *assms*(*1*) *desargues-config-2D-def*[*of A B C A′ B′ C′ P α β γ*]
by *auto*
have *f3*:*rk* $\{B, C, \alpha\} = 2$
using *assms*(*1*) *desargues-config-2D-def*[*of A B C A′ B′ C′ P α β γ*]
by *auto*
have *rk* $\{A, B, C, \beta, \gamma\}$ + *rk* $\{A\} \leq$ *rk* $\{A, B, \gamma\}$ + *rk* $\{A, C, \beta\}$
using *matroid-ax-3-alt*[*of* {*A*} {*A*, *B*, *γ*} {*A*, *C*, *β*}]
by (*simp add*: *insert-commute*)
then have *rk* $\{A, B, C, \beta, \gamma\} \leq 3$
using *f1 f2*
by (*metis Suc-1 Suc-eq-plus1 Suc-le-mono add-Suc-right numeral-3-eq-3 numeral-nat*(*1*) *numerals*(*1*)
rk-singleton)
have *rk* $\{A, B, C\} = 3$
using *assms*(*1*) *desargues-config-2D-def*[*of A B C A′ B′ C′ P α β γ*]
by *auto*
then have *f4*:*rk* $\{A, B, C, \beta, \gamma\} = 3$
using *matroid-ax-2*
by (*metis* ‹*rk* $\{A, B, C, \beta, \gamma\} \leq 3$› *empty-subsetI insert-mono le-antisym*)
have *f5*:*rk* $\{B, C\} = 2$
using ‹*rk* $\{A, B, C\} = 3$› *rk-couple rk-triple-to-rk-couple*
by *force*
have *rk* $\{A, B, C, \alpha, \beta, \gamma\}$ + *rk* $\{B, C\} \leq$ *rk* $\{A, B, C, \beta, \gamma\}$ + *rk* $\{B, C, \alpha\}$
using *matroid-ax-3-alt*[*of* {*B*, *C*} {*A*, *B*, *C*, *β*, *γ*} {*B*, *C*, *α*}]
by (*simp add*: *insert-commute*)
then have *rk* $\{A, B, C, \alpha, \beta, \gamma\} \leq 3$
using *f3 f4 f5*
by *linarith*
thus *rk* $\{A, B, C, \alpha, \beta, \gamma\} = 3$
using *matroid-ax-2*
by (*metis* ‹*rk* $\{A, B, C\} = 3$› *empty-subsetI insert-mono le-antisym*)
qed

lemma *rk-αβγ* :
assumes *desargues-config-2D A B C A′ B′ C′ P α β γ* **and** *rk* $\{A, B, C, A',$ $B', C', P, Q\} \geq 4$
and *rk* $\{Q, A', a\} = 2$ **and** *rk* $\{Q, B', b\} = 2$ **and** *rk* $\{Q, C', c\} = 2$ **and** *rk*

$\{P, Q, R\} = 2$ **and**
$rk\ \{P, R\} = 2$ **and** $rk\ \{Q, R\} = 2$ **and** $rk\ \{R, A, a\} = 2$ **and** $rk\ \{R, B, b\} = 2$
and
$rk\ \{R, C, c\} = 2$ **and** $rk\ \{A', P\} = 2$ **and** $rk\ \{B', P\} = 2$ **and** $rk\ \{C', P\} = 2$
shows $rk\ \{\alpha, \beta, \gamma\} \leq 2$
proof−
have $rk\ \{A, B, C, a, b, c\} \geq 4$
using *desargues-config-2D-rkABCabc*[*of* $A\ B\ C\ A'\ B'\ C'\ P\ \alpha\ \beta\ \gamma\ Q\ a\ R\ b\ c$] *assms*
by *auto*
have $\{A, B, C, a, b, c\} \subseteq \{A, B, C, a, b, c, \alpha, \beta, \gamma\}$
by *auto*
then have *f1*:$rk\ \{A, B, C, a, b, c, \alpha, \beta, \gamma\} \geq 4$
using *matroid-ax-2*
by (*meson* ‹$4 \leq rk\ \{A, B, C, a, b, c\}$› *dual-order.trans*)
have $rk\ \{A, B, C, a, b, c, \alpha, \beta, \gamma\} + rk\ \{\alpha, \beta, \gamma\} \leq rk\ \{A, B, C, \alpha, \beta, \gamma\} + rk\ \{a, b, c, \alpha, \beta, \gamma\}$
using *matroid-ax-3-alt*[*of* $\{\alpha, \beta, \gamma\}$ $\{A, B, C, \alpha, \beta, \gamma\}$ $\{a, b, c, \alpha, \beta, \gamma\}$]
by (*simp add*: *insert-commute*)
thus $rk\ \{\alpha, \beta, \gamma\} \leq 2$
using *f1 rk-ABC*$\alpha\beta\gamma$[*of* $A\ B\ C\ A'\ B'\ C'\ P\ \alpha\ \beta\ \gamma\ Q\ a\ b\ c\ R$] *rk-abc*$\alpha\beta\gamma$[*of* $A\ B\ C\ A'\ B'\ C'\ P\ \alpha\ \beta\ \gamma\ Q\ a\ b\ c\ R$]
by (*smt One-nat-def Suc-1 Suc-le-eq add-Suc-right add-le-imp-le-diff assms*(*1*) *assms*(*10*) *assms*(*11*)
assms(*12*) *assms*(*13*) *assms*(*14*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *assms*(*5*) *assms*(*6*) *assms*(*7*) *assms*(*8*)
assms(*9*) *diff-add-inverse2 le-antisym nat-1-add-1 not-less numeral-3-eq-3 one-plus-numeral*
rk-triple-le semiring-norm(*2*) *semiring-norm*(*4*))
qed

lemma *rk-*$\alpha\beta\gamma$*-special-case-1* :
assumes *desargues-config-2D* $A\ B\ C\ A'\ B'\ C'\ P\ \alpha\ \beta\ \gamma$ **and** $rk\ \{A', P\} = 1$
shows $rk\ \{\alpha, \beta, \gamma\} \leq 2$
proof−
have $A' = P$
by (*simp add*: *assms*(*2*) *rk-couple-to-singleton*)
then have $rk\ \{A', C', C, \beta\} + rk\ \{C', A'\} \leq rk\ \{C, C', P\} + rk\ \{A', C', \beta\}$
using *matroid-ax-3-alt*[*of* $\{C', A'\}$ $\{C, C', P\}$ $\{A', C', \beta\}$]
by (*simp add*: *insert-commute*)
then have $rk\ \{A', C', C, \beta\} \leq rk\ \{A', C', \beta\}$
using *assms*(*1*) *desargues-config-2D-def*[*of* $A\ B\ C\ A'\ B'\ C'\ P\ \alpha\ \beta\ \gamma$]
by (*metis* (*full-types*) *add-le-cancel-right insert-commute rk-triple-to-rk-couple*)
then have *f5*:$rk\ \{A', C', C, \beta\} = 2$
using *assms*(*1*) *desargues-config-2D-def*[*of* $A\ B\ C\ A'\ B'\ C'\ P\ \alpha\ \beta\ \gamma$]
by (*metis insert-commute le-antisym matroid-ax-2 subset-insertI*)
have $rk\ \{A, A', C, a, C', \beta\} + rk\ \{A, C, \beta\} \leq rk\ \{A, A', C', C, \beta\} + rk\ \{a, A, C, \beta\}$ **for** a
using *matroid-ax-3-alt*[*of* $\{A, C, \beta\}$ $\{A, A', C', C, \beta\}$ $\{a, A, C, \beta\}$]

by (*simp add*: *insert-commute*)
then have *f6*:*rk* $\{A, A', C', C, \beta\} \geq 3$
using *assms*(*1*) *desargues-config-2D-def*[*of* $A\ B\ C\ A'\ B'\ C'\ P\ \alpha\ \beta\ \gamma$] *rk-AA'Ca rk-acAC* β
by (*metis Un-insert-right Un-upper2* ‹$A' = P$› *insert-commute matroid-ax-2 sup-bot.right-neutral*)
have *rk* $\{A, A', C', C, \beta\}$ + *rk* $\{\beta, C\} \leq$ *rk* $\{A, C, \beta\}$ + *rk* $\{A', C', C, \beta\}$
using *matroid-ax-3-alt*[*of* $\{\beta, C\}$ $\{A, C, \beta\}$ $\{A', C', C, \beta\}$]
by (*simp add*: *insert-commute*)
then have *rk* $\{\beta, C\} \leq 1$
using *assms*(*1*) *desargues-config-2D-def*[*of* $A\ B\ C\ A'\ B'\ C'\ P\ \alpha\ \beta\ \gamma$] *f5 f6*
by *linarith*
then have $\beta = C$
using *rk-couple*
by *force*
have *rk* $\{A', B', B, \gamma\}$ + *rk* $\{B', A'\} \leq$ *rk* $\{B, B', P\}$ + *rk* $\{A', B', \gamma\}$
using *matroid-ax-3-alt*[*of* $\{B', A'\}$ $\{B, B', P\}$ $\{A', B', \gamma\}$]
by (*simp add*: ‹$A' = P$› *insert-commute*)
then have *rk* $\{A', B', B, \gamma\} \leq$ *rk* $\{A', B', \gamma\}$
using *assms*(*1*) *desargues-config-2D-def*[*of* $A\ B\ C\ A'\ B'\ C'\ P\ \alpha\ \beta\ \gamma$]
by (*metis* (*full-types*) *add-le-cancel-right insert-commute rk-triple-to-rk-couple*)
then have *f7*:*rk* $\{A', B', B, \gamma\} = 2$
using *assms*(*1*) *desargues-config-2D-def*[*of* $A\ B\ C\ A'\ B'\ C'\ P\ \alpha\ \beta\ \gamma$]
by (*metis insert-commute le-antisym matroid-ax-2 subset-insertI*)
have *rk* $\{A, A', B, a, B', \gamma\}$ + *rk* $\{A, B, \gamma\} \leq$ *rk* $\{A, A', B', B, \gamma\}$ + *rk* $\{a, A, B, \gamma\}$ **for** a
using *matroid-ax-3-alt*[*of* $\{A, B, \gamma\}$ $\{A, A', B', B, \gamma\}$ $\{a, A, B, \gamma\}$]
by (*simp add*: *insert-commute*)
then have *f8*:*rk* $\{A, A', B', B, \gamma\} \geq 3$
using *assms*(*1*) *desargues-config-2D-def*[*of* $A\ B\ C\ A'\ B'\ C'\ P\ \alpha\ \beta\ \gamma$] *rk-AA'Ca rk-acAC* β
by (*metis Un-insert-right Un-upper2* ‹$A' = P$› *insert-commute matroid-ax-2 sup-bot.right-neutral*)
have *rk* $\{A, A', B', B, \gamma\}$ + *rk* $\{\gamma, B\} \leq$ *rk* $\{A, B, \gamma\}$ + *rk* $\{A', B', B, \gamma\}$
using *matroid-ax-3-alt*[*of* $\{\gamma, B\}$ $\{A, B, \gamma\}$ $\{A', B', B, \gamma\}$]
by (*simp add*: *insert-commute*)
then have *rk* $\{\gamma, B\} \leq 1$
using *assms*(*1*) *desargues-config-2D-def*[*of* $A\ B\ C\ A'\ B'\ C'\ P\ \alpha\ \beta\ \gamma$] *f7 f8*
by *linarith*
then have $\gamma = B$
using *rk-couple*
by *force*
then have *rk* $\{\alpha, \beta, \gamma\}$ = *rk* $\{\alpha, C, B\}$
using ‹$\beta = C$ ›
by *auto*
thus *rk* $\{\alpha, \beta, \gamma\} \leq 2$
using *assms*(*1*) *desargues-config-2D-def*[*of* $A\ B\ C\ A'\ B'\ C'\ P\ \alpha\ \beta\ \gamma$]
by (*metis insert-commute order-refl*)
qed

lemma *rk-αβγ-special-case-2* :
  assumes *desargues-config-2D A B C A′ B′ C′ P α β γ* **and** *rk* {*B′*, *P*} = *1*
  shows *rk* {*α*, *β*, *γ*} ≤ *2*
proof−
  have *desargues-config-2D B A C B′ A′ C′ P β α γ*
    using *assms*(*1*) *desargues-config-2D-def*[*of A B C A′ B′ C′ P α β γ*]
      desargues-config-2D-def[*of B A C B′ A′ C′ P β α γ*]
    by (*metis insert-commute*)
  thus *rk* {*α*, *β*, *γ*} ≤ *2*
    using *rk-αβγ-special-case-1*[*of B A C B′ A′ C′ P β α γ*] *assms*(*2*)
    by (*simp add*: *insert-commute*)
qed

lemma *rk-αβγ-special-case-3* :
  assumes *desargues-config-2D A B C A′ B′ C′ P α β γ* **and** *rk* {*C′*, *P*} = *1*
  shows *rk* {*α*, *β*, *γ*} ≤ *2*
proof−
  have *desargues-config-2D C B A C′ B′ A′ P γ β α*
    using *assms*(*1*) *desargues-config-2D-def*[*of A B C A′ B′ C′ P α β γ*]
      desargues-config-2D-def[*of C B A C′ B′ A′ P γ β α*]
    by (*metis insert-commute*)
  thus *rk* {*α*, *β*, *γ*} ≤ *2*
    using *rk-αβγ-special-case-1*[*of C B A C′ B′ A′ P γ β α*] *assms*(*2*)
    by (*simp add*: *insert-commute*)
qed

theorem *desargues-2D* :
  assumes *desargues-config-2D A B C A′ B′ C′ P α β γ*
  shows *rk* {*α*, *β*, *γ*} ≤ *2*
proof−
  have *f1*:*rk* {*A*, *B*, *C*, *A′*, *B′*, *C′*, *P*} = *3*
    using *assms desargues-config-2D-def*[*of A B C A′ B′ C′ P α β γ*] *coplanar-ABCA′B′C′P*
    by *auto*
  obtain *Q* **where** *rk* {*A*, *B*, *C*, *A′*, *B′*, *C′*, *P*, *Q*} ≥ *4*
    using *f1 rk-ext*[*of* {*A*, *B*, *C*, *A′*, *B′*, *C′*, *P*}]
    by (*metis Un-insert-left add.commute one-plus-numeral order-refl semiring-norm*(*2*) *semiring-norm*(*4*)
      sup-bot.left-neutral)
  obtain *R* **where** *rk* {*P*, *Q*, *R*} = *2* **and** *rk* {*P*, *R*} = *2* **and** *rk* {*Q*, *R*} = *2*
    using *rk-ax-3-pts*
    by *auto*
  have *rk* {*Q*, *A′*, *R*, *A*, *P*} + *rk* {*P*} ≤ *rk* {*P*, *Q*, *R*} + *rk* {*A*, *A′*, *P*}
    using *matroid-ax-3-alt*[*of* {*P*} {*P*, *Q*, *R*} {*A*, *A′*, *P*}]
    by (*simp add*: *insert-commute*)
  then have *rk* {*Q*, *A′*, *R*, *A*, *P*} ≤ *3*
    using *rk-singleton assms desargues-config-2D-def*[*of A B C A′ B′ C′ P α β γ*]
    by (*metis Suc-1 Suc-eq-plus1 Suc-le-mono* ‹*rk* {*P*, *Q*, *R*} = *2*› *add-Suc-right*

eval-nat-numeral(*3*))
then have *f2*:*rk* {*Q*, *A*′, *R*, *A*} ≤ *3*
using *matroid-ax-2*
by (*metis* (*no-types*, *opaque-lifting*) *dual-order.trans insert-commute subset-insertI*)
obtain *a* **where** *rk* {*Q*, *A*′, *a*} = *2* **and** *rk* {*R*, *A*, *a*} = *2*
using *f2 rk-ax-pasch*
by *blast*
have *rk* {*Q*, *B*′, *R*, *B*, *P*} + *rk* {*P*} ≤ *rk* {*P*, *Q*, *R*} + *rk* {*B*, *B*′, *P*}
using *matroid-ax-3-alt*[*of* {*P*} {*P*, *Q*, *R*} {*B*, *B*′, *P*}]
by (*simp add*: *insert-commute*)
then have *rk* {*Q*, *B*′, *R*, *B*, *P*} ≤ *3*
using *rk-singleton assms desargues-config-2D-def*[*of A B C A*′ *B*′ *C*′ *P* α β γ]
by (*metis Suc-1 Suc-eq-plus1 Suc-le-mono* ‹*rk* {*P*, *Q*, *R*} = *2*› *add-Suc-right* *eval-nat-numeral*(*3*))
then have *f3*:*rk* {*Q*, *B*′, *R*, *B*} ≤ *3*
using *matroid-ax-2*
by (*metis* (*no-types*, *opaque-lifting*) *dual-order.trans insert-commute subset-insertI*)
obtain *b* **where** *rk* {*Q*, *B*′, *b*} = *2* **and** *rk* {*R*, *B*, *b*} = *2*
using *f3 rk-ax-pasch*
by *blast*
have *rk* {*Q*, *C*′, *R*, *C*, *P*} + *rk* {*P*} ≤ *rk* {*P*, *Q*, *R*} + *rk* {*C*, *C*′, *P*}
using *matroid-ax-3-alt*[*of* {*P*} {*P*, *Q*, *R*} {*C*, *C*′, *P*}]
by (*simp add*: *insert-commute*)
then have *rk* {*Q*, *C*′, *R*, *C*, *P*} ≤ *3*
using *rk-singleton assms desargues-config-2D-def*[*of A B C A*′ *B*′ *C*′ *P* α β γ]
by (*metis Suc-1 Suc-eq-plus1 Suc-le-mono* ‹*rk* {*P*, *Q*, *R*} = *2*› *add-Suc-right* *eval-nat-numeral*(*3*))
then have *f4*:*rk* {*Q*, *C*′, *R*, *C*} ≤ *3*
using *matroid-ax-2*
by (*metis* (*no-types*, *opaque-lifting*) *dual-order.trans insert-commute subset-insertI*)
obtain *c* **where** *rk* {*Q*, *C*′, *c*} = *2* **and** *rk* {*R*, *C*, *c*} = *2*
using *f4 rk-ax-pasch*
by *blast*
then have *rk* {α, β, γ} ≤ *2* **if** *rk* {*A*′, *P*} = *2* **and** *rk* {*B*′, *P*} = *2* **and** *rk* {*C*′, *P*} = *2*
using *rk-αβγ*[*of A B C A*′ *B*′ *C*′ *P* α β γ *Q a b c R*] ‹*4* ≤ *rk* {*A*, *B*, *C*, *A*′, *B*′, *C*′, *P*, *Q*}›
‹*rk* {*P*, *Q*, *R*} = *2*› ‹*rk* {*P*, *R*} = *2*› ‹*rk* {*Q*, *A*′, *a*} = *2*› ‹*rk* {*Q*, *B*′, *b*} = *2*› ‹*rk* {*Q*, *R*} = *2*›
‹*rk* {*R*, *A*, *a*} = *2*› ‹*rk* {*R*, *B*, *b*} = *2*› *assms that*(*1*) *that*(*2*) *that*(*3*)
by *blast*
have *rk* {α, β, γ} ≤ *2* **if** *rk* {*A*′, *P*} = *1*
using *rk-αβγ-special-case-1 assms that*
by *auto*
have *rk* {α, β, γ} ≤ *2* **if** *rk* {*B*′, *P*} = *1*
using *rk-αβγ-special-case-2 assms that*
by *auto*
have *rk* {α, β, γ} ≤ *2* **if** *rk* {*C*′, *P*} = *1*
using *rk-αβγ-special-case-3 assms that*

by *auto*
thus $rk\ \{\alpha, \beta, \gamma\} \leq 2$
using ‹$[\![rk\ \{A', P\} = 2;\ rk\ \{B', P\} = 2;\ rk\ \{C', P\} = 2]\!] \Longrightarrow rk\ \{\alpha, \beta, \gamma\} \leq 2$›
‹$rk\ \{A', P\} = 1 \Longrightarrow rk\ \{\alpha, \beta, \gamma\} \leq 2$› ‹$rk\ \{B', P\} = 1 \Longrightarrow rk\ \{\alpha, \beta, \gamma\} \leq 2$›
rk-couple rk-singleton-bis
by *blast*
qed
end

end
theory *Desargues-3D*
imports *Main Higher-Projective-Space-Rank-Axioms Matroid-Rank-Properties*
begin

Contents:

- We prove Desargues's theorem: if two triangles ABC and A'B'C' are perspective from a point P (ie. the lines AA', BB' and CC' are concurrent in P), then they are perspective from a line (ie. the points $\alpha = BC \cap B'C'$, $\beta = AC \cap A'C'$ and $\gamma = AB \cap A'B'$ are collinear). In this file we restrict ourself to the case where the two triangles ABC and A'B'C' are not coplanar.

9 Desargues's Theorem: The Non-coplanar Case

context *higher-projective-space-rank*
begin

definition *desargues-config-3D* ::
['point, 'point, 'point, 'point, 'point, 'point, 'point, 'point, 'point, 'point] => bool
where $desargues\text{-}config\text{-}3D\ A\ B\ C\ A'\ B'\ C'\ P\ \alpha\ \beta\ \gamma \equiv rk\ \{A, B, C\} = 3 \wedge rk\ \{A', B', C'\} = 3 \wedge$
$rk\ \{A, A', P\} = 2 \wedge rk\ \{B, B', P\} = 2 \wedge rk\ \{C, C', P\} = 2 \wedge rk\ \{A, B, C, A', B', C'\} \geq 4 \wedge$
$rk\ \{B, C, \alpha\} = 2 \wedge rk\ \{B', C', \alpha\} = 2 \wedge rk\ \{A, C, \beta\} = 2 \wedge rk\ \{A', C', \beta\} = 2 \wedge rk\ \{A, B, \gamma\} = 2 \wedge$
$rk\ \{A', B', \gamma\} = 2$

lemma *coplanar-4* :
assumes $rk\ \{A, B, C\} = 3$ **and** $rk\ \{B, C, \alpha\} = 2$
shows $rk\ \{A, B, C, \alpha\} = 3$
proof−
have $f1{:}rk\ \{A, B, C, \alpha\} \geq 3$
using *matroid-ax-2*
by (*metis assms*(*1*) *empty-subsetI insert-mono*)

have *rk* $\{A, B, C, \alpha\}$ + *rk* $\{B, C\} \leq$ *rk* $\{A, B, C\}$ + *rk* $\{B, C, \alpha\}$
using *matroid-ax-3-alt*
by (*metis Un-insert-right add-One-commute add-mono assms*(*1*) *assms*(*2*) *matroid-ax-2-alt*
nat-1-add-1 numeral-plus-one rk-singleton semiring-norm(*3*) *sup-bot.right-neutral*)
then have *f2*:*rk* $\{A, B, C, \alpha\} \leq 3$
by (*metis Un-insert-right add-One-commute assms*(*2*) *matroid-ax-2-alt numeral-plus-one*
semiring-norm(*3*) *sup-bot.right-neutral*)
from *f1* **and** *f2* **show** *rk* $\{A, B, C, \alpha\} = 3$
by *auto*
qed

lemma *desargues-config-3D-coplanar-4* :
assumes *desargues-config-3D* $A\ B\ C\ A'\ B'\ C'\ P\ \alpha\ \beta\ \gamma$
shows *rk* $\{A, B, C, \alpha\} = 3$ **and** *rk* $\{A', B', C', \alpha\} = 3$
proof−
show *rk* $\{A, B, C, \alpha\} = 3$
using *assms desargues-config-3D-def*[*of* $A\ B\ C\ A'\ B'\ C'\ P\ \alpha\ \beta\ \gamma$] *coplanar-4*
by *blast*
show *rk* $\{A', B', C', \alpha\} = 3$
using *assms desargues-config-3D-def*[*of* $A\ B\ C\ A'\ B'\ C'\ P\ \alpha\ \beta\ \gamma$] *coplanar-4*
by *blast*
qed

lemma *coplanar-4-bis* :
assumes *rk* $\{A, B, C\} = 3$ **and** *rk* $\{A, C, \beta\} = 2$
shows *rk* $\{A, B, C, \beta\} = 3$
by (*smt assms*(*1*) *assms*(*2*) *coplanar-4 insert-commute*)

lemma *desargues-config-3D-coplanar-4-bis* :
assumes *desargues-config-3D* $A\ B\ C\ A'\ B'\ C'\ P\ \alpha\ \beta\ \gamma$
shows *rk* $\{A, B, C, \beta\} = 3$ **and** *rk* $\{A', B', C', \beta\} = 3$
proof−
show *rk* $\{A, B, C, \beta\} = 3$
using *assms desargues-config-3D-def*[*of* $A\ B\ C\ A'\ B'\ C'\ P\ \alpha\ \beta\ \gamma$] *coplanar-4-bis*

by *auto*
show *rk* $\{A', B', C', \beta\} = 3$
using *assms desargues-config-3D-def*[*of* $A\ B\ C\ A'\ B'\ C'\ P\ \alpha\ \beta\ \gamma$] *coplanar-4-bis*

by *auto*
qed

lemma *coplanar-4-ter* :
assumes *rk* $\{A, B, C\} = 3$ **and** *rk* $\{A, B, \gamma\} = 2$
shows *rk* $\{A, B, C, \gamma\} = 3$
by (*smt assms*(*1*) *assms*(*2*) *coplanar-4 insert-commute*)

lemma *desargues-config-3D-coplanar-4-ter* :
assumes *desargues-config-3D A B C A′ B′ C′ P α β γ*
shows *rk* {*A*, *B*, *C*, *γ*} = *3* **and** *rk* {*A′*, *B′*, *C′*, *γ*} = *3*
proof−
show *rk* {*A*, *B*, *C*, *γ*} = *3*
using *assms desargues-config-3D-def*[*of A B C A′ B′ C′ P α β γ*] *coplanar-4-ter*

by *auto*
show *rk* {*A′*, *B′*, *C′*, *γ*} = *3*
using *assms desargues-config-3D-def*[*of A B C A′ B′ C′ P α β γ*] *coplanar-4-ter*

by *auto*
qed

lemma *coplanar-5* :
assumes *rk* {*A*, *B*, *C*} = *3* **and** *rk* {*B*, *C*, *α*} = *2* **and** *rk* {*A*, *C*, *β*} = *2*
shows *rk* {*A*, *B*, *C*, *α*, *β*} = *3*
proof−
have *f1*:*rk* {*A*, *B*, *C*, *α*} = *3*
using *coplanar-4*
by (*smt One-nat-def Un-assoc Un-commute add.commute add-Suc-right assms*(*1*) *assms*(*2*) *insert-is-Un*
le-antisym matroid-ax-2-alt numeral-2-eq-2 numeral-3-eq-3 one-add-one)
have *f2*:*rk* {*A*, *B*, *C*, *β*} = *3*
using *coplanar-4-bis*
by (*smt One-nat-def Un-assoc Un-commute add.commute add-Suc-right assms*(*1*) *assms*(*3*) *insert-is-Un*
le-antisym matroid-ax-2-alt numeral-2-eq-2 numeral-3-eq-3 one-add-one)
from *f1* **and** *f2* **show** *rk* {*A*, *B*, *C*, *α*, *β*} = *3*
using *matroid-ax-3-alt′*
by (*metis Un-assoc assms*(*1*) *insert-is-Un*)
qed

lemma *desargues-config-3D-coplanar-5* :
assumes *desargues-config-3D A B C A′ B′ C′ P α β γ*
shows *rk* {*A*, *B*, *C*, *α*, *β*} = *3* **and** *rk* {*A′*, *B′*, *C′*, *α*, *β*} = *3*
proof−
show *rk* {*A*, *B*, *C*, *α*, *β*} = *3*
using *assms desargues-config-3D-def coplanar-5*
by *auto*
show *rk* {*A′*, *B′*, *C′*, *α*, *β*} = *3*
using *assms desargues-config-3D-def coplanar-5*
by *auto*
qed

lemma *coplanar-5-bis* :
assumes *rk* {*A*, *B*, *C*} = *3* **and** *rk* {*B*, *C*, *α*} = *2* **and** *rk* {*A*, *B*, *γ*} = *2*
shows *rk* {*A*, *B*, *C*, *α*, *γ*} = *3*

by (*smt assms coplanar-5 insert-commute*)

lemma *desargues-config-3D-coplanar-5-bis* :
assumes *desargues-config-3D A B C A′ B′ C′ P α β γ*
shows *rk* {*A*, *B*, *C*, *α*, *γ*} = *3* **and** *rk* {*A′*, *B′*, *C′*, *α*, *γ*} = *3*
proof−
show *rk* {*A*, *B*, *C*, *α*, *γ*} = *3*
using *assms desargues-config-3D-def*[*of A B C A′ B′ C′ P α β γ*] *coplanar-5-bis*

by *auto*
show *rk* {*A′*, *B′*, *C′*, *α*, *γ*} = *3*
using *assms desargues-config-3D-def*[*of A B C A′ B′ C′ P α β γ*] *coplanar-5-bis*

by *auto*
qed

lemma *coplanar-6* :
assumes *rk* {*A*, *B*, *C*} = *3* **and** *rk* {*B*, *C*, *α*} = *2* **and** *rk* {*A*, *B*, *γ*} = *2* **and** *rk* {*A*, *C*, *β*} = *2*
shows *rk* {*A*, *B*, *C*, *α*, *β*, *γ*} = *3*
proof−
have *f1*:*rk* {*A*, *B*, *C*, *α*, *γ*} = *3*
using *coplanar-5-bis assms*(*1*) *assms*(*2*) *assms*(*3*)
by *auto*
have *f2*:*rk* {*A*, *B*, *C*, *α*, *β*} = *3*
using *coplanar-5 assms*(*1*) *assms*(*2*) *assms*(*4*)
by *auto*
have *f3*:*rk* {*A*, *B*, *C*, *α*} = *3*
using *coplanar-4 assms*(*1*) *assms*(*2*)
by *auto*
from *f1* **and** *f2* **and** *f3* **show** *rk* {*A*, *B*, *C*, *α*, *β*, *γ*} = *3*
using *matroid-ax-3-alt′*[*of* {*A*, *B*, *C*, *α*} *β γ*]
by (*metis Un-insert-left sup-bot.left-neutral*)
qed

lemma *desargues-config-3D-coplanar-6* :
assumes *desargues-config-3D A B C A′ B′ C′ P α β γ*
shows *rk* {*A*, *B*, *C*, *α*, *β*, *γ*} = *3* **and** *rk* {*A′*, *B′*, *C′*, *α*, *β*, *γ*} = *3*
proof−
show *rk* {*A*, *B*, *C*, *α*, *β*, *γ*} = *3*
using *assms desargues-config-3D-def*[*of A B C A′ B′ C′ P α β γ*] *coplanar-6*
by *auto*
show *rk* {*A′*, *B′*, *C′*, *α*, *β*, *γ*} = *3*
using *assms desargues-config-3D-def*[*of A B C A′ B′ C′ P α β γ*] *coplanar-6*
by *auto*
qed

lemma *desargues-config-3D-non-coplanar* :
assumes *desargues-config-3D A B C A′ B′ C′ P α β γ*

shows *rk* $\{A, B, C, A', B', C', \alpha, \beta, \gamma\} \geq 4$
proof−
have *rk* $\{A, B, C, A', B', C'\} \leq$ *rk* $\{A, B, C, A', B', C', \alpha, \beta, \gamma\}$
using *matroid-ax-2*
by *auto*
thus $4 \leq$ *rk* $\{A, B, C, A', B', C', \alpha, \beta, \gamma\}$
using *matroid-ax-2 assms desargues-config-3D-def*[*of* A B C A' B' C' P α β γ]
by *linarith*
qed

theorem *desargues-3D* :
assumes *desargues-config-3D* A B C A' B' C' P α β γ
shows *rk* $\{\alpha, \beta, \gamma\} \leq 2$
proof−
have *rk* $\{A, B, C, A', B', C', \alpha, \beta, \gamma\}$ + *rk* $\{\alpha, \beta, \gamma\} \leq$ *rk* $\{A, B, C, \alpha, \beta, \gamma\}$ + *rk* $\{A', B', C', \alpha, \beta, \gamma\}$
using *matroid-ax-3-alt*[*of* $\{\alpha, \beta, \gamma\}$ $\{A, B, C, \alpha, \beta, \gamma\}$ $\{A', B', C', \alpha, \beta, \gamma\}$]
by (*simp add*: *insert-commute*)
then have *rk* $\{\alpha, \beta, \gamma\} \leq$ *rk* $\{A, B, C, \alpha, \beta, \gamma\}$ + *rk* $\{A', B', C', \alpha, \beta, \gamma\}$ − *rk* $\{A, B, C, A', B', C', \alpha, \beta, \gamma\}$
by *linarith*
thus *rk* $\{\alpha, \beta, \gamma\} \leq 2$
using *assms desargues-config-3D-coplanar-6 desargues-config-3D-non-coplanar*
by *fastforce*
qed

end

end
theory *Projective-Space-Axioms*
imports *Main*
begin

Contents:

- We introduce the types *'point* of points and *'line* of lines and an incidence relation between them.
- A set of axioms for the (3-dimensional) projective space. An alternative set of axioms could use planes as basic objects in addition to points and lines

10 The axioms of the Projective Space

lemma *distinct4-def*:
distinct $[A,B,C,D] = ((A \neq B) \wedge (A \neq C) \wedge (A \neq D) \wedge (B \neq C) \wedge (B \neq D) \wedge (C \neq D))$

by *auto*

lemma *distinct3-def*:
distinct $[A, B, C] = (A \neq B \wedge A \neq C \wedge B \neq C)$
by *auto*

locale *projective-space* =

fixes *incid* :: *'point* ⇒ *'line* ⇒ *bool*
fixes *meet* :: *'line* ⇒ *'line* ⇒ *'point*
assumes *meet-def*: (*incid* (*meet l m*) *l* ∧ *incid* (*meet l m*) *m*)

assumes *incid-dec*: (*incid P l*) ∨ ¬(*incid P l*)

assumes *ax1-existence*: ∃ *l*. (*incid P l*) ∧ (*incid M l*)
assumes *ax1-uniqueness*: (*incid P k*) ⟶ (*incid M k*) ⟶ (*incid P l*) ⟶ (*incid M l*) ⟶ (*P* = *M*) ∨ (*k* = *l*)

assumes *ax2*: *distinct* [*A*,*B*,*C*,*D*] ⟶ (*incid A lAB* ∧ *incid B lAB*) ⟶ (*incid C lCD* ∧ *incid D lCD*) ⟶ (*incid A lAC* ∧ *incid C lAC*) ⟶ (*incid B lBD* ∧ *incid D lBD*) ⟶ (∃ *I*.(*incid I lAB* ∧ *incid I lCD*)) ⟶ (∃ *J*.(*incid J lAC* ∧ *incid J lBD*))

assumes *ax3*: ∃ *A B C*. *distinct3 A B C* ∧ (*incid A l*) ∧ (*incid B l*) ∧ (*incid C l*)

assumes *ax4*: ∃ *l m*.∀ *P*. ¬(*incid P l* ∧ *incid P m*)

assumes *ax5*: *distinct* [*l1*,*l2*,*l3*] ⟶ (∃ *l4 J1 J2 J3*. *distinct* [*J1*,*J2*,*J3*] ∧ *meet l1 l4* = *J1* ∧ *meet l2 l4* = *J2* ∧ *meet l3 l4* = *J3*)

end
theory *Higher-Projective-Space-Axioms*
imports *Main*

begin

Contents:

- We introduce the types of 'point and 'line and an incidence relation between them.
- A set of axioms for higher projective spaces, i.e. we allow models of dimension > 3.

11 The axioms for Higher Projective Geometry

lemma *distinct4-def*:
distinct $[A,B,C,D] = ((A \neq B) \wedge (A \neq C) \wedge (A \neq D) \wedge (B \neq C) \wedge (B \neq D) \wedge (C \neq D))$
by *auto*

lemma *distinct3-def*:
distinct $[A,B,C] = ((A \neq B) \wedge (A \neq C) \wedge (B \neq C))$
by *auto*

locale *higher-projective-space* =

fixes *incid* :: *'point* ⇒ *'line* ⇒ *bool*

assumes *ax1-existence*: $\exists\, l.\ (incid\ P\ l) \wedge (incid\ M\ l)$
assumes *ax1-uniqueness*: $(incid\ P\ k) \longrightarrow (incid\ M\ k) \longrightarrow (incid\ P\ l) \longrightarrow (incid\ M\ l) \longrightarrow (P = M) \vee (k = l)$

assumes *ax2*: *distinct* $[A,B,C,D] \longrightarrow (incid\ A\ lAB \wedge incid\ B\ lAB) \longrightarrow (incid\ C\ lCD \wedge incid\ D\ lCD) \longrightarrow (incid\ A\ lAC \wedge incid\ C\ lAC) \longrightarrow (incid\ B\ lBD \wedge incid\ D\ lBD) \longrightarrow (\exists\, I.(incid\ I\ lAB \wedge incid\ I\ lCD)) \longrightarrow (\exists\, J.(incid\ J\ lAC \wedge incid\ J\ lBD))$

assumes *ax3*: $\exists\, A\ B\ C.$ *distinct* $[A,B,C] \wedge (incid\ A\ l) \wedge (incid\ B\ l) \wedge (incid\ C\ l)$

assumes *ax4*: $\exists\, l\ m. \forall\, P.\ \neg(incid\ P\ l \wedge incid\ P\ m)$

end

12 Acknowledgements

The author was supported by the ERC Advanced Grant ALEXANDRIA (Project 742178) funded by the European Research Council and led by Professor Lawrence Paulson at the University of Cambridge, UK.

References

[1] M. Bezem and D. Hendriks. On the mechanization of the proof of hessenberg's theorem in coherent logic. *Journal of Automated Reasoning*, 40(1):61–85, Jan 2008.

[2] N. Magaud, J. Narboux, and P. Schreck. A case study in formalizing projective geometry in coq: Desargues theorem. *Computational Geometry*, 45(8):406–424, Oct 2012.