# A Variant of the Superposition Calculus 

Nicolas Peltier<br>CNRS/University of Grenoble (LIG)

September 13, 2023


#### Abstract

We provide a formalization in Isabelle/Isar of (a variant of) the superposition calculus [1, 4], together with formal proofs of soundness and refutational completeness (w.r.t. the usual redundancy criteria based on clause ordering). This version of the calculus uses all the standard restrictions of the superposition rules, together with the following refinement, inspired by the basic superposition calculus [2, 3]: each clause is associated with a set of terms which are assumed to be in normal form - thus any application of the replacement rule on these terms is blocked. The set is initially empty and terms may be added or removed at each inference step. The set of terms that are assumed to be in normal form includes any term introduced by previous unifiers as well as any term occurring in the parent clauses at a position that is smaller (according to some given ordering on positions) than a previously replaced term. This restriction is slightly weaker than that of the basic superposition calculus (since it is based on terms instead of positions), but it has the advantage that the irreducible terms may be propagated through the inferences (under appropriate conditions), even if they do not occur in the parent clauses. The standard superposition calculus corresponds to the case where the set of irreducible terms is always empty. The term representation and unification algorithm are taken from the theory Unification.thy provided in Isabelle.


## Contents

1 Preliminaries ..... 3
1.1 Multisets ..... 3
1.2 Well-Founded Sets ..... 13
2 Terms ..... 14
2.1 Basic Syntax ..... 14
2.2 Positions ..... 15
2.3 Substitutions and Most General Unifiers ..... 27
2.3.1 Minimum Idempotent Most General Unifier ..... 36
2.4 Congruences ..... 37
2.5 Renamings ..... 39
3 Equational Clausal Logic ..... 44
3.1 Syntax ..... 44
3.2 Semantics ..... 55
4 Definition of the Superposition Calculus ..... 58
4.1 Extended Clauses ..... 58
4.2 Orders and Selection Functions ..... 58
4.3 Clause Ordering ..... 63
4.4 Inference Rules ..... 91
4.5 Derivations ..... 96
5 Soundness ..... 97
6 Redundancy Criteria and Saturated Sets ..... 118
7 Refutational Completeness ..... 133
7.1 Model Construction ..... 133
7.2 Lifting ..... 180
7.3 Satisfiability of Saturated Sets with No Empty Clause ..... 198

## 1 Preliminaries

theory multisets－continued
imports Main HOL－Library．Multiset
begin

## 1．1 Multisets

We use the Multiset theory provided in Isabelle．We prove some additional （mostly trivial）lemmata．

```
lemma mset-set-inclusion:
    assumes finite E2
    assumes E1 \subset E2
    shows mset-set E1 \subset# (mset-set E2)
proof (rule ccontr)
    let ?s1 = mset-set E1
    let ?s2 = mset-set E2
    assume ᄀ?s1 \subset#?s2
    from assms(1) and assms(2) have finite E1 using finite-subset less-imp-le by
auto
    from «\neg ?s1 \subset# ?s2` obtain x where (count ?s1 x > count ?s2 x) using
subseteq-mset-def [of ?s1 ?s2]
    by (metis assms(1) assms(2) finite-set-mset-mset-set finite-subset less-imp-le
        less-le not-le-imp-less subset-mset.le-less)
    from this have count ?s1 x>0 by linarith
    from this and 〈finite E1〉 have count ?s1 x = 1 and x\inE1 using sub-
seteq-mset-def by auto
    from this and assms(2) have x E E2 by auto
    from this and <finite E2〉 have count ?s2 x = 1 by auto
    from this and <count ?s1 x = 1〉 and «(count ?s1 x > count ?s2 x)\rangle show False
by auto
qed
lemma mset-ordering-addition:
    assumes }A=B+
    shows B\subseteq# A
    using assms by simp
lemma equal-image-mset:
    assumes }\forallx\inE.(fx)=(gx
    shows {# (fx). x\in# (mset-set E)#} ={# (g x). x\in# (mset-set E) #}
by (meson assms count-eq-zero-iff count-mset-set(3) image-mset-cong)
lemma multiset-order-inclusion:
    assumes E\subset#F
```

```
    assumes trans \(r\)
    shows \((E, F) \in(\) mult \(r)\)
proof -
    let \(? G=F-E\)
    from \(\operatorname{assms}(1)\) have \(F=E+\) ? \(G\)
        by (simp add: subset-mset.add-diff-inverse subset-mset-def)
    from this assms (1) have ? \(G \neq\{\#\}\)
    by fastforce
    have \(E=E+\{\#\}\) by auto
    from this \(\langle F=E+? G\rangle\langle ? G \neq\{\#\}\rangle\) assms(2) show ?thesis using one-step-implies-mult
[of ? \(G\{\#\} r E]\) by auto
qed
lemma multiset-order-inclusion-eq:
    assumes \(E \subseteq \# F\)
    assumes trans \(r\)
    shows \(E=F \vee(E, F) \in(\) mult \(r)\)
proof (cases)
    assume \(E=F\)
    then show ?thesis by auto
next
    assume \(E \neq F\)
    from this and assms(1) have \(E \subset \# F\) by auto
    from this assms(2) and multiset-order-inclusion show ?thesis by auto
qed
lemma image-mset-ordering:
    assumes \(M 1=\{\#(f 1 u) \cdot u \in \# L \#\}\)
    assumes M2 \(=\{\#(f 2 u) \cdot u \in \# L \#\}\)
    assumes \(\forall u .(u \in \# L \longrightarrow(((f 1 u),(f 2 u)) \in r \vee(f 1 u)=(f 2 u)))\)
    assumes \(\exists u .(u \in \# L \wedge((f 1 u),(f 2 u)) \in r)\)
    assumes irrefl \(r\)
    shows \(((\) M1, M2 \() \in(\) mult \(r))\)
proof -
    let \(? L^{\prime}=\{\# u \in \# L . \quad(f 1 u)=(\) f2 \(u) \#\}\)
    let \(? L^{\prime \prime}=\{\# u \in \# L .(f 1 u) \neq(f 2 u) \#\}\)
    have \(L=? L^{\prime}+? L^{\prime \prime}\) by (simp)
    from \(\operatorname{assms}(3)\) have \(\forall u .\left(u \in \# ? L^{\prime \prime} \longrightarrow((f 1 u),(f 2 u)) \in r\right)\) by auto
    let \(? M 1^{\prime}=\left\{\#(f 1 u) . u \in \# ? L^{\prime} \#\right\}\)
    let ? M \(^{\prime}{ }^{\prime}=\left\{\#(f 2 u) . u \in \#\right.\) ? \(\left.L^{\prime} \#\right\}\)
    have ? \(\mathrm{M1}^{\prime}=\) ? \(\mathrm{M2}^{\prime}\)
    by (metis (mono-tags, lifting) mem-Collect-eq multiset.map-cong0 set-mset-filter)
    let ?M1" \(=\left\{\#(f 1 u) \cdot u \in \# ? L^{\prime \prime} \#\right\}\)
    let ?M2" \(=\left\{\#(f 2 u) \cdot u \in \# ? L^{\prime \prime} \#\right\}\)
    from \(\left\langle L=? L^{\prime}+? L^{\prime \prime}\right\rangle\) have \(M 1=? M 1^{\prime}+? M 1^{\prime \prime}\) by (metis assms(1) im-
age-mset-union)
    from \(\left\langle L=? L^{\prime}+? L^{\prime \prime}\right\rangle\) have \(M 2=? M 2^{\prime}+? M 2^{\prime \prime}\) by (metis assms(2) im-
age-mset-union)
```

```
have dom: \((\forall k \in\) set-mset ?M1'". \(\exists j \in\) set-mset ?M2". \((k, j) \in r)\)
proof
    fix \(k\) assume \(k \in\) set-mset ?M1"
    from this obtain \(u\) where \(k=(f 1 u)\) and \(u \in \#\) ? \(L^{\prime \prime}\) by auto
    from \(\left\langle u \in \#\right.\) ? \(\left.L^{\prime \prime}\right\rangle\) have ( \(f 2 u\) ) \(\in \#\) ?M2 \({ }^{\prime \prime}\) by simp
    from \(\left\langle\forall u .\left(u \in \# ? L^{\prime \prime} \longrightarrow((f 1 u),(f 2 u)) \in r\right)\right\rangle\) and \(\left\langle u \in \# ? L^{\prime \prime}\right\rangle\)
        have \(((f 1 u),(f 2 u)) \in r\) by auto
    from this and \(\langle k=(f 1 u)\rangle\) and \(\left\langle(f 2 u) \in\right.\) set-mset ?M2 \(\left.{ }^{\prime \prime}\right\rangle\)
        show \(\exists j \in\) set-mset ?M2 \({ }^{\prime \prime} .(k, j) \in r\) by auto
    qed
    have \(? L^{\prime \prime} \neq\{\#\}\)
    proof -
    from \(\operatorname{assms}(4)\) obtain \(u\) where \(u \in \# L\) and \(((f 1 u),(f 2 u)) \in r\) by auto
    from \(\operatorname{assms}(5)\langle((f 1 u),(f 2 u)) \in r\rangle\) have \(((f 1 u) \neq(f 2 u))\)
        unfolding irrefl-def by fastforce
    from \(\langle u \in \# L\rangle\langle((f 1 u) \neq(\) f2 \(u))\rangle\) have \(u \in \# ? L^{\prime \prime}\) by auto
    from this show ?thesis by force
qed
from this have \(?^{M 2^{\prime \prime}} \neq\{\#\}\) by auto
from this and dom and \(\left\langle M 1=? M 1^{\prime}+? M 1^{\prime \prime}\right\rangle\left\langle M 2=? M 2^{\prime}+? M 2^{\prime \prime}\right\rangle\left\langle ? M 1^{\prime}=? M 2^{\prime}\right\rangle\)
    show \((\) M1,M2 \() \in(\) mult \(r)\) by (simp add: one-step-implies-mult \()\)
qed
lemma image-mset-ordering-eq:
    assumes \(M 1=\{\#(f 1 u) . u \in \# L \#\}\)
    assumes \(M 2=\{\#(f 2 u) \cdot u \in \# L \#\}\)
    assumes \(\forall u .(u \in \# L \longrightarrow(((f 1 u),(f 2 u)) \in r \vee(f 1 u)=(f 2 u)))\)
    shows \((\) M1 \(=\) M2) \(\vee((\) M1, M2 \() \in(\) mult \(r))\)
proof (cases)
    assume \(M 1=M 2\) then show ?thesis by auto
    next assume \(M 1 \neq M 2\)
    let \(? L^{\prime}=\{\# u \in \# L .(f 1 u)=(f 2 u) \#\}\)
    let \({ }^{\prime} L^{\prime \prime}=\{\# u \in \# L . \quad(f 1 u) \neq(f 2 u) \#\}\)
    have \(L=? L^{\prime}+? L^{\prime \prime}\) by (simp)
    from \(\operatorname{assms}(3)\) have \(\forall u .\left(u \in \# ? L^{\prime \prime} \longrightarrow((f 1 u),(f 2 u)) \in r\right)\) by auto
    let \(? M 1^{\prime}=\left\{\#(f 1 u) . u \in \# ? L^{\prime} \#\right\}\)
    let ? M \(^{\prime}{ }^{\prime}=\left\{\#(f 2 u) \cdot u \in \#\right.\) ? \(\left.L^{\prime} \#\right\}\)
    have ? \(\mathrm{M1}^{\prime}=\) ? \(\mathrm{MQ}^{\prime}\)
    by (metis (mono-tags, lifting) mem-Collect-eq multiset.map-cong0 set-mset-filter)
    let ? \({ }^{\prime \prime} 1^{\prime \prime}=\left\{\#(f 1 u), u \in \# ? L^{\prime \prime} \#\right\}\)
    let ? \(^{\prime 2}\) ' \(^{\prime \prime}=\left\{\#(f 2 u), u \in \#\right.\) ? \(\left.L^{\prime \prime} \#\right\}\)
    from \(\left\langle L=? L^{\prime}+? L^{\prime \prime}\right\rangle\) have \(M 1=? M 1^{\prime}+? M 1^{\prime \prime}\) by (metis assms(1) im-
age-mset-union)
    from \(\left\langle L=? L^{\prime}+? L^{\prime \prime}\right\rangle\) have \(M 2=? M 2^{\prime}+?\) 2 \(^{\prime \prime}\) by (metis assms(2) im-
age-mset-union)
    have dom: \((\forall k \in\) set-mset ?M1'". \(\exists j \in\) set-mset ?M2'". \((k, j) \in r)\)
    proof
    fix \(k\) assume \(k \in\) set-mset ?M1"
```

from this obtain $u$ where $k=(f 1 u)$ and $u \in \# ? L^{\prime \prime}$ by auto
from $\left\langle u \in \# ? L^{\prime \prime}\right\rangle$ have $(f 2 u) \in \#$ ?M2 ${ }^{\prime \prime}$ by simp
from $\left\langle\forall u .\left(u \in \# ? L^{\prime \prime} \longrightarrow((f 1 u),(f 2 u)) \in r\right)\right\rangle$ and $\left\langle u \in \# ? L^{\prime \prime}\right\rangle$
have $((f 1 u),(f 2 u)) \in r$ by auto
from this and $\langle k=(f 1 u)\rangle$ and $\left\langle(f 2 u) \in\right.$ set-mset ?M2 $\left.{ }^{\prime \prime}\right\rangle$
show $\exists j \in$ set-mset ?M2'. $(k, j) \in r$ by auto
qed
from $\langle M 1 \neq M 2\rangle$ have ? $^{\prime 2} 2^{\prime \prime} \neq\{\#\}$
using $\langle M 1=$ image-mset f1 $\{\# u \in \# L . f 1 u=$ f2 $u \#\}+$ image-mset $f 1\{\#$ $u \in \# L . f 1 u \neq f 2 u \#\}\rangle\langle M 2=$ image-mset f2 $\{\# u \in \# L$. f1 $u=$ f2 $u \#\}+$ image-mset f2 $\{\# u \in \# L . f 1 u \neq f 2 u \#\}\rangle\left\langle i m a g e-m s e t f 1\left\{\# u \in \# L . f 1 u=f_{2}\right.\right.$ $u \#\}=$ image-mset f2 $\{\# u \in \# L . f 1 u=f 2 u \#\}\rangle$ by auto from this and dom and $\left\langle M 1=? M 1^{\prime}+? M 1^{\prime \prime}\right\rangle\left\langle M 2=? M 2^{\prime}+? M 2^{\prime \prime}\right\rangle\left\langle ? M 1^{\prime}=? M 2^{\prime}\right\rangle$
have $($ M1, M2 $) \in($ mult $r)$ by (simp add: one-step-implies-mult $)$
from this show ?thesis by auto
qed
lemma mult1-def-lemma :
assumes $M=M 0+\{\# a \#\} \wedge N=M 0+K \wedge(\forall b . b \in \# K \longrightarrow(b, a) \in r)$ shows $(N, M) \in($ mult1 $r)$
proof -
from assms(1) show ?thesis using mult1-def [of r] by auto
qed
lemma mset-ordering-add1:
assumes $(E 1, E 2) \in($ mult $r)$
shows $(E 1, E 2+\{\# a \#\}) \in($ mult $r)$
proof -
have $i:(E 2, E 2+\{\# a \#\}) \in($ mult1 $r)$ using mult1-def-lemma $[$ of E2 $+\{\#$ a \#\} E2 a E2 \{\#\} r] by auto
have $i:(E 2, E 2+\{\# a \#\}) \in($ mult1 $r)$ using mult1-def-lemma $[$ of E2 $+\{\#$ a \#\} E2 a E2 \{\#\} r]
by auto
from assms(1) have $(E 1, E 2) \in(\text { mult1 } r)^{+}$using mult-def by auto
from this and $i$ have $(E 1, E 2+\{\# a \#\}) \in(\text { mult1 } r)^{+}$by auto
then show ?thesis using mult-def by auto
qed
lemma mset-ordering-singleton:
assumes $\forall x .(x \in \# E 1 \longrightarrow(x, a) \in r)$
shows $(E 1,\{\# a \#\}) \in($ mult $r)$
proof -
let $? K=E 1$
let $? M 0=\{\#\}$
have $i: E 1=? M 0+? K$ by auto
have $i i$ : $\{\# a \#\}=$ ? $M 0+\{\# a \#\}$ by auto
from $\operatorname{assms}(1)$ have iii: $\forall x .(x \in \# ? K \longrightarrow(x, a) \in r)$ by auto
from $i$ and $i i$ and iii show ?thesis using mult1-def-lemma $[o f\{\# a \#\}$ ?M0 a E1 ? K r] mult-def by auto
qed
lemma monotonic-fun-mult1:
assumes $\bigwedge t s .((t, s) \in r \Longrightarrow((f t),(f s)) \in r)$
assumes $(E 1, E 2) \in($ mult1 $r)$
shows $(\{\#(f x) . x \in \# E 1 \#\},\{\#(f x) . x \in \# E 2 \#\}) \in($ mult1 $r)$
proof -
let ? $E 1=\{\#(f x) \cdot x \in \# E 1 \#\}$
let $? E 2=\{\#(f x) \cdot x \in \# E 2 \#\}$
from $\operatorname{assms}(2)$ obtain $M 0$ a $K$ where $E 2=M 0+\{\# a \#\}$ and $E 1=M 0+$
$K$ and $(\forall b . b \in \# K \longrightarrow(b, a) \in r)$
unfolding mult1-def [of $r$ ] by auto
let $? K=\{\#(f x) . x \in \# K \#\}$
let $? M 0=\{\#(f x), x \in \# M 0 \#\}$
from $\langle E 2=M 0+\{\# a \#\}\rangle$ have $? E 2=? M 0+\{\#(f a) \#\}$ by simp
from $\langle E 1=M 0+K\rangle$ have ? $E 1=? M 0+? K$ by simp
have $(\forall b . b \in \#$ ? $K \longrightarrow(b,(f a)) \in r)$
proof ((rule allI),(rule impI))
fix $b$ assume $b \in \#$ ? $K$
from $\langle b \in \# ? K\rangle$ obtain $b^{\prime}$ where $b=\left(f b^{\prime}\right)$ and $b^{\prime} \in \# K$
by (auto simp: insert-DiffM2 msed-map-invR union-single-eq-member)
from $\left\langle b^{\prime} \in \# K\right\rangle$ and $\langle(\forall b . b \in \# K \longrightarrow(b, a) \in r)\rangle$ have $\left(b^{\prime}, a\right) \in r$ by auto
from $\operatorname{assms}(1)$ and this and $\left\langle b=\left(f b^{\prime}\right)\right\rangle$ show $(b,(f a)) \in r$ by auto
qed
from $\langle ? E 1=? M 0+? K\rangle$ and $\langle ? E 2=? M 0+\{\#(f a) \#\}\rangle$ and $\langle(\forall b . b \in \#$ $? K \longrightarrow(b,(f a)) \in r)$ 〉
show $(? E 1, ? E 2) \in($ mult1 $r)$ by (metis mult1-def-lemma)
qed
lemma monotonic-fun-mult:
assumes $\bigwedge t s .((t, s) \in r \Longrightarrow((f t),(f s)) \in r)$
assumes (E1,E2) $\in$ (mult r)
shows $(\{\#(f x) . x \in \# E 1 \#\},\{\#(f x) . x \in \# E 2 \#\}) \in($ mult $r)$
proof -
let ? $E 1=\{\#(f x) . x \in \# E 1 \#\}$
let $?$ E2 $=\{\#(f x), x \in \# E 2 \#\}$
let $? P=\lambda x$. $(? E 1,\{\#(f y) . y \in \# x \#\}) \in($ mult $r)$
show ?thesis
proof (rule trancl-induct [of E1 E2 (mult1 r) ?P])
from $\operatorname{assms}(1)$ show $(E 1, E 2) \in(\text { mult1 } r)^{+}$using assms(2) mult-def by blast
next
fix $x$ assume $(E 1, x) \in$ mult1 $r$
have (image-mset $f$ E1, image-mset $f x$ ) $\in$ mult1 $r$
by ( simp add: $\langle(E 1, x) \in$ mult1 $r\rangle$ assms (1) monotonic-fun-mult1)
from this show (image-mset $f$ E1, image-mset $f x$ ) $\in$ mult $r$ by (simp add: mult-def)

```
    next
    fix }xz\mathrm{ assume (E1,x) ( mult1 r)+
        (x,z) & mult1 r and (image-mset f E1, image-mset f x) \in mult r
    from «(x,z)\in mult1 r〉 have (image-mset f x, image-mset f z) \in mult1 r
        by (simp add: assms(1) monotonic-fun-mult1)
    from this and «(image-mset f E1, image-mset f x) \in mult r>
        show (image-mset f E1, image-mset fz)\in mult r
        using mult-def trancl.trancl-into-trancl by fastforce
    qed
qed
lemma mset-set-insert-eq:
    assumes finite E
    shows mset-set }(E\cup{x})\subseteq# mset-set E + {# x #
proof (rule ccontr)
    assume \neg?thesis
    from this obtain y where (count (mset-set (E\cup{x})) y)
        > (count (mset-set E + {# x#}) y)
        by (meson leI subseteq-mset-def)
    from assms(1) have finite ( }E\cup{x})\mathrm{ by auto
    have (count (mset-set E + {# x #}) y) = (count (mset-set E) y) + (count {#
x#} y) by auto
    have }x\not=
    proof
        assume }x=
        then have }y\inE\cup{x}\mathrm{ by auto
    from <finite (E\cup{x })> this have (count (mset-set (E\cup{x})) y)=1
            using count-mset-set(1) by auto
        from this and «(count (mset-set }(E\cup{x}))y)>(\mathrm{ count (mset-set E + {#
x#}) y)> have
            (count (mset-set E + {# x #}) y) = 0 by auto
            from <(count (mset-set E + {# x #}) y) = 0> have count {# x #} y = 0
by auto
    from }\langlex=y\rangle\mathrm{ have count {#x#} y = 1 using count-mset-set by auto
    from this and <count {#x#} y=0\rangle show False by auto
    qed
    have y&E
    proof
    assume y\inE
    then have }y\inE\cup{x}\mathrm{ by auto
    from<finite (E\cup{x})\rangle this have (count (mset-set (E\cup{x})) y)=1
        using count-mset-set(1) by auto
    from this and «(count (mset-set (E\cup{x })) y)> (count (mset-set E + {#
x #}) y)> have
        (count (mset-set E + {#x#}) y)=0 by auto
    from〈(count (mset-set E +{# x#}) y)=0\rangle have count (mset-set E) y=
O by (simp split: if-splits)
    from }\langley\inE\rangle\langlefinite E\rangle have count (mset-set E) y=1 using count-mset-set(1
by auto
```

from this and «count（mset－set E）$y=0$ 〉 show False by auto
qed
from this and $\langle x \neq y\rangle$ have $y \notin E \cup\{x\}$ by auto
from this have（count（mset－set $(E \cup\{x\})) y)=0$ by auto
from this and $\langle(\operatorname{count}($ mset－set $(E \cup\{x\})) y)$
$>($ count $($ mset－set $E+\{\# x \#\}) y)\rangle$ show False by auto
qed
lemma mset－set－insert：
assumes $x \notin E$
assumes finite $E$
shows mset－set $(E \cup\{x\})=$ mset－set $E+\{\# x \#\}$
proof（rule ccontr）
assume $\neg$ ？thesis
from this obtain $y$ where（count（mset－set $(E \cup\{x\})) y$ ） $\neq($ count（mset－set $E+\{\# x \#\})$ ）by（meson multiset－eqI）
have $($ count $($ mset－set $E+\{\# x \#\}) y)=(\operatorname{count}($ mset－set $E) y)+($ count $\{\#$ $x \#\} y)$ by auto
from $\operatorname{assms}(2)$ have finite $(E \cup\{x\})$ by auto
have $x \neq y$
proof
assume $x=y$
then have $y \in E \cup\{x\}$ by auto
from 〈finite $(E \cup\{x\})$ 〉 this have $($ count $(\operatorname{mset}-$ set $(E \cup\{x\})) y)=1$ using count－mset－set（1）by auto
from $\langle x=y\rangle$ have count $\{\# x \#\} y=1$ using count－mset－set by auto
from $\langle x=y\rangle\langle x \notin E\rangle$ have（count（mset－set E）y）$=0$ using count－mset－set by auto
from $\langle$ count $\{\# x \#\} y=1\rangle\langle($ count $($ mset－set $E) y)=0\rangle$
$\langle($ count $($ mset－set $E+\{\# x \#\}) y)=(\operatorname{count}($ mset－set $E) y)+($ count $\{\# x$ \＃\} $y$ ）＞
have（count（mset－set $E+\{\# x \#\}) y)=1$ by auto
from this and $\langle($ count（mset－set $(E \cup\{x\})) y)=1\rangle$ and $\langle($ count（mset－set $(E \cup\{x\})) y)$
$\neq($ count $($ mset－set $E+\{\# x \#\}) y)\rangle$ show False by auto
qed
from $\langle x \neq y\rangle$ have count $\{\# x \#\} y=0$ using count－mset－set by auto
have $y \notin E$
proof
assume $y \in E$
then have $y \in E \cup\{x\}$ by auto
from 〈finite $(E \cup\{x\})$ 〉 this have（count（mset－set $(E \cup\{x\})) y)=1$
using count－mset－set（1）by auto
from $\operatorname{assms}(\mathcal{2})\langle y \in E\rangle$ have $($ count $($ mset－set $E) y)=1$ using count－mset－set by auto
from $\langle$ count $\{\# x \#\} y=0\rangle\langle($ count $($ mset－set $E) y)=1\rangle$
$\prec($ count $($ mset－set $E+\{\# x \#\}) y)=($ count $($ mset－set $E) y)+($ count $\{\# x$ \＃\} y) >
have（count（mset－set $E+\{\# x \#\}) y)=1$ by auto
from this and $\langle($ count $($ mset－set $(E \cup\{x\})) y)=1\rangle$ and $\langle($ count（mset－set $(E \cup\{x\})) y)$
$\neq($ count（mset－set $E+\{\# x \#\}) y$ ）$\rangle$ show False by auto
qed
from this and $\langle x \neq y\rangle$ have $y \notin E \cup\{x\}$ by auto
from this have（count（mset－set $(E \cup\{x\})) y)=0$ by auto
from $\langle y \notin E\rangle$ have（count（mset－set $E) y)=0$ using count－mset－set by auto
from $\langle$ count $\{\# x \#\} y=0\rangle\langle($ count（mset－set $E) y)=0\rangle$
$\langle($ count $($ mset－set $E+\{\# x \#\}) y)=($ count $($ mset－set $E) y)+($ count $\{\# x$ \＃\} y) >
have（count（mset－set $E+\{\# x \#\}) y)=0$ by auto
from this and $\langle($ count（mset－set $(E \cup\{x\})) y)=0\rangle$ and $\langle($ count（mset－set $(E \cup\{x\})) y)$
$\neq($ count $($ mset－set $E+\{\# x \#\}) y)\rangle$ show False by auto
qed
lemma mset－image－comp：
shows $\{\#(f x) \cdot x \in \#\{\#(g x) \cdot x \in \# E \#\} \#\}=\{\#(f(g x)) \cdot x \in \# E \#\}$
by（simp add：image－mset．compositionality comp－def）
lemma mset－set－mset－image：
shows $\bigwedge E$ ．card $E=N \Longrightarrow$ finite $E \Longrightarrow \operatorname{mset-set}\left(g^{\prime} E\right) \subseteq \#\{\#(g x) . x \in \#$
mset－set（E）\＃\}
proof（induction $N$ ）
case 0
assume card $E=0$
assume finite $E$
from this and «card $E=0\rangle$ have $E=\{ \}$ by auto
then show mset－set $(g ' E) \subseteq \#\{\#(g x) . x \in \#$ mset－set $(E) \#\}$ by auto next

```
    case (Suc N)
```

        assume card \(E=(\) Suc \(N)\)
        assume finite \(E\)
    from this and \(\langle\) card \(E=(\) Suc \(N)\rangle\) have \(E \neq\{ \}\) by auto
    from this obtain \(x\) where \(x \in E\) by auto
    let \(? E=E-\{x\}\)
    from 〈finite \(E\rangle\langle\) card \(E=(\) Suc \(N)\rangle\) and \(\langle x \in E\rangle\) have card \(? E=N\) by auto
    from 〈finite \(E\rangle\) have finite ? \(E\) by auto
    from this and Suc.IH \([\) of ? \(E]\) and 〈card \(? E=N\rangle\)
            have ind: mset-set \((g\) '? \(E) \subseteq \#\{\#(g x) . x \in \#\) mset-set (?E) \#\} by force
    from \(\langle x \in E\rangle\) have \(E=? E \cup\{x\}\) by auto
    have \(x \notin\) ? \(E\) by auto
    from 〈finite ? \(E\rangle\langle E=? E \cup\{x\}\rangle\) and \(\langle x \notin ? E\rangle\) have mset-set \((? E \cup\{x\})\)
    $=$ mset-set $? E+\{\# x \#\}$
using mset-set-insert [of $x$ ? $E$ ] by auto
from this have
$\{\#(g x) \cdot x \in \#$ mset-set $(? E \cup\{x\}) \#\}=\{\#(g x) \cdot x \in \#$ mset-set ? $E \#\}$
$+\{\#(g x) \#\}$
by auto
have $(g '(? E \cup\{x\})=(g ' ? E) \cup\{g x\})$ by auto
from this have $i$ : mset-set $(g '(? E \cup\{x\}))=\operatorname{mset}-\operatorname{set}((g ' ? E) \cup\{g x\}$ ) by auto
from〈finite ? $E$ 〉 have finite ( $g$ '? $E$ ) by auto
from this have mset-set $\left(\left(g^{\prime} ? E\right) \cup\{g x\}\right) \subseteq \#$ mset-set $\left(g^{\prime} ? E\right)+\{\#(g$ x) \#\}
using mset-set-insert-eq $[o f(g$ ' $e \mathrm{E})(g x)]$ by meson
from this $i$ have $i i$ : mset-set $(g '(? E \cup\{x\})) \subseteq \#$ mset-set $(g \prime ? E)+\{\#$ $(g x) \#\}$ by auto
from ind have mset-set $(g$ ' ? $E)+\{\#(g x) \#\} \subseteq \#\{\#(g x) . x \in \#$ mset-set $(? E) \#\}+\{\#(g x) \#\}$
using Multiset.subset-mset.add-right-mono by metis
from this and $i i$ have mset-set $(g ‘(? E \cup\{x\})) \subseteq \#\{\#(g x) . x \in \#$ mset-set $(? E) \#\}+\{\#(g x) \#\}$
using subset-mset.trans $[$ of mset-set $(g$ ' $(? E \cup\{x\}))]$ by metis
from this and $\langle E=? E \cup\{x\}\rangle\langle\{\#(g x) . x \in \# \operatorname{mset}$-set $(? E \cup\{x\}) \#\}$ $=\{\#(g x) \cdot x \in \#$ mset-set ? $E \#\}+\{\#(g x) \#\}\rangle$
show mset-set $\left(g^{\prime} E\right) \subseteq \#\{\#(g x) . x \in \#$ mset-set $E \#\}$
by metis
qed
lemma split-mset-set:
assumes $C=C 1 \cup C 2$
assumes $C 1 \cap C 2=\{ \}$
assumes finite C1
assumes finite C2
shows $($ mset-set $C)=($ mset-set C1 $)+($ mset-set C2 $)$
proof (rule ccontr)
assume $($ mset-set $C) \neq($ mset-set C1 $)+($ mset-set C2)
then obtain $x$ where count (mset-set $C) x \neq$ count $(($ mset-set $C 1)+($ mset-set C2)) $x$
by (meson multiset-eqI)
from assms(3) assms(4) assms(1) have finite $C$ by auto
have count $(($ mset-set C1 $)+($ mset-set C2 $)) x=($ count $($ mset-set C1 $) x)+$ (count (mset-set C2) $x$ )
by auto
from this and $<$ count (mset-set $C) x \neq \operatorname{count}(($ mset-set C1) $+($ mset-set C2) $)$ $x^{\prime}$ have
count (mset-set $C) x \neq($ count (mset-set C1) $x)+($ count $($ mset-set C2) $x)$ by auto
have $x \in C 1 \vee x \in C 2$
proof (rule ccontr)
assume $\neg(x \in C 1 \vee x \in C 2)$
then have $x \notin C 1$ and $x \notin C 2$ by auto
from $\operatorname{assms}(1)\langle x \notin C 1\rangle$ and $\langle x \notin C 2\rangle$ have $x \notin C$ by auto
from $\langle x \notin C 1\rangle$ have (count (mset-set C1) $x$ ) $=0$ by auto
from $\langle x \notin C 2\rangle$ have (count (mset-set C2) $x$ ) $=0$ by auto
from $\langle x \notin C\rangle$ have (count (mset-set $C$ ) $x$ ) $=0$ by auto

```
    from «(count (mset-set C1) x)=0〉 «(count (mset-set C2) x)=0`
        `(count (mset-set C) x) = 0`
    <count (mset-set C) x = (count (mset-set C1) x) + (count (mset-set C2) x)`
    show False by auto
qed
    have( }x\not\inC1\veex\inC2
    proof (rule ccontr)
    assume }\neg(x\not\inC1\veex\inC2
    then have x\inC1 and x\not\inC2 by auto
    from assms(1) <x\inC1> have x\inC by auto
    from assms(3) <x\inC1` have (count (mset-set C1) x)=1 by auto
    from }\langlex\not\inC2\rangle\mathrm{ have (count (mset-set C2) x)=0 by auto
    from assms(3) assms(4) assms(1) have finite C by auto
    from <finite C\rangle\langlex\inC\rangle have (count (mset-set C) x)=1 by auto
    from «(count (mset-set C1) x)= 1> <(count (mset-set C2) x)=0>
        \(count (mset-set C) x)= 1>
        <count (mset-set C) x ( count (mset-set C1) x) + (count (mset-set C2) x)`
        show False by auto
qed
have ( }x\not\inC2\veex\inC1
proof (rule ccontr)
    assume }\neg(x\not\inC2\veex\inC1
    then have }x\inC2\mathrm{ and }x\not\inC1\mathrm{ by auto
    from assms(1) <x\inC2` have x\inC by auto
    from assms(4)<x\inC2` have (count (mset-set C2) x)=1 by auto
    from }\langlex\not\inC1\rangle have (count (mset-set C1) x)=0 by auto
    from \langlefinite C\rangle\langlex\inC\rangle have (count (mset-set C) x)=1 by auto
    from «(count (mset-set C2) x)=1`«(count (mset-set C1) x)=0`
        <(count (mset-set C) x)=1>
        «count (mset-set C) x = (count (mset-set C1) x) + (count (mset-set C2) x)`
        show False by auto
    qed
```



```
    have }x\inC1\wedgex\inC2 by blas
    from this and assms(2) show False by auto
qed
lemma image-mset-thm:
    assumes E={# (fx).x\in# E'#}
    assumes }x\in#
    shows \exists y. ((y\in# E')^x=(fy))
using assms by auto
lemma split-image-mset:
    assumes M=M1+M2
```



```
#}
by (simp add: assms)
```


## end

theory well-founded-continued

## imports Main

begin

### 1.2 Well-Founded Sets

Most useful lemmata are already proven in the Well_Founded theory available in Isabelle. We only establish a few convenient results for constructing well-founded sets and relations.

```
lemma measure-wf:
    assumes \(w f\left(r::\left({ }^{\prime} a \times{ }^{\prime} a\right)\right.\) set \()\)
    assumes \(r^{\prime}=\{(x, y) .((m x),(m y)) \in r\}\)
    shows \(w f r^{\prime}\)
proof -
    have \(\left(\forall Q:::^{\prime} b\right.\) set. \(\forall x:: \quad\) ' \(\left.b . x \in Q \longrightarrow\left(\exists z \in Q . \forall y .(y, z) \in r^{\prime}-->y \notin Q\right)\right)\)
    proof \(((\) rule allI \()+,(\) rule impI \())\)
        fix \(Q:\) : ' \(b\) set fix \(x::{ }^{\prime} b\) assume \(x \in Q\)
        let ? \(Q^{\prime}=\left(m^{\prime} Q\right)\)
    from \(\langle x \in Q\rangle\) have \(Q^{\prime}\)-not-empty: \(m x \in\) ? \(Q^{\prime}\) by auto
    from \(\operatorname{assms}(1)\) and \(Q^{\prime}\)-not-empty obtain \(z^{\prime}\) where \(z^{\prime} \in ? Q^{\prime}\) and \(z^{\prime}\) min: \(\forall y\).
\(\left(y, z^{\prime}\right) \in r\)
            \(\longrightarrow y \notin ? Q^{\prime}\) using wf-eq-minimal [of \(r\) ] by blast
    from \(\left\langle z^{\prime} \in ? Q^{\prime}\right\rangle\) obtain \(z\) where \(z^{\prime}=(m z)\) and \(z \in Q\) by auto
    have \(\forall y .(y, z) \in r^{\prime} \longrightarrow y \notin Q\)
    proof ((rule allI),(rule impI))
            fix \(y\) assume \((y, z) \in r^{\prime}\)
            from assms(2) and this and \(\left\langle z^{\prime}=(m z)\right\rangle\) have \(\left((m y), z^{\prime}\right) \in r\) by auto
            from this and \(z^{\prime} \min\) have \((m y) \notin ? Q^{\prime}\) by auto
            then show \(y \notin Q\) by auto
            qed
            from this and \(\langle z \in Q\rangle\) show \(\left(\exists z \in Q . \forall y .(y, z) \in r^{\prime}-->y \notin Q\right)\) by auto
    qed
    then show ?thesis using wf-eq-minimal by auto
qed
lemma finite-proj-wf:
    assumes finite \(E\)
    assumes \(x \in E\)
    assumes acyclic \(r\)
    \(\operatorname{shows}(\exists y . y \in E \wedge(\forall z .(z, y) \in r \longrightarrow z \notin E))\)
proof -
    let \(? r=\{(u, v) .(u \in E \wedge v \in E \wedge(u, v) \in r)\}\)
    from \(\operatorname{assms}(1)\) have finite \((E \times E)\) by auto
```

```
    have ?r}\subseteq\subseteq(E\timesE)\mathrm{ by auto
    have ?}r\subseteqr\mathrm{ by auto
    from〈?r \subseteq (E < E) > and<finite (E\timesE)\rangle have finite ?r using finite-subset
by auto
    from assms(3) and 〈?r \subseteq r〉 have acyclic ?r unfolding acyclic-def using
trancl-mono by blast
    from <acyclic ?r><finite ?r> have wf ?r using finite-acyclic-wf by auto
    from this assms(2) obtain y where y}\inE\mathrm{ and i: \z. (z,y) f?r >z#E
    using wfE-min [of ?r x E] by blast
    have }\forallz.(z,y)\inr\longrightarrowz\not\in
    proof (rule allI,rule impI)
    fix z assume (z,y)\inr
    show z&E
    proof
            assume z\inE
            from this and }\langley\inE\rangle\mathrm{ and }\langle(z,y)\inr\rangle\mathrm{ have (z,y) E?r by auto
            from this and i[of z] and }\langlez\inE\rangle\mathrm{ show False by auto
        qed
    qed
    from this and }\langley\inE\rangle\mathrm{ show ?thesis by auto
qed
end
theory terms
```

imports $H O L-e x . U n i f i c a t i o n$
begin

## 2 Terms

## 2．1 Basic Syntax

We use the same term representation as in the Unification theory provided in Isabelle．Terms are represented by binary trees built on variables and constant symbols．
fun is－a－variable
where
$($ is-a-variable $($ Var $x))=$ True $\mid$
$($ is-a-variable $($ Const $x))=$ False
$($ is-a-variable $($ Comb $x y))=$ False
fun $i s$－$a$－constant
where

```
(is-a-constant (Var x)) = False 
(is-a-constant (Const x)) = True 
```

```
    (is-a-constant (Comb x y)) = False
```

fun is-compound

## where

$$
\begin{aligned}
& (\text { is-compound }(\text { Var } x))=\text { False } \mid \\
& (\text { is-compound }(\text { Const } x))=\text { False } \mid \\
& (\text { is-compound }(\text { Comb } x y))=\text { True }
\end{aligned}
$$

definition ground-term :: 'a trm $\Rightarrow$ bool

## where

```
    (ground-term t)}=(\mathrm{ vars-of t }={}
```

lemma constants-are-not-variables :
assumes is-a-constant $x$
shows $\neg(i s-a$-variable $x)$
by (metis assms is-a-constant.elims(2) is-a-variable.elims(2) trm.distinct(2))
lemma constants-are-ground :
assumes is-a-constant $x$
shows ground-term $x$
proof -
from assms obtain $y$ where $x=$ (Const $y$ ) using is-a-constant.elims(2) by
auto
then show ?thesis by (simp add: ground-term-def)
qed

### 2.2 Positions

We define the notion of a position together with functions to access to subterms and replace them. We establish some basic properties of these functions.

Since terms are binary trees, positions are sequences of binary digits.

```
datatype indices = Left | Right
```

type-synonym position $=$ indices list
fun left-app
where left-app $x=$ Left $\# x$
fun right-app
where right-app $x=$ Right $\# x$
definition strict-prefix
where

$$
\text { strict-prefix p } q=(\exists r .(r \neq[]) \wedge(q=(\text { append } p r)))
$$

fun subterm $::$ 'a trm $\Rightarrow$ position $\Rightarrow$ 'a trm $\Rightarrow$ bool where

```
(subterm T [] S) = (T=S)|
(subterm (Var v) (first # next) S)= False |
(subterm (Const c) (first # next) S)= False |
(subterm (Comb x y) (Left # next) S) = (subterm x next S)|
(subterm (Comb x y) (Right # next) S)=(subterm y next S)
```


## definition occurs-in :: 'a trm $\Rightarrow{ }^{\prime}$ a trm $\Rightarrow$ bool

    where
        occurs-in \(t s=(\exists\) p. subterm s \(p t)\)
    definition position-in $::$ position $\Rightarrow{ }^{\prime}$ a trm $\Rightarrow$ bool
where
position-in ps$=(\exists t$. subterm spt)
fun subterms-of
where
subterms-of $t=\{s$. (occurs-in st) $\}$
fun proper-subterms-of
where
proper-subterms-of $t=\{s . \exists p .(p \neq$ Nil $\wedge($ subterm $t p s))\}$
fun pos-of
where
pos-of $t=\{p .($ position-in $p t)\}$
fun replace-subterm ::
'a trm $\Rightarrow$ position $\Rightarrow$ 'a trm $\Rightarrow$ 'a trm $\Rightarrow$ bool
where
(replace-subterm $T[] u S)=(S=u) \mid$
(replace-subterm (Var x) (first \# next) u S)=False |
(replace-subterm (Const c) (first \# next) u S) = False
(replace-subterm (Comb x y) (Left \# next) u S) =
$(\exists$ S1. (replace-subterm x next u S1) $\wedge(S=$ Comb S1 y) ) |
(replace-subterm (Comb x y) (Right \# next) uS)=
$(\exists$ S2. (replace-subterm y next u S2 $) \wedge(S=$ Comb x S2 $)$ )
lemma replace-subterm-is-a-function:
shows $\bigwedge t u v$. subterm $t p u \Longrightarrow \exists$ s. replace-subterm t pvs
proof (induction p,auto)
next case (Cons i q)
from $\langle$ subterm $t($ Cons $i q) u\rangle$ obtain $t 1$ t2 where $t=($ Comb t1 t2 $)$
using subterm.elims(2) by blast
have $i=$ Right $\vee i=$ Left using indices.exhaust by auto
then show ?case
proof
assume $i=$ Right
from this and $\langle t=($ Comb t1 t2) $\rangle$ and $\langle s u b t e r m ~ t($ Cons $i q) u\rangle$ have subterm t2 $q u$ by auto
from this obtain $s$ where replace-subterm t2 $q v s$ using Cons.IH [of t2 $u$ ] by auto
from this and $\langle t=($ Comb t1 t2 $)\rangle$ and $\langle i=$ Right $\rangle$ have replace-subterm $t$ (Cons iq) $v($ Comb t1 $s)$
by auto
from this show ?case by auto
next assume $i=$ Left
from this and $\langle t=($ Comb t1 t2) $\rangle$ and $\langle$ subterm $t($ Cons $i q) u\rangle$ have subterm t1 qu by auto
from this obtain $s$ where replace-subterm t1 qvs using Cons.IH [of t1 $u$ ] by auto
from this and $\langle t=($ Comb t1 t2) $\rangle$ and $\langle i=$ Left $\rangle$ have replace-subterm $t$ (Cons iq) v(Comb st2)
by auto
from this show? ?ase by auto
qed
qed
We prove some useful lemmata concerning the set of variables or subterms occurring in a term.
lemma root-subterm:
shows $t \in$ (subterms-of $t)$
by (metis mem-Collect-eq occurs-in-def subterm.simps(1) subterms-of.simps)
lemma root-position:
shows Nil $\in$ (pos-of $t$ )
by (metis mem-Collect-eq subterm.simps(1) position-in-def pos-of.simps)
lemma subterms-of-an-atomic-term:
assumes is-a-variable $t \vee i s$-a-constant $t$
shows subterms-of $t=\{t\}$
proof
show subterms-of $t \subseteq\{t\}$
proof
fix $x$ assume $x \in$ subterms-of $t$
then have occurs-in $x t$ by auto
then have $\exists p$. (subterm t $p x)$ unfolding occurs-in-def by auto
from this and assms have $x=t$ by (metis is-a-constant.simps(3) is-a-variable.simps(3) subterm.elims(2)) thus $x \in\{t\}$ by auto
qed
next
show $\{t\} \subseteq$ subterms-of $t$
proof fix $x$ assume $x \in\{t\}$ then show $x \in$ subterms-of $t$ using root-subterm by auto
qed
qed

```
lemma positions-of-an-atomic-term:
    assumes is-a-variable t V is-a-constant t
    shows pos-of t={Nil}
proof
    show pos-of t\subseteq{Nil }
    proof
        fix }x\mathrm{ assume }x\in\mathrm{ pos-of t
        then have position-in xt by auto
        then have }\existss\mathrm{ . (subterm tx s) unfolding position-in-def by auto
        from this and assms have x=Nil
            by (metis is-a-constant.simps(3) is-a-variable.simps(3) subterm.elims(2))
        thus }x\in{Nil} by aut
    qed
next
    show {Nil }\subseteq pos-of t
    proof
        fix x :: indices list assume }x\in{\mathrm{ Nil }
        then show }x\in\mathrm{ pos-of t using root-position by auto
    qed
qed
lemma subterm-of-a-subterm-is-a-subterm :
    assumes subterm u q v
    shows }\t\mathrm{ t. subterm t p u ב subterm t (append p q) v
proof (induction p)
    case Nil
        show ?case using Nil.prems assms by auto
    next case (Cons i p')
        from <subterm t(Cons i p')u` obtain t1 t2 where t=(Comb t1 t2)
            using subterm.elims(2) by blast
        have i= Right \veei= Left using indices.exhaust by auto
    then show ?case
    proof
                assume i = Right
                from this and <subterm t (Cons i p') u> and <t = (Comb t1 t2)>
                have subterm t2 p' u by auto
            from this have subterm t2 (append p' q) v by (simp add: Cons.IH)
            from this and }\langlet=(\mathrm{ Comb t1 t2)> and }<i=\mathrm{ Right> show subterm t (append
(Cons i p') q) v
            by simp
        next assume i= Left
            from this and <subterm t (Cons i p') u> and <t = (Comb t1 t2)>
                    have subterm t1 p' u by auto
            from this have subterm t1 (append p' q) v by (simp add: Cons.IH)
            from this and <t = (Comb t1 t2)> and }\langlei=\mathrm{ Left> show subterm t (append
(Cons i p')q) v
                by simp
            qed
qed
```

```
lemma occur-in-subterm
    assumes occurs-in ut
    assumes occurs-in t s
    shows occurs-in u s
by (meson assms(1) assms(2) occurs-in-def subterm-of-a-subterm-is-a-subterm)
lemma vars-of-subterm :
    assumes }x\in\mathrm{ vars-of s
    shows }\bigwedget\mathrm{ . subterm t p s < x vars-of t
proof (induction p)
    case Nil
        show ?case using Nil.prems assms by auto
    next case (Cons i p
        from <subterm t (Cons i p') s` obtain t1 t2 where t = (Comb t1 t2)
            using subterm.elims(2) by blast
        have i= Right \vee i = Left using indices.exhaust by auto
        then show ?case
        proof
            assume i= Right
            from this and <subterm t (Cons i p') s\rangle and «t = (Comb t1 t2)>
                have subterm t2 p's by auto
                from this have x vars-of t2 by (simp add: Cons.IH)
                from this and }\langlet=(\mathrm{ Comb t1 t2)> and }\langlei=\mathrm{ Right> show ?case
                    by simp
        next assume i= Left
            from this and <subterm t (Cons i p') s\rangle and <t = (Comb t1 t2)>
                have subterm t1 p's by auto
                from this have x vars-of t1 by (simp add: Cons.IH)
                from this and <t = (Comb t1 t2)\rangle and <i=Left\rangle show ?case
                    by simp
        qed
qed
lemma vars-subterm :
    assumes subterm t p s
    shows vars-of s\subseteq vars-of t
by (meson assms subsetI vars-of-subterm)
lemma vars-subterms-of :
    assumes }s\in\mathrm{ subterms-of }
    shows vars-of s\subseteqvars-of t
using assms occurs-in-def vars-subterm by fastforce
lemma subterms-of-a-non-atomic-term:
    shows subterms-of (Comb t1 t2) = (subterms-of t1) \cup (subterms-of t2) \cup{
(Comb t1 t2) }
proof
    show subterms-of (Comb t1 t2) \subseteq(subterms-of t1) \cup (subterms-of t2) \cup { (Comb
```

```
t1 t2) }
    proof
        fix x assume }x\in(\mathrm{ subterms-of (Comb t1 t2))
        then have occurs-in x (Comb t1 t2) by auto
        then obtain p}\mathrm{ where subterm (Comb t1 t2) px unfolding occurs-in-def by
auto
    have }p=[]\vee(\existsiq.p=i#q) using neq-Nil-conv by blas
    then show }x\in(\mathrm{ subterms-of t1) }\cup(\mathrm{ subterms-of t2) }\cup{(Comb t1 t2) }
    proof
        assume p=[]
        from this and <subterm (Comb t1 t2) p x\rangle show ?thesis by auto
    next
        assume }\existsiq.p=i#
        then obtain iq where p=i#q by auto
        have i=Left \veei= Right using indices.exhaust by blast
        then show }x\in(\mathrm{ subterms-of t1) }\cup(\mathrm{ subterms-of t2) }\cup{(Comb t1 t2) }
        proof
            assume i= Left
            from this and <p=i# q> and <subterm (Comb t1 t2) p x〉
                have subterm t1 qx}\mathrm{ by auto
            then have occurs-in x t1 unfolding occurs-in-def by auto
            then show }x\in(\mathrm{ subterms-of t1) }\cup(\mathrm{ subterms-of t2) }\cup{(Comb t1 t2) } by
auto
        next
            assume i= Right
            from this and }\langlep=i#q\rangle and «subterm (Comb t1 t2) p x〉
                have subterm t2 qx by auto
            then have occurs-in x t2 unfolding occurs-in-def by auto
            then show }x\in(\mathrm{ subterms-of t1) U(subterms-of t2) }\cup{(Comb t1 t2) } by
auto
            qed
        qed
    qed
next
    show (subterms-of t1) \cup (subterms-of t2) \cup{(Comb t1 t2) }\subseteq subterms-of
(Comb t1 t2)
    proof
            fix }x\mathrm{ assume }x\in(\mathrm{ subterms-of t1) U(subterms-of t2) U {(Comb t1 t2) }
            then have }x\in(\mathrm{ subterms-of t1) }\vee(x\in(\mathrm{ subterms-of t2) }\veex=(\mathrm{ Comb t1 t2))
by auto
    thus x 䄱terms-of (Comb t1 t2)
    proof
                assume }x\in(\mathrm{ subterms-of t1)
                then have occurs-in x t1 by auto
                then obtain p}\mathrm{ where subterm t1 px unfolding occurs-in-def by auto
                then have subterm (Comb t1 t2) (Left # p) x by auto
                then have occurs-in x (Comb t1 t2) using occurs-in-def by blast
            then show }x\in\mathrm{ subterms-of (Comb t1 t2) by auto
        next
```

```
        assume (x\in(subterms-of t2) \vee x =(Comb t1 t2))
        then show }x\in\mathrm{ subterms-of (Comb t1 t2)
        proof
            assume }x\in\mathrm{ (subterms-of t2)
            then have occurs-in x t2 by auto
            then obtain p}\mathrm{ where subterm t2 px unfolding occurs-in-def by auto
            then have subterm (Comb t1 t2) (Right # p) x by auto
            then have occurs-in x (Comb t1 t2) using occurs-in-def by blast
            then show }x\in\mathrm{ subterms-of (Comb t1 t2) by auto
            next
            assume x = (Comb t1 t2)
            show }x\in\mathrm{ subterms-of (Comb t1 t2) using <x = t1 · t2> root-subterm by
blast
            qed
                qed
    qed
qed
lemma positions-of-a-non-atomic-term:
    shows pos-of (Comb t1 t2) = (left-app'(pos-of t1)) \cup(right-app'(pos-of t2))
\cup Nil }
proof
    show pos-of (Comb t1 t2)\subseteq(left-app'(pos-of t1))\cup(right-app'(pos-of t2))
\cup \{ N i l \}
    proof
        fix x assume x fos-of (Comb t1 t2)
        then have position-in x (Comb t1 t2) by auto
        then obtain s}\mathrm{ where subterm (Comb t1 t2) xs unfolding position-in-def by
auto
        have }x=[]\vee(\existsiq.x=i#q) using neq-Nil-conv by blas
        then show }x\in(left-app'(pos-of t1))\cup(right-app'(pos-of t2)) \cup { Nil 
        proof
            assume }x=[
            from this and <subterm (Comb t1 t2) x s` show ?thesis by auto
        next
            assume }\existsiq. x=i#
            then obtain iq where x=i#q by auto
            have i=Left \vee i= Right using indices.exhaust by blast
            then show }x\in(left-app'(pos-of t1))\cup(right-app'(pos-of t2)) \cup { Nil 
            proof
                assume i= Left
                from this and <x =i# q> and «subterm (Comb t1 t2) x s`
                    have subterm t1 q s by auto
                    then have position-in q t1 unfolding position-in-def by auto
                    from this and <x =i# q><i= Left>
                    show }x\in(left-app'(pos-of t1))\cup(right-app'(pos-of t2)) \cup {Nil } by
auto
            next
                assume i= Right
```

```
            from this and }\langlex=i#q\rangle and <subterm (Comb t1 t2) x s`
                    have subterm t2 q s by auto
            then have position-in q t2 unfolding position-in-def by auto
            from this and \langlex = i# q\rangle\langlei= Right>
                show }x\in(left-app'(pos-of t1))\cup(right-app'(pos-of t2)) \cup{Nil } by
auto
            qed
        qed
    qed
next
    show (left-app'(pos-of t1)) \cup(right-app'(pos-of t2))\cup{ Nil } \subseteq pos-of (Comb
t1 t2)
    proof
        fix x assume }x\in(left-app '(pos-of t1)) \cup(right-app'(pos-of t2)) \cup{Nil 
        then have (x\inleft-app'(pos-of t1)) \vee ((x\in(right-app'(pos-of t2))) \vee (x
= Nil)) by auto
        thus }x\in\mathrm{ pos-of (Comb t1 t2)
        proof
            assume x left-app'(pos-of t1)
            then obtain q}\mathrm{ where }x=\mathrm{ Left # q and position-in q t1 by auto
            then obtain s}\mathrm{ where subterm t1 qs unfolding position-in-def by auto
            then have subterm (Comb t1 t2) (Left # q) s by auto
                    from this and <x = Left # q> have position-in x (Comb t1 t2) using
position-in-def by blast
            then show }x\in\mathrm{ pos-of (Comb t1 t2) by auto
        next
            assume (x\in(right-app'(pos-of t2))) \vee (x=Nil)
            then show }x\in\mathrm{ pos-of (Comb t1 t2)
            proof
                assume x f right-app '(pos-of t2)
            then obtain q}\mathrm{ where x= Right # q and position-in q t2 by auto
            then obtain s}\mathrm{ where subterm t2 q s unfolding position-in-def by auto
            then have subterm (Comb t1 t2) (Right # q) s by auto
                    from this and <x = Right # q> have position-in x (Comb t1 t2) using
position-in-def by blast
            then show }x\in\mathrm{ pos-of (Comb t1 t2) by auto
        next
            assume x = Nil
            show x \in pos-of (Comb t1 t2) using <x = Nil` root-position by blast
        qed
    qed
    qed
qed
lemma set-of-subterms-is-finite :
    shows (finite (subterms-of (t :: 'a trm)))
proof (induction t)
    case (Var x)
    then show ?case using subterms-of-an-atomic-term [of Var x] by simp
```

```
    next
        case (Const x)
        then show ?case using subterms-of-an-atomic-term [of Const x] by simp
    next
    case (Comb t1 t2) assume finite (subterms-of t1) and finite (subterms-of t2)
    have subterms-of (Comb t1 t2) = subterms-of t1 \cup subterms-of t2 \cup { Comb
t1 t2 }
            using subterms-of-a-non-atomic-term by auto
    from this and <finite (subterms-of t1)〉 and <finite (subterms-of t2)>
        show finite (subterms-of (Comb t1 t2)) by simp
qed
lemma set-of-positions-is-finite :
    shows (finite (pos-of (t :: 'a trm)))
proof (induction t)
    case (Var x)
    then show ?case using positions-of-an-atomic-term [of Var x] by simp
    next
        case (Const x)
        then show ?case using positions-of-an-atomic-term [of Const x] by simp
    next
        case (Comb t1 t2) assume finite (pos-of t1) and finite (pos-of t2)
    from<finite (pos-of t1)> have i: finite (left-app'(pos-of t1)) by auto
    from<finite (pos-of t2)> have ii: finite (right-app'(pos-of t2)) by auto
    have pos-of (Comb t1 t2) = (left-app'(pos-of t1)) \cup(right-app'(pos-of t2))
\cup Nil }
        using positions-of-a-non-atomic-term by metis
    from this and i ii show finite (pos-of (Comb t1 t2)) by simp
qed
lemma vars-of-instances:
    shows vars-of (subst t \sigma)
    = \bigcup{V.\existsx. (x\in (vars-of t))}\wedge(V=\operatorname{vars-of (subst (Var x) \sigma)) }
proof (induction t)
    case (Const a)
        have vars-of (Const a)={} by auto
        then have rhs-empty: \bigcup{V.\existsx.(x\in(vars-of (Const a)))}\wedge(V=\mathrm{ vars-of
(subst (Var x)\sigma)) } = {} by auto
        have lhs-empty:(subst (Const a) \sigma)=(Const a) by auto
    from rhs-empty and lhs-empty show ?case by auto
    next
    case (Var a)
    have vars-of (Var a)={a} by auto
        then have rhs: \bigcup{V.\existsx.(x\in(vars-of (Var a))) ^(V = vars-of (subst
(Var x)\sigma)) } =
            vars-of (subst (Var a) \sigma) by auto
    have lhs:(subst (Var a) \sigma)=(subst (Var a) \sigma) by auto
    from rhs and lhs show ?case by auto
    next
```

```
case (Comb t1 t2)
    have vars-of (Comb t1 t2) =(vars-of t1) \cup(vars-of t2) by auto
    then have }\bigcup{V.\existsx.(x\in(vars-of (Comb t1 t2))) ^(V=vars-of (subs
(Var x)\sigma)) }
            =\bigcup{V.\existsx.(x\in(vars-of t1))}\wedge(V=\operatorname{vars-of (subst(Var x) \sigma)) }
            \cup {V.\existsx.(x\in(vars-of t2))}\wedge(V=\operatorname{vars-of (subst (Var x)\sigma))}
            by auto
    then have rhs: \bigcup {V.\existsx.(x\in(vars-of (Comb t1 t2)))}\wedge(V=vars-o
(subst (Var x)\sigma)) }
    =(vars-of (subst t1 \sigma)) \cup(vars-of (subst t2 \sigma))
    using <vars-of (subst t1 \sigma)
                = \bigcup{V.\existsx.(x\in(vars-of t1 ))}\wedge(V=\operatorname{vars-of (subst (Var x ) \sigma)) }>
            and
                    <vars-of (subst t2 \sigma)
                    = U{V.\existsx.(x\in(vars-of t2))}\wedge(V=vars-of (subst (Var x) \sigma)) }>
    by auto
    have (subst (Comb t1 t2) \sigma)=(Comb (subst t1 \sigma) (subst t2 \sigma))
        by auto
    then have lhs: (vars-of (subst (Comb t1 t2) \sigma)) =
        (vars-of (subst t1 \sigma)) \cup(vars-of (subst t2 \sigma)) by auto
    from lhs and rhs show ?case by auto
qed
lemma subterms-of-instances :
    \foralluvu's. (u=(substvs)\longrightarrow(subterm u pu')
    \longrightarrow ( \exists x ~ q 1 ~ q 2 . ~ ( i s - a - v a r i a b l e ~ x ) ~ \wedge ( s u b t e r m ~ ( s u b s t ~ x ~ s ) ~ q 1 ~ u ' ) ~ \wedge ~
                (subterm v q2 x) ^(p=(append q2 q1))) \vee
            ((\exists v
?prop p)
proof (induction p)
    case Nil
    show ?case
    proof ((rule allI)+,(rule impI)+)
    fix u ::'a trm fix v u's assume }u=(\mathrm{ subst v s) and subterm u [] u'
    then have }u=\mp@subsup{u}{}{\prime}\mathrm{ by auto
    show (\existsx q1 q2. (is-a-variable x) ^(subterm (subst x s) q1 u')^
                (subterm v q2 x) ^([] = (append q2 q1))) \vee
            ((\exists v
    proof (cases)
    assume is-a-variable v
            from }\langleu=\mp@subsup{u}{}{\prime}>\mathrm{ and }\langleu=(\mathrm{ subst v s)>
                    have (subterm (subst v s)[] u') by auto
            have subterm v [] v by auto
            from this and <(subterm (subst vs) [] u}\mp@subsup{|}{}{\prime})\rangle\mathrm{ and 〈is-a-variable v>
                show ?thesis by auto
    next assume }\negis-a-variable 
            from }\langleu=\mp@subsup{u}{}{\prime}\rangle\mathrm{ and }\langleu=(\mathrm{ subst v s)>
            have ((subterm v [] v) ^( (u'=(subst v s))) by auto
            then show ?thesis by auto
```

```
    qed
    qed
    next
    case (Cons i q)
    show ?case
    proof ((rule allI)+,(rule impI)+)
    fix u ::'a trm fix v u's assume }u=(\mathrm{ subst v s)
        and subterm u(Cons i q) u'
    show (\existsx q1 q2. (is-a-variable x) ^(subterm (subst x s) q1 u') ^
                (subterm v q2 x ) ^((Cons i q) = (append q2 q1))) \vee
            ((\exists v}\mp@subsup{v}{}{\prime}.((\neg\mathrm{ is-a-variable v})\wedge(\mathrm{ subterm v (Cons i q) v') ^ (u' = (subst v'
s)))|)
    proof (cases v)
        fix }x\mathrm{ assume }v=(\operatorname{Var}x
        then have subterm v [] v by auto
        from }\langlev=(\operatorname{Var}x)\rangle\mathrm{ have is-a-variable v by auto
        have Cons i q = (append [] (Cons i q)) by auto
        from <subterm u(Cons iq) u'\rangle and }\langleu=(\mathrm{ subst v s)>
            and}\langlev=(\operatorname{Var}x)\rangle\mathrm{ have subterm (subst v s)(Cons i q) u' by auto
    from <is-a-variable v\rangle and <subterm v [] v\rangle and <Cons i q = (append [] (Cons
i q))> and this
        show ?thesis by blast
    next
        fix }x\mathrm{ assume v=(Const x)
        from this and }\langleu=(\mathrm{ subst vs)> have u=v by auto
        from this and }\langlev=(\mathrm{ Const x)` and «subterm u (Cons i q) u'` show ?thesis
by auto
    next
        fix t1 t2 assume v = ( Comb t1 t2)
        from this and <u =(subst v s)\rangle
            have u=(Comb (subst t1 s) (subst t2 s)) by auto
    have i=Left \veei= Right using indices.exhaust by auto
    from <i= Left \vee i= Right> and <u = (Comb (subst t1 s) (subst t2 s))>
        and <subterm u (Cons iq) u'> obtain ti where
        subterm(subst ti s) q u' and ti=t1\vee ti=t2 and subterm v [i] ti
        using <v = t1 \cdot t2> by auto
    from \?prop q> and <subterm (subst ti s) q u'> have
            (\existsx q1 q2. (is-a-variable x) ^(subterm (subst x s) q1 u')^
                                    (subterm ti q2 x) ^ (q=(append q2 q1))) \vee
                            ((\exists v
s)))))
        by auto
        then show ?thesis
    proof
        assume (\existsx q1 q2. (is-a-variable x) ^(subterm (subst x s) q1 u') ^
                            (subterm ti q2 x) ^(q=(append q2 q1)))
        then obtain x q1 q2 where is-a-variable x and subterm (subst x s) q1 u'
                and subterm ti q2 x and q=(append q2 q1)
            by auto
```

```
            from 〈subterm ti q2 x〉 and <subterm v [i] ti〉 have subterm v (i##q2) x
            using <i= indices.Left \vee i= indices.Right\rangle\langlev=t1 \cdot t2\rangle by auto
            from <q = append q2 q1> have i# q=(append (i# q2) q1) by auto
            from <i# q=(append (i# q2) q1)\rangle and <is-a-variable x〉
            and <subterm(subst x s) q1 u'〉 and <subterm v (i # q2) x〉
            show ?thesis by blast
        next
            assume ((\exists v}..((\neg is-a-variable v') ^(subterm ti q v') ^(u'=(subst v'
s)))))
            then obtain }\mp@subsup{v}{}{\prime}\mathrm{ where ( }\neg\mathrm{ is-a-variable v}\mp@subsup{v}{}{\prime})(\mathrm{ subterm ti q v})\mathrm{ ) and }\mp@subsup{u}{}{\prime}=(\mathrm{ subst
v
            from 〈subterm ti q v`` and «subterm v [i] ti\rangle have subterm v (i# q) v'
            using <i = indices.Left \vee i= indices.Right\rangle\langlev=t1 \cdot t2\rangle by auto
            from this and }\langle(\neg\mathrm{ is-a-variable v}\mp@subsup{v}{}{\prime})\rangle\langlesubterm ti q v > and <u' = (subst v'
s)>
            show ?thesis by auto
            qed
        qed
    qed
qed
lemma vars-of-replacement:
    shows \ts.x\in vars-of s\Longrightarrow replace-subterm tpvs\Longrightarrowx\in(vars-of t) \cup
(vars-of v)
proof (induction p)
    case Nil
        from «replace-subterm t Nil v s` have s=v by auto
        from this and \langlex\in vars-of s\rangle show ?case by auto
    next case (Cons i q)
    from <replace-subterm t (Cons i q) v s` obtain t1 t2 where
                t=(Comb t1 t2)
            by (metis is-a-variable.cases replace-subterm.simps(2) replace-subterm.simps(3))
    have i= Left \veei= Right using indices.exhaust by blast
    then show ?case
    proof
            assume i= Left
            from this and <t = Comb t1 t2\rangle and <replace-subterm t (Cons i q) v s`
                obtain s1 where s=Comb s1 t2 and replace-subterm t1 qv s1
                    using replace-subterm.simps(4) by auto
            from <s=Comb s1 t2` and }\langlex\in\mathrm{ vars-of s> have }x\in\mathrm{ vars-of s1 }\veex\in\mathrm{ vars-of
t2
            by simp
            then show ?case
            proof
                assume x vars-of s1
                    from this and Cons.IH [of s1 t1] and <replace-subterm t1 q v s1> have x
\epsilon(vars-of t1) \cup (vars-of v)
            by auto
```

```
            from this and <t = (Comb t1 t2)\rangle show ?case by auto
        next
            assume x vars-of t2
            from this and <t = (Comb t1 t2)> show ?case by auto
        qed
    next
        assume i= Right
        from this and <t=Comb t1 t2\rangle and <replace-subterm t(Cons i q)v s`
            obtain s2 where s=Comb t1 s2 and replace-subterm t2 q v s2
                using replace-subterm.simps by auto
    from }\langles=Comb t1 s2\rangle and \langlex\invars-of s\rangle have x\in vars-of t1 \vee x vars-o
s2
            by simp
        then show ?case
        proof
            assume x \in vars-of s2
            from this and Cons.IH [of s2 t2] and <replace-subterm t2 q v s2> have x
\epsilon(vars-of t2) \cup (vars-of v)
                    by auto
            from this and <t = (Comb t1 t2)\rangle show ?case by auto
        next
            assume x f vars-of t1
            from this and <t = (Comb t1 t2)> show ?case by auto
        qed
    qed
qed
lemma vars-of-replacement-set:
    assumes replace-subterm t p vs
    shows vars-of s}\subseteq(\mathrm{ vars-of t) U(vars-of v)
by (meson assms subsetI vars-of-replacement)
```


### 2.3 Substitutions and Most General Unifiers

Substitutions are defined in the Unification theory. We provide some additional definitions and lemmata.

```
fun subst-set :: 'a trm set \(=>\) 'a subst \(=>\) ' \(a\) trm set
    where
    \((\) subst-set \(S \sigma)=\{u . \exists t . u=(\) subst \(t \sigma) \wedge t \in S\}\)
```

definition subst-codomain
where
subst-codomain $\sigma V=\{x . \exists y .($ subst $(\operatorname{Var} y) \sigma)=(\operatorname{Var} x) \wedge(y \in V)\}$
lemma subst-codomain-is-finite:
assumes finite $A$
shows finite (subst-codomain $\eta$ A)
proof -
have $i:((\lambda x .(\operatorname{Var} x))$ ' $($ subst-codomain $\eta A)) \subseteq((\lambda x .(\operatorname{subst}(\operatorname{Var} x) \eta))$ ' $A)$

```
    proof
    fix }x\mathrm{ assume }x\in((\lambdax.(Var x))'(subst-codomain \eta A)
    from this obtain y where }y\in(\mathrm{ subst-codomain }\etaA)\mathrm{ and }x=(\mathrm{ Var y) by auto
    from }\langley\in(\mathrm{ subst-codomain }\etaA)\rangle\mathrm{ obtain z where (subst (Var z) }\eta)=(\mathrm{ Var
y) (z\inA)
            unfolding subst-codomain-def by auto
    from}\langle(z\inA)\rangle\langlex=(\operatorname{Var}y)\rangle\langle(subst (Var z) \eta)=(Var y)\rangle this show
            x\in((\lambdax. (subst (Var x) \eta))'A)using image-iff by fastforce
    qed
    have inj-on ( }\lambdax.(Var x)) (subst-codomain \eta A) by (meson inj-onI trm.inject(1)
    from assms(1) have finite ((\lambdax. (subst (Var x) \eta))' A) by auto
    from this and i have finite (( }\lambdax.(\operatorname{Var}x))'(subst-codomain \eta A)
        using rev-finite-subset by auto
    from this and <inj-on ( }\lambdax\mathrm{ . (Var x)) (subst-codomain }\eta\mathrm{ A)> show ?thesis using
finite-imageD [of ( }\lambdax.(\operatorname{Var}x))\mathrm{ subst-codomain }\eta\mathrm{ A]
    by auto
qed
```

The notions of unifiers, most general unifiers, the unification algorithm and a proof of correctness are provided in the Unification theory. Below, we prove that the algorithm is complete.

```
lemma subt-decompose:
    shows \(\forall t 1\) t2. Comb t1 t2 \(\prec s \longrightarrow(t 1 \prec s \wedge t 2 \prec s)\)
proof \(((\) induction \(s),(\) simp \(),(\) simp \())\)
    case (Comb s1 s2)
    show ? case
    proof ((rule allI)+,(rule impI))
    fix \(t 1\) t2 assume Comb t1 t2 \(\prec\) Comb s1 s2
        show \(t 1 \prec(\) Comb s1 s2 \() \wedge t 2 \prec(C o m b s 1 s 2)\)
        proof (rule ccontr)
            assume neg: \(\neg(t 1 \prec(\) Comb s1 s2 \() \wedge t 2 \prec(\) Comb s1 s2 \())\)
            from 〈Comb t1 t2 \(\prec\) Comb s1 s2〉 have
                \(d:\) Comb t1 t2 \(=s 1 \vee\) Comb t1 t2 \(=s 2 \vee\) Comb t1 t2 \(\prec\) s1 \(\vee\) Comb t1
t2 \(\prec\) s2
                by auto
        have \(i: \neg(\) Comb t1 t2 \(=s 1)\)
        proof
            assume (Comb t1 t2 = s1)
            then have \(t 1 \prec s 1\) and \(t 2 \prec s 1\) by auto
            from this and neg show False by auto
        qed
        have \(i i\) : \(\neg(\) Comb t1 t2 \(=s 2)\)
        proof
            assume (Comb t1 t2 \(=\) s2)
            then have \(t 1 \prec s 2\) and \(t 2 \prec s 2\) by auto
                from this and neg show False by auto
            qed
            have \(i i i\) : \(\neg(\) Comb t1 t2 \(\prec s 1)\)
            proof
```

```
                assume (Comb t1 t2 \prec s1)
                    then have t1 \prec s1^t2 \precs1 using Comb.IH by metis
                    from this and neg show False by auto
                qed
                have iv: \neg(Comb t1 t2 \prec s2)
                proof
                    assume (Comb t1 t2 \prec s\mathcal{Q})
                    then have t1 \prec s2 ^t2 \prec s2 using Comb.IH by metis
                    from this and neg show False by auto
                qed
                from d and i ii iii iv show False by auto
        qed
    qed
qed
lemma subt-irref:
    shows \neg (s\precs)
proof ((induction s),(simp),(simp))
    case (Comb t1 t2)
        show ?case
        proof
        assume Comb t1 t2 \prec Comb t1 t2
        then have i: Comb t1 t2 \not= t1 using Comb.IH(1) by fastforce
    from 〈Comb t1 t2 \prec Comb t1 t2` have ii: Comb t1 t2 = t2 using Comb.IH(2)
by fastforce
        from i ii and <Comb t1 t2 \precComb t1 t2` have Comb t1 t2 \prec t1 \vee Comb
t1 t2 \prec t2 by auto
        then show False
        proof
            assume Comb t1 t2 }\prec t
            then have t1 \prec t1 using subt-decompose [of t1] by metis
            from this show False using Comb.IH by auto
        next
            assume Comb t1 t2 \prec t2
            then have t2 \prec t2 using subt-decompose [of t2] by metis
            from this show False using Comb.IH by auto
        qed
    qed
qed
lemma MGU-exists:
    shows }\forall\sigma.((\mathrm{ subst t }\sigma)=(\mathrm{ subst s }\sigma)\longrightarrow\mathrm{ unify t s}\not=~\mathrm{ None )
proof (rule unify.induct)
    fix x s1 s2 show }\forall\sigma\mathrm{ :: 'a subst .((subst (Const x) }\sigma)=(\mathrm{ subst (Comb s1 sZ)
\sigma)
    unify (Const x)(Comb s1 s2) }\not==\mathrm{ None) by simp
    next
    fix t1 t2 y show \forall\sigma :: 'a subst.(subst (Comb t1 t2) \sigma) = (subst (Const y) \sigma)
            \longrightarrow \text { unify (Comb t1 t2) (Const y) } = \text { None by simp}
```

next
fix $x y$ show $\forall \sigma::$ 'a subst. (subst (Const $x) \sigma)=($ subst (Var $y) \sigma)$
$\longrightarrow$ unify (Const $x)($ Var $y) \neq$ None using unify.simps(3) by fastforce
next
fix $t 1$ t2 $y$ show $\forall \sigma$ :: 'a subst.(subst ( Comb t1 t2) $\sigma)=($ subst $(\operatorname{Var} y) \sigma)$
$\longrightarrow$ unify (Comb t1 t2) (Var y) $\neq$ None
by (metis option.distinct(1) subst-mono subt-irrefl unify.simps(4))
next
fix $x s$ show $\forall \sigma::$ 'a subst. $($ subst $(\operatorname{Var} x) \sigma)=($ subst $s \sigma)$

$$
\longrightarrow \text { unify }(\operatorname{Var} x) s \neq \text { None }
$$

by (metis option.distinct(1) subst-mono subt-irrefl unify.simps(5))
next
fix $x y$ show $\forall \sigma::{ }^{\prime} a$ subst. $($ subst $($ Const $x) \sigma)=($ subst $($ Const $y) \sigma)$
$\longrightarrow$ unify (Const $x$ ) (Const $y) \neq$ None by simp
next
fix $t 1$ t2 s1 s2
show $\forall \sigma::$ 'a subst. (subst t1 $\sigma=$ subst s1 $\sigma \longrightarrow$ unify t1 s1 $\neq$ None) $\Longrightarrow$
( $\bigwedge$ x2. unify t1 s1 $=$ Some x2 $\Longrightarrow$
$\forall \sigma$. subst $(t 2 \triangleleft x 2) \sigma=$ subst $(s 2 \triangleleft x 2) \sigma \longrightarrow$
unify $($ t2 $\triangleleft x 2)(s 2 \triangleleft x 2) \neq$ None $) \Longrightarrow$
$\forall \sigma$. (subst (t1 • t2) $\sigma=$ subst $(s 1 \cdot s 2) \sigma \longrightarrow$
unify $(t 1 \cdot t 2)(s 1 \cdot s 2) \neq$ None $)$
proof -
assume $h 1: \forall \sigma$. (subst t1 $\sigma=$ subst s1 $\sigma \longrightarrow$ unify t1 s1 $\neq$ None)
assume h2: ( $\bigwedge x 2$. unify t1 s1 $=$ Some $x 2 \Longrightarrow$
$\forall \sigma$. subst $(t 2 \triangleleft x 2) \sigma=$ subst $(s 2 \triangleleft x 2) \sigma \longrightarrow$
unify $($ t2 $\triangleleft x 2)(s 2 \triangleleft x 2) \neq$ None $)$
show $\forall \sigma$. (subst ( $t 1 \cdot t 2$ ) $\sigma=\operatorname{subst}(s 1 \cdot$ s2) $\sigma \longrightarrow$
unify $(t 1 \cdot t 2)(s 1 \cdot s \mathcal{2}) \neq$ None $)$
proof $(($ rule allI $),($ rule impI $))$
fix $\sigma$ assume $h 3$ : subst ( $t 1 \cdot t 2$ ) $\sigma=\operatorname{subst}(s 1 \cdot s 2) \sigma$
from $h 3$ have subst t1 $\sigma=$ subst s1 $\sigma$ by auto
from this and h1 have unify t1 s1 $\neq$ None by auto
from this obtain $\vartheta$ where unify t1 s1 = Some $\vartheta$ and $M G U \vartheta$ t1 s1
by (meson option.exhaust unify-computes-MGU)
from 〈subst t1 $\sigma=$ subst s1 $\sigma\rangle$ have Unifier $\sigma$ t1 s1
unfolding Unifier-def by auto
from this and $\langle M G U \vartheta t 1 s 1\rangle$ obtain $\eta$ where $\sigma \doteq \vartheta \diamond \eta$ using $M G U$-def
by metis
from $h 3$ have subst t2 $\sigma=$ subst s2 $\sigma$ by auto
from this and $\langle\sigma \doteq \vartheta \diamond \eta\rangle$
have subst (subst t2 $\vartheta$ ) $\eta$
= subst (subst s2 $\vartheta$ ) $\eta$
by (simp add: subst-eq-def)
from this and «unify t1 s1 = Some $\vartheta\rangle$ and $h 2\left[\begin{array}{ll}\text { of } & \text { ] have unify }(t 2 ~ \\ \text { h }\end{array}\right)$
$(s 2 \triangleleft \vartheta) \neq$ None
by auto
from this show unify $(t 1 \cdot t 2)(s 1 \cdot s 2) \neq$ None
by (simp add: 〈unify t1 s1 = Some $\vartheta\rangle$ option.case-eq-if)

```
        qed
    qed
qed
```

We establish some useful properties of substitutions and instances.
definition ground-on :: 'a set $\Rightarrow$ 'a subst $\Rightarrow$ bool
where ground-on $V \sigma=(\forall x \in V$. (ground-term (subst (Var $x) \sigma))$ )
lemma comp-subst-terms:
assumes $\sigma \doteq \vartheta \diamond \eta$
shows (subst $t \sigma)=($ subst $($ subst $t \geqslant) \eta)$
proof -
from $\langle\sigma \doteq \vartheta \diamond \eta\rangle$ have $(($ subst $t \sigma)=($ subst $t(\vartheta \diamond \eta)))$ by auto
have (subst $t(\vartheta \diamond \eta)=($ subst $($ subst $t \vartheta) \eta))$ by auto
from this and $\ulcorner(($ subst $t \sigma)=($ subst $t(\vartheta \diamond \eta)))\rangle$ show ?thesis by auto
qed
lemma ground-instance:
assumes ground-on (vars-of $t$ ) $\sigma$
shows ground-term (subst $t \sigma$ )
proof (rule ccontr)
assume $\neg$ ground-term (subst $t \sigma$ )
then have vars-of (subst $t \sigma) \neq\{ \}$ unfolding ground-term-def by auto
then obtain $x$ where $x \in$ vars-of (subst $t \sigma$ ) by auto
then have $x \in \bigcup\{V . \exists x .(x \in($ vars-of $t)) \wedge(V=\operatorname{vars}$-of (subst $($ Var $x) \sigma))$
\}
using vars-of-instances by force
then obtain $y$ where $x \in($ vars-of (subst (Var y) $\sigma)$ ) and $y \in($ vars-of $t$ ) by blast
from $\operatorname{assms}(1)$ and $« y \in($ vars-of $t$ ) $>$ have ground-term (subst (Var $y$ ) $\sigma$ ) un-
folding ground-on-def
by auto
from this and $\langle x \in($ vars-of (subst (Var y) $\sigma$ )) $\rangle$ show False unfolding ground-term-def
by auto
qed
lemma substs-preserve-groundness:
assumes ground-term $t$
shows ground-term (subst $t \sigma$ )
by (metis assms equals0D ground-instance ground-on-def ground-term-def)
lemma ground-subst-exists :
finite $V \Longrightarrow \exists \sigma$. (ground-on $V \sigma$ )
proof (induction rule: finite.induct)
case emptyI
show ?case unfolding ground-on-def by simp
next
fix $A::$ 'a set and $a:: ' a$
assume finite $A$

```
assume hyp-ind: \exists\sigma. ground-on A \sigma
then obtain }\sigma\mathrm{ where ground-on A }\sigma\mathrm{ by auto
then show }\exists\sigma\mathrm{ . ground-on (insert a A) }
proof cases
    assume a\inA
    from this and hyp-ind show \exists\sigma. ground-on (insert a A) \sigma
        unfolding ground-on-def by auto
next
    assume a\not\inA
    obtain c where c=(Const a) and is-a-constant c by auto
    obtain \vartheta where \vartheta}=(a,c)#\sigma\mathrm{ by auto
    have }\forallx.(x\in\mathrm{ insert a A }\longrightarrow(\mathrm{ ground-term (subst (Var x) }\vartheta))
    proof ((rule allI)+,(rule impI)+)
        fix }x\mathrm{ assume }x\in\mathrm{ insert a A
        show ground-term (subst (Var x) \vartheta)
        proof cases
            assume }x=
            from this and }\langle\vartheta=(a,c)#\sigma\rangle\mathrm{ have (subst (Var x) })=c\mathrm{ by auto
            from <is-a-constant c\rangle have ground-term c using constants-are-ground by
auto
            from this and «(subst (Var x) \vartheta) = c> show ground-term (subst (Var x)
\vartheta) by auto
    next
            assume }x\not=
            from }\langlex\not=a\rangle\mathrm{ and }\langlex\in\mathrm{ insert }aA\rangle\mathrm{ have }x\inA\mathrm{ by auto
            from }\langlex\not=a\rangle\mathrm{ and }\langle\vartheta=(a,c)#\sigma\rangle\mathrm{ have (subst (Var x) ७) = (subst (Var
x) \sigma) by auto
            from this and }\langlex\inA\rangle\mathrm{ and <ground-on A }\sigma
                    show ground-term (subst (Var x) \vartheta) unfolding ground-on-def by auto
        qed
    qed
    from this show ?thesis unfolding ground-on-def by auto
    qed
qed
lemma substs-preserve-ground-terms :
    assumes ground-term t
    shows subst t \sigma=t
by (metis agreement assms equals0D ground-term-def subst-Nil)
lemma substs-preserve-subterms :
    shows }\ts.\mathrm{ subterm t ps" subterm(subst t }\sigma\mathrm{ ) p(subst s }\sigma\mathrm{ )
proof (induction p)
    case Nil
        then have t=s using subterm.elims(2) by auto
        from }\langlet=s\rangle\mathrm{ have (subst t }\sigma\mathrm{ ) =(subst s }\sigma\mathrm{ ) by auto
        from this show ?case using Nil.prems by auto
    next case (Cons i q)
    from <subterm t (i# q) s` obtain t1 t2 where
```

```
    t=(Comb t1 t2) using subterm.elims(2) by blast
    have i= Left \veei= Right using indices.exhaust by blast
    then show subterm (subst t \sigma) (i# q) (subst s \sigma)
    proof
    assume i= Left
    from this and 〈t=Comb t1 t2\rangle and <subterm t(i# q) s\rangle
        have subterm t1 qs by auto
    from this have subterm (subst t1 \sigma) q (subst s \sigma) using Cons.IH by auto
    from this and <t=Comb t1 t2>
        show subterm (subst t \sigma) (i# q) (subst s \sigma)
                by (simp add:<i= indices.Left>)
    next assume i= Right
    from this and <t=Comb t1 t2> and <subterm t (i# q) s\rangle
            have subterm t2 q s by auto
    from this have subterm (subst t2 \sigma) q (subst s \sigma) using Cons.IH by auto
    from this and <t= Comb t1 t2>
        show subterm (subst t \sigma) (i# q) (subst s \sigma)
            by (simp add: <i = indices.Right>)
    qed
qed
lemma substs-preserve-occurs-in:
    assumes occurs-in st
    shows occurs-in (subst s \sigma) (subst t \sigma)
using substs-preserve-subterms
    using assms occurs-in-def by blast
definition coincide-on
where coincide-on \sigma \eta V = (\forallx\inV.(subst (Var x) \sigma)=((subst (Var x) \eta)))
lemma coincide-sym:
    assumes coincide-on \sigma \eta V
    shows coincide-on \eta \sigma V
by (metis assms coincide-on-def)
lemma coincide-on-term:
    shows }\bigwedge\sigma\eta\mathrm{ . coincide-on }\sigma\eta(\mathrm{ vars-of t) > subst t }\sigma=\mathrm{ subst t }
proof (induction t)
    case (Var x)
        from this show subst (Var x) \sigma= subst (Var x) \eta
            unfolding coincide-on-def by auto
    next case (Const x)
        show subst (Const x) \sigma = subst (Const x) }\eta\mathrm{ by auto
    next case (Comb t1 t2)
        from this show ?case unfolding coincide-on-def by auto
qed
lemma ground-replacement:
    assumes replace-subterm t pvs
```

```
    assumes ground-term (subst t \sigma)
    assumes ground-term (subst v \sigma)
    shows ground-term (subst s \sigma)
proof -
    from assms(1) have vars-of s\subseteq(vars-of t)\cup(vars-of v) using vars-of-replacement-set
[of t p v s]
    by auto
    from assms(2) have ground-on (vars-of t) \sigma unfolding ground-on-def
        by (metis contra-subsetD ex-in-conv ground-term-def
            occs-vars-subset subst-mono vars-iff-occseq)
    from assms(3) have ground-on (vars-of v) \sigma unfolding ground-on-def
        by (metis contra-subsetD ex-in-conv ground-term-def
            occs-vars-subset subst-mono vars-iff-occseq)
    from <vars-of s}\subseteq(vars-of t)\cup(vars-of v)\rangle\langleground-on (vars-of t) \sigma>
        and «ground-on (vars-of v) \sigma〉 have ground-on (vars-of s) \sigma
        by (meson UnE ground-on-def rev-subsetD)
    from this show ?thesis using ground-instance by blast
qed
We now show that two disjoint substitutions can always be fused．
lemma combine－substs：
assumes finite V1
assumes \(V 1 \cap V 2=\{ \}\)
assumes ground－on V1 \(\eta 1\)
shows \(\exists \sigma\) ．（coincide－on \(\sigma \eta 1\) V1 \() \wedge(\) coincide－on \(\sigma \eta 2\) V2 \()\)
proof－
have finite \(V 1 \Longrightarrow\) ground－on V1 \(\eta 1 \Longrightarrow V 1 \cap V 2=\{ \} \Longrightarrow \exists \sigma\) ．（coincide－on
\(\sigma \eta 1\) V1）\(\wedge(\) coincide－on \(\sigma\) \(\eta 2\) V2 \()\)
proof（induction rule：finite．induct）
case emptyI
show ？case unfolding coincide－on－def by auto
next fix \(V 1\) ：：＇a set and \(a::^{\prime} a\)
assume finite V1
assume hyp－ind：ground－on V1 \(\eta 1 \Longrightarrow V 1 \cap V 2=\{ \}\)
\(\Longrightarrow \exists \sigma\) ．（coincide－on \(\sigma \eta 1\) V1）\(\wedge(\) coincide－on \(\sigma \eta 2\) V2 \()\)
assume ground－on（insert a V1）\(\eta 1\)
assume（insert a V1）\(\cap V 2=\{ \}\)
from this have \(V 1 \cap V 2=\{ \}\) by auto
from 〈ground－on（insert a V1）\(\eta 1\) 〉 have ground－on V1 \(\eta 1\)
unfolding ground－on－def by auto
from this and hyp－ind and 〈V1 \(\cap V 2=\{ \}\rangle\) obtain \(\sigma^{\prime}\)
where \(c:\left(\right.\) coincide－on \(\sigma^{\prime} \eta 1\) V1 \() \wedge\left(\right.\) coincide－on \(\sigma^{\prime} \eta 2\) V2 \()\) by auto
let \(? t=\) subst \((\) Var a）\(\eta 1\)
from \(\operatorname{assms}(2)\) have ground－term ？t
by（meson 〈ground－on（insert a V1）\(\eta 1\) 〉 ground－on－def insertI1）
let ？\(\sigma=\operatorname{comp}[(a, ? t)] \sigma^{\prime}\)
have coincide－on ？\(\sigma \eta 1\)（insert a V1）
proof（rule ccontr）
assume \(\neg\) coincide－on ？\(\sigma \eta 1\)（insert a V1）
```

```
    then obtain \(x\) where \(x \in(\) insert \(a\) V1) and
        \((\) subst \((\operatorname{Var} x) ? \sigma) \neq((\) subst \((\operatorname{Var} x) \eta 1))\)
        unfolding coincide-on-def by blast
    have subst (Var a) ? \(\sigma=\) subst ?t \(\sigma^{\prime}\) by simp
    from 〈ground-term ? \(t\rangle\) have subst (Var a) ? \(\sigma=? t\)
        using substs-preserve-ground-terms by auto
    from this and «(subst \((\operatorname{Var} x) ? \sigma) \neq((\) subst \((\operatorname{Var} x) \eta 1))\rangle\)
        have \(x \neq a\) by blast
    from this and \(\langle x \in(\) insert \(a V 1)\rangle\) have \(x \in V 1\) by auto
    from \(\langle x \neq a\rangle\) have \(\left(\right.\) subst \((\operatorname{Var} x)\) ? \(\sigma\) ) \(=\left(\right.\) subst \(\left.(\operatorname{Var} x) \sigma^{\prime}\right)\) by auto
    from \(c\) and \(\langle x \in V 1\rangle\) have \(\left(\right.\) subst \(\left.(\operatorname{Var} x) \sigma^{\prime}\right)=(\) subst \((\operatorname{Var} x) \eta 1)\)
        unfolding coincide-on-def by blast
    from this and \(\left\langle(\right.\) subst \((\operatorname{Var} x) ? \sigma)=\left(\right.\) subst \(\left.\left.(\operatorname{Var} x) \sigma^{\prime}\right)\right\rangle\)
    and \(\langle(\operatorname{subst}(\operatorname{Var} x) ? \sigma) \neq((\) subst \((\operatorname{Var} x) \eta 1))\rangle\) show False by auto
    qed
    have coincide-on ? \(\sigma\) 72 V2
    proof (rule ccontr)
    assume \(\neg\) coincide-on ? \(\quad \eta 2 \mathrm{~V} 2\)
    then obtain \(x\) where \(x \in V 2\) and
        \((\) subst \((\operatorname{Var} x) ? \sigma) \neq((\operatorname{subst}(\operatorname{Var} x) \eta 2))\)
        unfolding coincide-on-def by blast
    from \(\langle(\) insert a V1 \() \cap V 2=\{ \}\rangle\) and \(\langle x \in V 2\rangle\) have \(x \neq a\) by auto
    from this have \((\operatorname{subst}(\operatorname{Var} x) ? \sigma)=\left(\right.\) subst \(\left.(\operatorname{Var} x) \sigma^{\prime}\right)\) by auto
    from \(c\) and \(\langle x \in V 2\rangle\) have (subst \(\left.(\operatorname{Var} x) \sigma^{\prime}\right)=(\) subst \((\operatorname{Var} x) \eta 2)\)
        unfolding coincide-on-def by blast
    from this and \(\left\langle\left(\right.\right.\) subst \((\operatorname{Var} x)\) ? \(\sigma\) ) \(=\left(\right.\) subst \(\left.\left.(\operatorname{Var} x) \sigma^{\prime}\right)\right\rangle\)
        and \(\langle(\) subst \((\operatorname{Var} x) ? \sigma) \neq((\) subst \((\operatorname{Var} x) \eta 2))\rangle\) show False by auto
    qed
    from 〈coincide-on ? \(\sigma \eta 1\) (insert a V1)〉〈coincide-on ? \(\sigma\) \(\eta 2\) V2〉
        show \(\exists \sigma\). (coincide-on \(\sigma \eta 1(\) insert \(a \operatorname{V1})) \wedge(\) coincide-on \(\sigma \eta 2\) V2) by
auto
    qed
    from this and assms show ?thesis by auto
qed
```

We define a map function for substitutions and prove its correctness．

```
fun map-subst
    where map-subst \(f\) Nil \(=\) Nil
        \(\mid\) map-subst \(f((x, y) \# l)=(x,(f y)) \#(\) map-subst \(f l)\)
```

lemma map-subst-lemma:
shows $(($ subst $(\operatorname{Var} x) \sigma) \neq(\operatorname{Var} x) \vee($ subst $(\operatorname{Var} x) \sigma) \neq($ subst $(\operatorname{Var} x)$
(map-subst $f \sigma)$ )
$\longrightarrow(($ subst $(\operatorname{Var} x)($ map-subst $f \sigma))=(f($ subst $(\operatorname{Var} x) \sigma)))$
proof (induction $\sigma$, simp)
next case (Cons po)
let ? $u=(f s t p)$
let $? v=($ snd $p)$
show ?case

```
    proof
    assume ((subst (Var x) (Cons p \sigma))}\not=(\mathrm{ Var x)
        \vee ~ ( s u b s t ~ ( V a r ~ x ) ~ ( C o n s ~ p ~ \sigma ) ) ~
            # (subst (Var x) (map-subst f (Cons p \sigma))))
    have map-subst f(Cons p \sigma)=((?u,(f ?v)) # (map-subst f \sigma))
            by (metis map-subst.simps(2) prod.collapse)
    show (subst (Var x)(map-subst f(Cons p \sigma))) =(f (subst (Var x) (Cons p
\sigma)))
    proof cases
        assume x = ?u
        from this have subst (Var x) (Cons p \sigma) =?v
            by (metis assoc.simps(2) prod.collapse subst.simps(1))
        from <map-subst f(Cons p\sigma)=((?u,(f ?v)) # (map-subst f \sigma))}\mathrm{ \
            and <x = ?u>
            have subst (Var x) (map-subst f(Cons p \sigma)) = (f ?v) by simp
    from <subst (Var x) (Cons p \sigma)= ?v`<subst (Var x) (map-subst f (Cons p
\sigma))=(f ?v)> show ?thesis by auto
    next
        assume }x\not=?
        from this have subst (Var x) (Cons p \sigma) =( subst (Var x) \sigma)
        by (metis assoc.simps(2) prod.collapse subst.simps(1))
    from <map-subst f(Cons p\sigma)=((?u,(f?v)) # (map-subst f \sigma))`
        and <x\not=? ?u`
        have subst (Var x) (map-subst f (Cons p \sigma)) =
            subst (Var x) (map-subst f \sigma) by simp
        from this and Cons.IH have
            subst (Var x) (map-subst f(Cons p \sigma)) =(f(subst (Var x) \sigma))
            using <subst (Var x) (p# \sigma) = subst (Var x) \sigma`<subst (Var x) (p# | )
# Var x \vee subst (Var x) (p#\sigma) = subst (Var x) (map-subst f (p#\sigma))> by auto
    from this and <subst (Var x) (Cons p \sigma)=(subst (Var x) \sigma)〉 show ?thesis
by auto
    qed
    qed
qed
```


### 2.3.1 Minimum Idempotent Most General Unifier

definition min-IMGU :: 'a subst $\Rightarrow$ 'a trm $\Rightarrow$ ' $a$ trm $\Rightarrow$ bool where
$\min -I M G U \mu t u \longleftrightarrow$
$I M G U \mu t u \wedge f s t$ ' set $\mu \subseteq$ vars-of $t \cup$ vars-of $u \wedge$ range-vars $\mu \subseteq$ vars-of $t \cup$ vars-of $u$
lemma unify-computes-min-IMGU:
unify $M N=$ Some $\sigma \Longrightarrow \min -I M G U \sigma M N$
by (simp add: min-IMGU-def IMGU-iff-Idem-and-MGU unify-computes-MGU unify-gives-Idem
unify-gives-minimal-domain unify-gives-minimal-range)

### 2.4 Congruences

We now define the notion of a congruence on ground terms, i.e., an equivalence relation that is closed under contextual embedding.
type-synonym 'a binary-relation-on-trms $=$ 'a trm $\Rightarrow$ 'a trm $\Rightarrow$ bool
definition reflexive :: 'a binary-relation-on-trms $\Rightarrow$ bool where
$($ reflexive $x)=(\forall y .(x y y))$
definition symmetric :: 'a binary-relation-on-trms $\Rightarrow$ bool where
$($ symmetric $x)=(\forall y . \forall z .((x y z)=(x z y)))$
definition transitive :: 'a binary-relation-on-trms $\Rightarrow$ bool where
$($ transitive $x)=(\forall y . \forall z . \forall u .(x y z) \longrightarrow(x z u) \longrightarrow(x y u))$
definition equivalence-relation :: 'a binary-relation-on-trms $\Rightarrow$ bool where

```
    (equivalence-relation x)=((reflexive }x)\wedge(\mathrm{ symmetric }x)\wedge(\mathrm{ transitive }x)
```

definition compatible-with-structure :: ('a binary-relation-on-trms) $\Rightarrow$ bool where
(compatible-with-structure $x)=(\forall$ t1 t2 s1 s2.

$$
(x \text { t1 s1 }) \longrightarrow(x \text { t2 s2 }) \longrightarrow(x(\text { Comb t1 t2 })(\text { Comb s1 s2 })))
$$

definition congruence :: 'a binary-relation-on-trms $\Rightarrow$ bool where
$($ congruence $x)=(($ equivalence-relation $x) \wedge($ compatible-with-structure $x))$
lemma replacement-preserves-congruences :
shows $\bigwedge t$ s. $($ congruence $I) \Longrightarrow(I$ (subst $u \sigma)($ subst $v \sigma))$
$\Longrightarrow$ subterm t $p u \Longrightarrow$ replace-subterm $t p$ vs
$\Longrightarrow(I$ (subst $t \sigma) \quad($ subst $s \sigma))$
proof (induction p)
case Nil
from $\langle$ subterm $t$ Nil $u\rangle$ have $t=u$ by auto
from 〈replace-subterm $t$ Nil $v s\rangle$ have $s=v$ by auto
from $\langle t=u\rangle$ and $\langle s=v\rangle$ and $\langle(I($ subst $u \sigma) \quad($ subst $v \sigma))\rangle$
show ?case by auto
next case (Cons i q)
from $\langle$ subterm $t(i \# q) u\rangle$ obtain $t 1$ t2 where
$t=($ Comb t1 t2) using subterm.elims(2) by blast
have $i=$ Left $\vee i=$ Right using indices.exhaust by blast
then show $I$ (subst $t \sigma$ ) (subst $s \sigma$ )
proof
assume $i=$ Left
from this and $\langle t=$ Comb t1 t2〉 and $\langle\operatorname{subterm} t(i \# q) u\rangle$
have subterm t1 qu by auto
from $\langle i=$ Left $\rangle$ and $\langle t=$ Comb $t 1$ t2 $\rangle$ and $\langle$ replace－subterm $t(i \# q) v s\rangle$ obtain $t 1^{\prime}$ where replace－subterm $t 1 q v t 1$＇and $s=C o m b t 1 ' t 2$ by auto
from 〈congruence $I\rangle$ and $\langle(I$（subst $u \sigma)($ subst $v \sigma))\rangle$
and «subterm t1 qu〉 and 〈replace－subterm t1 q v t1＇〉 have
$I$（subst t1 $\sigma$ ）（subst t1＇$\sigma$ ）using Cons．IH Cons．prems（1）by blast
from 〈congruence $I$ 〉 have $I$（subst t2 $\sigma$ ）（subst t2 $\sigma$ ）
unfolding congruence－def equivalence－relation－def reflexive－def by auto
from $\langle I$（subst t1 $\sigma$ ）（subst t1＇$\sigma$ ）〉
and $\langle I$（subst t2 $\sigma$ ）（subst t2 $\sigma$ ）＞
and $\langle$ congruence $I\rangle$ and $\langle t=(\operatorname{Comb} t 1$ t2 $)\rangle$ and $\left\langle s=\left(\operatorname{Comb} t 1^{\prime} t 2\right)\right\rangle$
show $I$（subst $t \sigma$ ）（subst s $\sigma$ ）
unfolding congruence－def compatible－with－structure－def by auto
next
assume $i=$ Right
from this and $\left\langle t=\right.$ Comb t1 t2 ${ }^{2}$ and $\langle\operatorname{subterm} t(i \# q) u\rangle$
have subterm t2 $q u$ by auto
from $\langle i=$ Right $\rangle$ and $\langle t=$ Comb $t 1$ t2 $\rangle$ and $\langle$ replace－subterm $t(i \# q) v s\rangle$
obtain t2 $^{\prime}$ where replace－subterm t2 $q v$ t2 $^{\prime}$ and $s=\operatorname{Comb}$ t1 t2＇by auto
from 〈congruence $I\rangle$ and $\langle(I$（subst $u \sigma)($ subst $v \sigma))\rangle$
and «subterm t2 $q u$ end 〈replace－subterm t2 $\left.q v t^{2}\right\rangle$ have
I（subst t2 $\sigma$ ）（subst t2＇$\sigma$ ）using Cons．IH Cons．prems（1）by blast
from 〈congruence $I$ 〉 have $I$（subst t1 $\sigma$ ）（subst t1 $\sigma$ ）
unfolding congruence－def equivalence－relation－def reflexive－def by auto
from 〈I（subst t2 $\sigma$ ）（subst t2＇$\sigma$ ）〉
and $\langle I$（subst t1 $\sigma$ ）（subst t1 $\sigma$ ）〉
and $\langle$ congruence $I\rangle$ and $\left\langle t=(\operatorname{Comb} t 1\right.$ t2）$\rangle$ and $\left\langle s=\left(\right.\right.$ Comb t1 t2 $\left.\left.{ }^{\prime}\right)\right\rangle$
show $I$（subst $t \sigma$ ）（subst s $\sigma$ ）
unfolding congruence－def compatible－with－structure－def by auto
qed
qed
definition equivalent－on
where equivalent－on $\sigma \eta V I=(\forall x \in V$ ．
$(I(\operatorname{subst}(\operatorname{Var} x) \sigma)((\operatorname{subst}(\operatorname{Var} x) \eta))))$
lemma equivalent－on－term：
assumes congruence I
shows $\wedge \sigma \eta$ ．equivalent－on $\sigma \eta($ vars－of $t) I \Longrightarrow(I$（subst $t \sigma)($ subst $t \eta))$
proof（induction $t$ ）
case（Var $x$ ）
from this show $(I(\operatorname{subst}(\operatorname{Var} x) \sigma)(\operatorname{subst}(\operatorname{Var} x) \eta))$
unfolding equivalent－on－def by auto
next case（Const $x$ ）
from $\operatorname{assms}(1)$ show $(I$（subst（Const $x) \sigma)($ subst（Const $x) \eta))$
unfolding congruence－def equivalence－relation－def reflexive－def by auto
next case（Comb t1 t2）
from this assms（1）show ？case unfolding equivalent－on－def
unfolding congruence-def compatible-with-structure-def by auto qed

### 2.5 Renamings

We define the usual notion of a renaming. We show that fresh renamings always exist (provided the set of variables is infinite) and that renamings admit inverses.
definition renaming
where
renaming $\sigma V=(\forall x \in V .(i s-a$-variable $($ subst $(\operatorname{Var} x) \sigma))$
$\wedge(\forall x y .((x \in V) \longrightarrow(y \in V) \longrightarrow x \neq y \longrightarrow($ subst $($ Var $x) \sigma) \neq($ subst $(\operatorname{Var} y) \sigma)))$ )
lemma renamings-admit-inverse:
shows finite $V \Longrightarrow$ renaming $\sigma V \Longrightarrow \exists \vartheta .(\forall x \in V$. (subst (subst (Var $x$ ) $\sigma$ ) $\vartheta)=(\operatorname{Var} x))$
$\wedge(\forall x .(x \notin($ subst-codomain $\sigma V) \longrightarrow(\operatorname{subst}(\operatorname{Var} x) \vartheta)=(\operatorname{Var} x)))$
$\wedge(\forall x$. is-a-variable (subst $(\operatorname{Var} x) \vartheta))$
proof (induction rule: finite.induct)
case emptyI
let ? $\vartheta=[]$
have $i:(\forall x \in\{ \}$. (subst (subst $(\operatorname{Var} x) \sigma)$ ? $\vartheta)=(\operatorname{Var} x))$ by auto
have $i i:(\forall x .(x \notin($ subst-codomain $\sigma\{ \}) \longrightarrow(\operatorname{subst}(\operatorname{Var} x)$ ? $\vartheta)=(\operatorname{Var} x)))$ by auto
have iii: $\forall x$. is-a-variable (subst ( $\operatorname{Var} x$ ) ? $V$ ) by simp
from $i$ ii iii show ?case by metis
next
fix $A:: ' a$ set and $a::^{\prime} a$
assume finite $A$
assume hyp-ind: renaming $\sigma A \Longrightarrow \exists \vartheta .(\forall x \in A .(\operatorname{subst}(\operatorname{subst}(\operatorname{Var} x) \sigma) \vartheta)$ $=(\operatorname{Var} x))$
$\wedge(\forall x .(x \notin($ subst-codomain $\sigma A) \longrightarrow(\operatorname{subst}(\operatorname{Var} x) \vartheta)=(\operatorname{Var} x)))$
$\wedge(\forall x$. is-a-variable (subst $(\operatorname{Var} x) \vartheta))$
show renaming $\sigma$ (insert $a A) \Longrightarrow \exists \vartheta .(\forall x \in$ (insert a $A$ ). (subst (subst (Var x) $\sigma) \vartheta=(\operatorname{Var} x))$
$\wedge(\forall x .(x \notin($ subst-codomain $\sigma($ insert $a A)) \longrightarrow(\operatorname{subst}(\operatorname{Var} x) \vartheta)=(\operatorname{Var} x)))$
$\wedge(\forall x$. is-a-variable $($ subst $(\operatorname{Var} x) \vartheta))$
proof -
assume renaming $\sigma$ (insert a $A$ )
show $\exists \vartheta .(\forall x \in($ insert $a A)$. (subst $(\operatorname{subst}(\operatorname{Var} x) \sigma) \vartheta)=(\operatorname{Var} x))$
$\wedge(\forall x .(x \notin($ subst-codomain $\sigma($ insert $a A)) \longrightarrow(\operatorname{subst}(\operatorname{Var} x) \vartheta)=(\operatorname{Var} x)))$
$\wedge(\forall x$.is-a-variable (subst $(\operatorname{Var} x) \vartheta))$
proof (cases)
assume $a \in A$
from this have insert a $A=A$ by auto
from this and 〈renaming $\sigma$ (insert a $A$ )〉 hyp-ind show ?thesis by metis
next assume $a \notin A$
from $\langle$ renaming $\sigma($ insert $a A)\rangle$ have renaming $\sigma A$ unfolding renaming-def by blast
from this and hyp-ind obtain $\vartheta$ where $i:(\forall x \in A$. (subst (subst (Var $x$ ) $\sigma) \vartheta)=($ Var $x))$ and
ii: $(\forall x .(x \notin($ subst-codomain $\sigma A) \longrightarrow(\operatorname{subst}(\operatorname{Var} x) \vartheta)=(\operatorname{Var} x)))$ and iii: $\forall x$. is-a-variable ( $\operatorname{Var} x \triangleleft \vartheta$ ) by metis
from 〈renaming $\sigma$ (insert a $A$ ) 〉 have is-a-variable (subst (Var a) $\sigma$ ) unfolding renaming-def by blast
from this obtain $b$ where (subst (Var a) $\sigma$ ) $=(\operatorname{Var} b)$ using is-a-variable.elims(2) by auto

$$
\text { let } ? \eta=(b,(\text { Var } a)) \# \vartheta
$$

have $i^{\prime}:(\forall x \in($ insert $a \operatorname{A})$. $(\operatorname{subst}(\operatorname{subst}(\operatorname{Var} x) \sigma) ? \eta)=(\operatorname{Var} x))$

## proof

fix $x$ assume $x \in($ insert $a A)$
show (subst (subst $(\operatorname{Var} x) \sigma) ? \eta)=(\operatorname{Var} x)$
proof (cases)
assume $x=a$
from this
have $($ subst $($ Var $b)((b,($ Var $a)) \# N i l))=($ Var $a)$
by $\operatorname{simp}$
have $b \notin($ subst-codomain $\sigma A)$
proof
assume $b \in($ subst-codomain $\sigma A)$
from this have $\exists y .($ subst $(\operatorname{Var} y) \sigma)=(\operatorname{Var} b) \wedge(y \in A)$ unfolding subst-codomain-def
by force
then obtain $a^{\prime}$ where $a^{\prime} \in A$ and subst $\left(\operatorname{Var} a^{\prime}\right) \sigma=(\operatorname{Var} b)$
by metis
from $\left\langle a^{\prime} \in A\right\rangle$ and $\langle a \notin A\rangle$ have $a \neq a^{\prime}$ by auto
have $a \in($ insert $a A)$ by auto
from $\left\langle a \neq a^{\prime}\right\rangle$ and $\left\langle a^{\prime} \in A\right\rangle$ and $\langle a \in($ insert $a A)\rangle$ and $\langle$ renaming $\sigma$ (insert a A)>
have (subst (Var a) $\sigma \neq\left(\right.$ subst $\left(\right.$ Var $\left.\left.\left.a^{\prime}\right) \sigma\right)\right)$
unfolding renaming-def by blast
from this and $\left\langle\right.$ subst $\left(\right.$ Var $\left.a^{\prime}\right) \sigma=($ Var $\left.b)\right\rangle\langle($ subst $($ Var a) $\sigma)=($ Var
b) >
show False by auto
qed
from this and $i i$ have (subst $(\operatorname{Var} b) \vartheta)=(\operatorname{Var} b)$ by auto
from this and $\langle x=a\rangle\langle($ subst $(\operatorname{Var} a) \sigma)=($ Var $b)\rangle$
$\langle($ subst $(\operatorname{Var} b)((b,($ Var $a)) \# N i l))=($ Var $a)\rangle$
show $($ subst $($ subst $(\operatorname{Var} x) \sigma) ? \eta)=(\operatorname{Var} x)$
by $\operatorname{simp}$
next assume $x \neq a$
from this and $\langle x \in$ insert $a A\rangle$ obtain $x \in A$ by auto
from this $i$ have $($ subst $($ subst $(\operatorname{Var} x) \sigma) \vartheta)=(\operatorname{Var} x)$
by auto
then show (subst (subst $(\operatorname{Var} x) \sigma) ? \eta)=(\operatorname{Var} x)$
by (metis $\langle$ subst (Var a) $\sigma=$ Var $b\rangle\langle$ renaming $\sigma($ insert $a \operatorname{A})\rangle$

```
                <x\in insert a A\rangle\langlex\not=a\rangle insertI1 is-a-variable.elims(2)
                    occs.simps(1) renaming-def repl-invariance vars-iff-occseq)
        qed
    qed
    have ii': (\forallx. (x\not\in(subst-codomain \sigma (insert a A)) \longrightarrow(subst (Var x) ? \eta)
=(Var x)))
        proof ((rule allI),(rule impI))
            fix }x\mathrm{ assume }x\not\in\mathrm{ subst-codomain }\sigma\mathrm{ (insert a A)
    from this «(subst (Var a) \sigma)=(Var b)〉 have x\not=b unfolding subst-codomain-def
            by auto
            from this have (subst (Var x) ? }\eta)=(\mathrm{ subst (Var x) ७) by auto
            from <x\not\in subst-codomain \sigma (insert a A)> have x #(subst-codomain \sigma A)
unfolding subst-codomain-def
                by auto
            from this and ii have (subst (Var x) \vartheta) = (Var x) by auto
            from〈(subst (Var x) ? \eta) = (subst (Var x) \vartheta)\rangle
                and}<(\mathrm{ subst (Var x) ७) =(Var x)> show (subst (Var x) ? }\eta)=(\operatorname{Var}x
            by auto
        qed
        have iii':}\forallx. is-a-variable (subst (Var x) ? \eta
            using iii by auto
        from }\mp@subsup{i}{}{\prime
    qed
    qed
qed
lemma renaming-exists:
    assumes \neg finite (Vars :: ('a set))
    shows finite V\Longrightarrow(\forall\mp@subsup{V}{}{\prime}::'a set. finite V'\longrightarrow
\etaV)\cap V')={})))
proof (induction rule: finite.induct)
    case emptyI
    let ? }\eta=[
    show ?case unfolding renaming-def subst-codomain-def by auto
next
    fix }A\mathrm{ :: ' a set and }a::'
    assume finite }
    assume hyp-ind: }\forall\mp@subsup{V}{}{\prime}:: 'a set. finite V' \longrightarrow (\exists\eta. renaming \eta A ^ subst-codomain
\etaA\cap\mp@subsup{V}{}{\prime}={})
    show \forall\mp@subsup{V}{}{\prime}:: 'a set. finite }\mp@subsup{V}{}{\prime}\longrightarrow(\exists\eta\mathrm{ . renaming }\eta\mathrm{ (insert a A) ^ subst-codomain
\eta(insert a A) \cap V'={})
    proof ((rule allI),(rule impI))
        fix V':: 'a set assume finite V'
        from this have finite (insert a V') by auto
        from this and hyp-ind obtain }\eta\mathrm{ where renaming }\etaA\mathrm{ and (subst-codomain
\etaA)\cap(\mathrm{ insert a }\mp@subsup{V}{}{\prime})={} by metis
        from〈finite A〉 have finite (subst-codomain \eta A)
            using subst-codomain-is-finite by auto
```

from this $\left\langle\right.$ finite $\left.V^{\prime}\right\rangle$ have finite $\left(V^{\prime} \cup(\right.$ subst－codomain $\left.\eta A)\right)$ by auto
from this have finite $\left(\left(\right.\right.$ insert a $\left.V^{\prime}\right) \cup($ subst－codomain $\left.\eta A)\right)$ by auto
from this $\neg \neg$ finite Vars〉 have $\neg\left(\right.$ Vars $\subseteq\left(\left(\right.\right.$ insert a $\left.V^{\prime}\right) \cup$（subst－codomain $\eta$ A）））using rev－finite－subset
by metis
from this obtain $n v$ where $n v \in \operatorname{Vars}$ and $n v \notin\left(\right.$ insert $\left.a V^{\prime}\right)$ and $n v \notin$ （subst－codomain $\eta A$ ）by auto
let $? \eta=(a,($ Var $n v)) \# \eta$
have $i:(\forall x \in($ insert a $A)$ ．$($ is－a－variable $(\operatorname{subst}(\operatorname{Var} x)$ ？$\eta)))$
proof（rule ccontr）
assume $\neg(\forall x \in($ insert $a A) .($ is－a－variable $(\operatorname{subst}(\operatorname{Var} x)$ ？$\eta)))$
then obtain $x$ where $x \in$（insert a A）and $\neg$ is－a－variable（subst（Var $x$ ）
？$\eta)$
by auto
from «ᄀis－a－variable（subst（Var $x$ ）？$\eta$ ）〉 have $x \neq a$ by auto
from this and $\langle x \in$（insert a $A$ ）$\rangle$ have $x \in A$ by auto
from $\langle x \neq a\rangle$ have $($ subst $(\operatorname{Var} x)$ ？$\eta$ ）$=($ subst $(\operatorname{Var} x) \eta)$ by auto
from $\langle r e n a m i n g ~ \eta A\rangle$ and $\langle x \in A\rangle$ have is－a－variable（subst（Var $x) \eta$ ）
unfolding renaming－def by metis
from this and 〈 $\neg$ is－a－variable（subst（Var $x$ ）？$\eta$ ）〉
$\langle($ subst $(\operatorname{Var} x) ? \eta)=($ subst $(\operatorname{Var} x) \eta)\rangle$ show False by auto
qed
have $i i:(\forall x y .((x \in($ insert a $A)) \longrightarrow(y \in($ insert $a A)) \longrightarrow x \neq y$ $\longrightarrow($ subst $(\operatorname{Var} x) ? \eta) \neq(\operatorname{subst}(\operatorname{Var} y) ? \eta)))$
proof（rule ccontr）
assume $\neg(\forall x y .((x \in($ insert $a A)) \longrightarrow(y \in($ insert $a A)) \longrightarrow x \neq y$ $\longrightarrow($ subst $(\operatorname{Var} x) ? \eta) \neq($ subst $($ Var $y) ? \eta)))$
from this obtain $x y$ where $x \in$ insert a $A y \in$ insert a $A x \neq y$ （subst $(\operatorname{Var} x)$ ？$\eta$ ）$=($ subst $(\operatorname{Var} y)$ ？$\eta)$ by blast from $i$ obtain $y^{\prime}$ where（subst $($ Var $y)$ ？$\left.\eta\right)=\left(\right.$ Var $\left.y^{\prime}\right)$ using is－a－variable．simps using $\langle y \in$ insert $a A\rangle i s$－$a$－variable．elims（2）by auto
from $i$ obtain $x^{\prime}$ where $($ subst $(\operatorname{Var} x) ? \eta)=\left(\operatorname{Var} x^{\prime}\right)$
using is－a－variable．simps using $\langle x \in$ insert $a A\rangle$ is－a－variable．elims（2）by
auto
from $\left\langle(\operatorname{subst}(\operatorname{Var} x) ? \eta)=\left(\operatorname{Var} x^{\prime}\right)\right\rangle\left\langle(\right.$ subst $($ Var $y) ? \eta)=\left(\right.$ Var $\left.\left.y^{\prime}\right)\right\rangle$
$\langle($ subst $(\operatorname{Var} x) ? \eta)=($ subst $(\operatorname{Var} y) ? \eta)\rangle$ have $x^{\prime}=y^{\prime}$ by auto
have $x \neq a$
proof
assume $x=a$
from this and $\langle x \neq y\rangle$ and $\langle y \in$ insert $a A\rangle$ have $y \in A$ by auto
from this and $\langle x \neq y\rangle$ and $\langle x=a\rangle$ and $\left\langle(\right.$ subst $\left.(\operatorname{Var} y) ? \eta)=\left(\operatorname{Var} y^{\prime}\right)\right\rangle$
have $y^{\prime} \in$（subst－codomain $\eta$ A）unfolding subst－codomain－def by auto
from $\langle x=a\rangle$ and $\left\langle(\right.$ subst $\left.(\operatorname{Var} x) ? \eta)=\left(\operatorname{Var} x^{\prime}\right)\right\rangle$ have $x^{\prime}=n v$ by auto
from this and $\left\langle y^{\prime} \in(\right.$ subst－codomain $\left.\eta A)\right\rangle$ and $\left\langle x^{\prime}=y^{\prime}\right\rangle$ and $\langle n v \notin$ （subst－codomain $\eta A$ ）＞
show False by auto
qed
from this and $\langle x \in$ insert $a A\rangle$ have $x \in A$ and

```
        \((\) subst \((\operatorname{Var} x) ? \eta)=(\operatorname{subst}(\operatorname{Var} x) \eta)\) by auto
```

    have \(y \neq a\)
    proof
        assume \(y=a\)
        from this and \(\langle x \neq y\rangle\) and \(\langle x \in\) insert \(a A\rangle\) have \(x \in A\) by auto
        from this and \(\langle x \neq y\rangle\) and \(\langle y=a\rangle\) and \(\left\langle(\right.\) subst \(\left.(\operatorname{Var} x) ? \eta)=\left(\operatorname{Var} x^{\prime}\right)\right\rangle\)
            have \(x^{\prime} \in(\) subst-codomain \(\eta\) A) unfolding subst-codomain-def by auto
            from \(\langle y=a\rangle\) and \(\left\langle(\right.\) subst \((\operatorname{Var} y) ? \eta)=\left(\right.\) Var \(\left.\left.y^{\prime}\right)\right\rangle\) have \(y^{\prime}=n v\) by auto
            from this and \(\left\langle x^{\prime} \in(\right.\) subst-codomain \(\left.\eta A)\right\rangle\) and \(\left\langle x^{\prime}=y^{\prime}\right\rangle\) and \(\langle n v \notin\)
    (subst-codomain $\eta A$ )>
show False by auto
qed
from this and $\langle y \in$ insert $a A\rangle$ have $y \in A$ and
$($ subst $(\operatorname{Var} y) ? \eta)=($ subst $(\operatorname{Var} y) \eta)$ by auto
from 〈(subst $(\operatorname{Var} x)$ ? $\eta)=(\operatorname{subst}(\operatorname{Var} x) \eta)\rangle$
$\langle($ subst $(\operatorname{Var} y) ? \eta)=($ subst $(\operatorname{Var} y) \eta)\rangle$
$\langle($ subst $(\operatorname{Var} x) ? \eta)=($ subst $(\operatorname{Var} y) ? \eta)\rangle$
have $(\operatorname{subst}(\operatorname{Var} x) \eta)=($ subst $(\operatorname{Var} y) \eta)$ by auto
from this and $\langle x \in A\rangle$ and $\langle y \in A\rangle$ and $\langle r e n a m i n g ~ \eta A\rangle$ and $\langle x \neq y\rangle$ show
False
unfolding renaming-def by metis
qed
from $i$ ii have renaming ? $\eta$ (insert a $A$ ) unfolding renaming-def by auto
have $\left((\right.$ subst-codomain ? $\eta($ insert $\left.a A)) \cap V^{\prime}\right)=\{ \}$
proof (rule ccontr)
assume (subst-codomain ? $\eta($ insert a $A)) \cap V^{\prime} \neq\{ \}$
then obtain $x$ where $x \in\left(\right.$ subst-codomain ? $\eta($ insert $a A)$ ) and $x \in V^{\prime}$ by
auto
from $\langle x \in($ subst-codomain $? \eta($ insert $a A))\rangle$ obtain $x^{\prime}$ where $x^{\prime} \in($ insert $a$
A)
and subst $\left(\operatorname{Var} x^{\prime}\right) ? \eta=(\operatorname{Var} x)$ unfolding subst-codomain-def by blast
have $x^{\prime} \neq a$
proof
assume $x^{\prime}=a$
from this and $\left\langle\right.$ subst (Var $\left.\left.x^{\prime}\right) ? \eta=(\operatorname{Var} x)\right\rangle$ have $x=n v$ by auto
from this and $\left\langle x \in V^{\prime}\right\rangle$ and $\left\langle n v \notin\left(\right.\right.$ insert $\left.\left.a V^{\prime}\right)\right\rangle$ show False by auto
qed
from this and $\left\langle x^{\prime} \in(\right.$ insert $\left.a A)\right\rangle$ have $x^{\prime} \in A$ by auto
from $\left\langle x^{\prime} \neq a\right\rangle$ and $\left\langle\right.$ subst $\left.\left(\operatorname{Var} x^{\prime}\right) \quad ? \eta=(\operatorname{Var} x)\right\rangle$ have
$(\operatorname{Var} x)=\left(\right.$ subst $\left.\left(\operatorname{Var} x^{\prime}\right) \eta\right)$ by auto
from this and $\left\langle x^{\prime} \in A\right\rangle$ have $x \in$ subst-codomain $\eta A$ unfolding subst-codomain-def
by auto
from $\langle x \in$ subst-codomain $\eta A\rangle$ and $\left\langle(\right.$ subst-codomain $\eta A) \cap\left(\right.$ insert a $\left.V^{\prime}\right)$
$=\{ \}\rangle$ and $\left\langle x \in V^{\prime}\right\rangle$
show False by auto
qed
from this and 〈renaming ? $\eta$ (insert a $A$ )〉
show $\exists \eta$. renaming $\eta($ insert a $A) \wedge$ subst-codomain $\eta($ insert a $A) \cap V^{\prime}=$
\{\} by auto

```
    qed
qed
end
theory equational-clausal-logic
imports Main terms HOL-Library.Multiset
begin
```


## 3 Equational Clausal Logic

The syntax and semantics of clausal equational logic are defined as usual. Interpretations are congruences on binary trees.

### 3.1 Syntax

We first define the syntax of equational clauses.

```
datatype 'a equation \(=E q\) 'a trm 'a trm
fun lhs
    where lhs \((\) Comb t1 t2 \()=t 1\)
        lhs \((\) Var \(x)=(\) Var \(x) \mid\)
        lhs \((\) Const \(x)=(\) Const \(x)\)
fun rhs
    where rhs \((\) Comb t1 t2 \()=t 2 \mid\)
        rhs \((\operatorname{Var} x)=(\operatorname{Var} x) \mid\)
        rhs \((\) Const \(x)=(\) Const \(x)\)
datatype 'a literal \(=\) Pos 'a equation \(\mid\) Neg 'a equation
fun atom :: 'a literal \(\Rightarrow\) ' \(a\) equation
    where
        \((\operatorname{atom}(\operatorname{Pos} x))=x \mid\)
        \((\operatorname{atom}(\operatorname{Neg} x))=x\)
datatype sign \(=\) pos \(\mid\) neg
fun get-sign :: 'a literal \(\Rightarrow\) sign
    where
        \((\) get-sign \((\) Pos \(x))=\) pos
        \((\) get-sign \((N e g x))=n e g\)
    fun positive-literal :: 'a literal \(\Rightarrow\) bool
```

```
where
    \((\) positive-literal \((\) Pos \(x))=\) True \(\mid\)
    \((\) positive-literal \((\operatorname{Neg} x))=\) False
fun negative-literal \(::\) 'a literal \(\Rightarrow\) bool
    where
        \((\) negative-literal \((\) Pos \(x))=\) False \(\mid\)
        \((\) negative-literal \((\operatorname{Neg} x))=\) True
fun \(m k\)-lit \(::\) sign \(\Rightarrow\) 'a equation \(\Rightarrow\) 'a literal
    where
    \((m k\)-lit pos \(x)=(\) Pos \(x) \mid\)
    \((m k\)-lit neg \(x)=(\operatorname{Neg} x)\)
definition decompose-equation
    where decompose-equation e \(t s=(e=(E q t s) \vee(e=(E q s t)))\)
definition decompose-literal
    where decompose-literal \(L\) ts \(p=\)
        \((\exists e .((p=\operatorname{pos} \wedge(L=(\) Pos \(e)) \wedge\) decompose-equation et \(s)\)
        \(\vee(p=\operatorname{neg} \wedge(L=(\) Neg e) \() \wedge\) decompose-equation et \(s)))\)
fun subterms-of-eq
    where subterms-of-eq \((E q t s)=(\) subterms-of \(t \cup\) subterms-of \(s)\)
fun vars-of-eq
    where vars-of-eq \((E q t s)=(\) vars-of \(t \cup\) vars-of \(s)\)
lemma decompose-equation-vars:
    assumes decompose-equation ets
    shows vars-of \(t \subseteq\) vars-of-eq e
by (metis assms decompose-equation-def sup.cobounded1 sup-commute vars-of-eq.simps)
fun subterms-of-lit
    where
        subterms-of-lit (Pos e) \(=(\) subterms-of-eq e) \(\mid\)
        subterms-of-lit \((\) Neg e \()=(\) subterms-of-eq e)
fun vars-of-lit
    where
        vars-of-lit (Pos e) \(=(\) vars-of-eq e) \(\mid\)
        vars-of-lit \((\) Neg e) \(=(\) vars-of-eq e)
fun vars-of-cl
    where vars-of-cl \(C=\{x . \exists L . x \in(\) vars-of-lit \(L) \wedge L \in C\}\)
fun subterms-of-cl
    where subterms-of-cl \(C=\{x . \exists L . x \in(\) subterms-of-lit \(L) \wedge L \in C\}\)
```

Note that clauses are defined as sets and not as multisets (identical literals
are always merged).
type-synonym 'a clause $=$ 'a literal set
fun ground-clause :: 'a clause $\Rightarrow$ bool where

$$
(\text { ground-clause } C)=((\text { vars-of-cl } C)=\{ \})
$$

fun subst-equation $::$ ' $a$ equation $\Rightarrow$ ' $a$ subst $\Rightarrow$ 'a equation

## where

$$
\begin{aligned}
& \text { (subst-equation }(E q u v) \text { s) } \\
& \quad=(E q(\text { subst } u \text { s) }(\text { subst } v s))
\end{aligned}
$$

fun subst-lit :: 'a literal $\Rightarrow{ }^{\prime}$ a subst $\Rightarrow$ ' $a$ literal
where

$$
\begin{aligned}
& \text { (subst-lit }(\text { Pos e) s) } \\
& =(\text { Pos }(\text { subst-equation e s) }) \\
& (\text { subst-lit }(\text { Neg e) s) } \\
& =(\text { Neg (subst-equation e s) })
\end{aligned}
$$

fun subst-cl :: 'a clause $\Rightarrow$ 'a subst $\Rightarrow$ 'a clause
where

$$
(\text { subst-cl } C \text { s })=\left\{L .\left(\exists L^{\prime} .\left(L^{\prime} \in C\right) \wedge\left(L=\left(\text { subst-lit } L^{\prime} \text { s }\right)\right)\right)\right\}
$$

We establish some properties of the functions returning the set of variables occurring in an object.

```
lemma decompose-literal-vars:
    assumes decompose-literal L tsp
    shows vars-of t\subseteq vars-of-lit L
by (metis assms decompose-equation-vars decompose-literal-def vars-of-lit.simps(1)
vars-of-lit.simps(2))
lemma vars-of-cl-lem:
    assumes L}\in
    shows vars-of-lit L\subseteqvars-of-cl C
using assms by auto
lemma set-of-variables-is-finite-eq:
    shows finite (vars-of-eq e)
proof -
    obtain t and s}\mathrm{ where e=Eqts using equation.exhaust by auto
    then have vars-of-eq e=(vars-of t)\cup(vars-of s) by auto
    from this show ?thesis by auto
qed
lemma set-of-variables-is-finite-lit:
    shows finite (vars-of-lit l)
proof -
    obtain e where l= Pos e\veel=Neg e using literal.exhaust by auto
    then have vars-of-lit l=(vars-of-eq e) by auto
```

from this show ？thesis using set－of－variables－is－finite－eq by auto
qed
lemma set－of－variables－is－finite－cl：
assumes finite $C$
shows finite（vars－of－cl C）
proof－
let $? S=\{x . \exists l . x=$ vars－of－lit $l \wedge l \in C\}$
have vars－of－cl $C=\bigcup$ ？S by auto
from assms have finite ？S by auto
\｛ fix $x$ have $x \in ? S \Longrightarrow$ finite $x$ using set－of－variables－is－finite－lit by auto \}
from this and 〈finite ？S〉 have finite（ $\cup$ ？S）using finite－Union by auto
from this and «vars－of－cl $C=\bigcup$ ？$S$ 〉 show ？thesis by auto
qed
lemma subterm－lit－vars ：
assumes $u \in$ subterms－of－lit $L$
shows vars－of $u \subseteq$ vars－of－lit $L$
proof－
obtain $e$ where def－e：$L=($ Pos e）$\vee L=($ Neg e）and vars－of－lit $L=$ vars－of－eq $e$
by（metis negative－literal．elims（2）negative－literal．elims（3）
vars－of－lit．simps（1）vars－of－lit．simps（2））
obtain $t$ and $s$ where def－ts：$e=(E q t s) \vee e=(E q s t)$ and vars－of－eq $e=$ vars－of $t \cup$ vars－of $s$
by（metis equation．exhaust vars－of－eq．simps）
from this and 〈vars－of－lit $L=$ vars－of－eq e〉 have vars－of－lit $L=$ vars－of $t \cup$ vars－of $s$ by auto
from assms（1）and def－e def－ts have $u \in$ subterms－of $t \cup$ subterms－of $s$ by auto
from this have vars－of $u \subseteq$ vars－of $t \cup$ vars－of $s$
by（meson UnE sup．coboundedI1 sup．coboundedI2 vars－subterms－of）
from this and «vars－of－lit $L=$ vars－of $t \cup$ vars－of $s\rangle$ show ？thesis by auto qed
lemma subterm－vars ：
assumes $u \in$ subterms－of－cl $C$
shows vars－of $u \subseteq$ vars－of－cl $C$
proof－
from $\operatorname{assms}(1)$ obtain $L$ where $u \in$ subterms－of－lit $L$ and $L \in C$ by auto
from $\langle u \in$ subterms－of－lit $L\rangle$ have vars－of $u \subseteq$ vars－of－lit $L$ using subterm－lit－vars by auto
from $\langle L \in C\rangle$ have vars－of－lit $L \subseteq$ vars－of－cl $C$ using vars－of－cl．simps by auto
from this and «vars－of $u \subseteq$ vars－of－lit $L\rangle$ show ？thesis by auto
qed
We establish some basic properties of substitutions．
lemma subterm－cl－subst：
assumes $x \in($ subterms－of－cl $C)$
shows（subst x $\sigma$ ）$\in($ subterms－of－cl（subst－cl $C \sigma))$

```
proof -
    from assms(1) obtain L where L\inC and x\in subterms-of-lit L by auto
    from }\langleL\inC\rangle\mathrm{ have (subst-lit L }\sigma)\in(\mathrm{ subst-cl C }\sigma\mathrm{ ) by auto
    obtain e where L=(Pos e)\veeL=(Neg e) using literal.exhaust by auto
    then show ?thesis
    proof
        assume L = (Pos e)
        from this and }\langlex\in\mathrm{ subterms-of-lit L> have x subterms-of-eq e by auto
        from <L = (Pos e)\rangle have (subst-lit L \sigma) =(Pos (subst-equation e \sigma))
            by auto
    obtain ts where e=(Eqts) using equation.exhaust by auto
    from this have (subst-equation e \sigma)=(Eq(subst t \sigma) (subst s \sigma))
        by auto
    from }\langlex\in\mathrm{ subterms-of-eq e〉 and }\langlee=(Eqts)\rangle have x\in subterms-of t\veex
subterms-of s by auto
    then show ?thesis
    proof
            assume x f subterms-of t
            then have occurs-in x t by auto
            then obtain p}\mathrm{ where subterm t px unfolding occurs-in-def by blast
            from this have subterm (subst t \sigma) p (subst x \sigma)
                using substs-preserve-subterms by auto
            from this have occurs-in (subst x \sigma) (subst t \sigma) unfolding occurs-in-def by
auto
            then have (subst x \sigma) \in subterms-of (subst t \sigma) by auto
            then have (subst x \sigma)\in subterms-of-eq (Eq (subst t \sigma) (subst s \sigma)) by auto
            from this and }\langleL=(\mathrm{ Pos e)> and }\langlee=Eqt s
                have (subst x \sigma) \in(subterms-of-lit (subst-lit L \sigma)) by auto
            from this and «(subst-lit L \sigma) \in(subst-cl C \sigma)\rangle
                show (subst x \sigma) \in subterms-of-cl (subst-cl C \sigma) by auto
    next
        assume }x\in\mathrm{ subterms-of s
        then have occurs-in x s by auto
        then obtain p}\mathrm{ where subterm s p x unfolding occurs-in-def by blast
        from this have subterm (subst s\sigma) p(subst x \sigma)
            using substs-preserve-subterms by auto
        from this have occurs-in (subst x \sigma) (subst s \sigma) unfolding occurs-in-def by
auto
    then have (subst x \sigma)\in subterms-of (subst s \sigma) by auto
    then have (subst x \sigma) \in subterms-of-eq (Eq (subst t \sigma) (subst s \sigma)) by auto
    from this and }\langleL=(\mathrm{ Pos e)> and }\langlee=Eqts
        have (subst x \sigma) ( (subterms-of-lit (subst-lit L \sigma)) by auto
    from this and «(subst-lit L \sigma) \in(subst-cl C \sigma)>
        show (subst x \sigma) \in subterms-of-cl (subst-cl C \sigma) by auto
    qed
    next
    assume L = (Neg e)
    from this and }\langlex\in\mathrm{ subterms-of-lit L> have x subterms-of-eq e by auto
    from }\langleL=(Neg e)\rangle have (subst-lit L \sigma)=(Neg (subst-equation e \sigma))
```

```
        by auto
    obtain ts where e=(Eqts) using equation.exhaust by auto
    from this have (subst-equation e \sigma)=(Eq(subst t \sigma) (subst s \sigma))
        by auto
    from \langlex \in subterms-of-eq e> and }\langlee=(Eqts)\rangle have x\in subterms-of t\veex
subterms-of s by auto
    then show ?thesis
    proof
        assume x fubterms-of t
        then have occurs-in xt by auto
        then obtain p}\mathrm{ where subterm t px unfolding occurs-in-def by blast
        from this have subterm (subst t \sigma) p (subst x \sigma)
            using substs-preserve-subterms by auto
        from this have occurs-in (subst x \sigma) (subst t \sigma) unfolding occurs-in-def by
auto
    then have (subst x \sigma) \in subterms-of (subst t \sigma) by auto
    then have (subst x \sigma)\in subterms-of-eq (Eq (subst t \sigma) (subst s \sigma)) by auto
    from this and }\langleL=(Nege)\rangle and <e=Eqt s
            have (subst x \sigma)\in(subterms-of-lit (subst-lit L \sigma)) by auto
            from this and «(subst-lit L \sigma) \in(subst-cl C \sigma)>
                show (subst x \sigma) \in subterms-of-cl (subst-cl C \sigma) by auto
    next
        assume }x\in\mathrm{ subterms-of s
        then have occurs-in x s by auto
        then obtain p}\mathrm{ where subterm s p x unfolding occurs-in-def by blast
        from this have subterm (subst s \sigma) p (subst x \sigma)
            using substs-preserve-subterms by auto
        from this have occurs-in (subst x \sigma) (subst s \sigma) unfolding occurs-in-def by
auto
            then have (subst x \sigma) \in subterms-of (subst s \sigma) by auto
            then have (subst x \sigma) \in subterms-of-eq (Eq (subst t \sigma) (subst s \sigma)) by auto
            from this and }\langleL=(Neg e)\rangle and <e=Eqt s
                have (subst x \sigma) ( (subterms-of-lit (subst-lit L \sigma)) by auto
            from this and «(subst-lit L \sigma) \in(subst-cl C \sigma)>
                show (subst x \sigma) \in subterms-of-cl (subst-cl C \sigma) by auto
    qed
    qed
qed
lemma ground-substs-yield-ground-clause:
    assumes ground-on (vars-of-cl C) \sigma
    shows ground-clause (subst-cl C \sigma)
proof (rule ccontr)
    let ?D = (subst-cl C \sigma)
    let ?V = (vars-of-cl C)
    assume }\neg\mathrm{ (ground-clause ?D)
    then obtain x where }x\in(vars-of-cl ?D) by aut
    then obtain l where l\inC and x\in(vars-of-lit (subst-lit l \sigma)) by auto
    from <l }\inC>\mathrm{ have vars-of-lit l }\subseteq\mathrm{ vars-of-cl C by auto
```

obtain $e$ where $l=$ Pos $e \vee l=$ Neg e using literal．exhaust by auto
then have vars－of－lit $l=$ vars－of－eq e by auto
let ？$l^{\prime}=($ subst－lit $l \sigma)$
let $? e^{\prime}=($ subst－equation $e \sigma)$
obtain $t$ and $s$ where $e=E q t s$ using equation．exhaust by auto
then have vars－of－eq $e=$ vars－of $t \cup$ vars－of $s$ by auto
let ${ }^{\prime} t^{\prime}=($ subst $t \sigma)$
let $? s^{\prime}=($ subst $s \sigma)$
from $\langle e=E q t s\rangle$ have ？$e^{\prime}=\left(E q ? t^{\prime} ? s^{\prime}\right)$ by auto
from $\langle l=$ Pos $e \vee l=N e g e\rangle$ have $? l^{\prime}=P o s ? e^{\prime} \vee ? l^{\prime}=N e g ? e^{\prime}$ by auto
from $\langle l \in C\rangle$ have ？$l^{\prime} \in ? D$ by auto
from 〈？$l^{\prime}=$ Pos ？$e^{\prime} \vee ? l^{\prime}=$ Neg ？$\left.e^{\prime}\right\rangle$ and $\left\langle x \in\left(\right.\right.$ vars－of－lit ？l $\left.\left.l^{\prime}\right)\right\rangle$
have $x \in\left(\right.$ vars－of－eq ？$\left.e^{\prime}\right)$ by auto
from this and $\left\langle ? e^{\prime}=\left(E q ? t^{\prime} ? s^{\prime}\right)\right\rangle$ have $x \in\left(\right.$ vars－of $? t^{\prime} \cup$ vars－of $\left.? s^{\prime}\right)$ by auto
then have $i: \neg$（ground－term ？$\left.t^{\prime}\right) \vee \neg\left(\right.$ ground－term ？$\left.s^{\prime}\right)$ unfolding ground－term－def by auto
from 〈vars－of－eq $e=$ vars－of $t \cup$ vars－of $s\rangle$ and $\langle v a r s-o f-l i t ~ l=v a r s-o f-e q ~ e 〉$ and
$\langle$ vars－of－lit $l \subseteq ? V\rangle$ have vars－of $t \subseteq$ ？$V$ and vars－of $s \subseteq$ ？$V$ by auto
from 〈vars－of $t \subseteq$ ？$V\rangle$ and 〈ground－on？$V \sigma\rangle$ have ground－on（vars－of $t$ ）$\sigma$ unfolding ground－on－def by auto
then have ii：ground－term ？$t^{\prime}$ using ground－instance by auto
from 〈vars－of $s \subseteq$ ？$V\rangle$ and 〈ground－on？$V \sigma\rangle$ have ground－on（vars－of $s$ ）$\sigma$ unfolding ground－on－def by auto
then have iii：ground－term？$s^{\prime}$ using ground－instance by auto
from $i$ and $i i$ and $i i i$ show False by auto
qed
lemma ground－clauses－and－ground－substs：
assumes ground－clause（subst－cl C $\sigma$ ）
shows ground－on（vars－of－cl C）$\sigma$
proof（rule ccontr）
assume $\neg$ ground－on（vars－of－cl C）$\sigma$
from this obtain $x$ where $x \in$ vars－of－cl $C$ and $\neg$ ground－term（subst（Var $x$ ） $\sigma$ ）
unfolding ground－on－def by auto
from $\langle\neg$ ground－term（subst（Var $x) \sigma$ ） obtain $y$ where $y \in$ vars－of（subst（Var $x$ ）$\sigma$ ）unfolding ground－term－def by auto
from $\langle x \in$ vars－of－cl $C\rangle$ obtain $L$ where $L \in C$ and $x \in$ vars－of－lit $L$ by auto
from $\langle x \in$ vars－of－lit $L\rangle$ obtain $e$ where $L=$ Pos $e \vee L=N e g e$ and $x \in$ vars－of－eq e
by（metis vars－of－lit．elims）
from $\langle x \in$ vars－of－eq $e\rangle$ obtain $t s$ where $e=(E q t s)$ and $x \in$ vars－of $t \cup$ vars－of $s$
by（metis vars－of－eq．elims）
from this have $x \in$ vars－of $t \vee x \in$ vars－of $s$ by auto
then have $y \in$ vars－of－eq（subst－equation e $\sigma$ ）
proof

```
    assume x vars-of t
    have i: vars-of (subst t \sigma)=\bigcup{V.\existsx.x vars-of t\wedgeV=vars-of (subst
(Var x)\sigma)}
            using vars-of-instances [of t \sigma] by meson
            from <x \in vars-of t> }
                have vars-of (subst (Var x) \sigma)\subseteq vars-of (subst t \sigma)
            by auto
                            from this and <y\in vars-of (subst (Var x) \sigma)\rangle\langlee= (Eqts)\rangle show ?thesis by
auto
    next
            assume }x\in\mathrm{ vars-of s
            have i: vars-of (subst s \sigma)=\bigcup{V.\existsx.x vars-of s\wedge V = vars-of (subst
(Var x)\sigma)}
            using vars-of-instances [of s \sigma] by meson
            from }\langlex\in\mathrm{ vars-of s> i
                have vars-of (subst (Var x) \sigma)\subseteq vars-of (subst s \sigma)
            by auto
                            from this and }\langley\invars-of (subst (Var x) \sigma)\rangle\langlee=(Eqts)\rangle show?thesis by
auto
    qed
    from this and }\langleL=P\mathrm{ Pos e }\veeL=Neg e> have y \in vars-of-lit (subst-lit L \sigma)
        by auto
    from this and }\langleL\inC\rangle have y\invars-of-cl (subst-cl C \sigma) by aut
    from this and assms(1) show False by auto
qed
lemma ground-instance-exists:
    assumes finite C
    shows }\exists\sigma\mathrm{ . (ground-clause (subst-cl C }\sigma\mathrm{ ))
proof -
    let ?V = (vars-of-cl C)
    from assms have finite ?V using set-of-variables-is-finite-cl by auto
    from this obtain \sigma where ground-on ?V \sigma
            using ground-subst-exists by blast
    let ?D = (subst-cl C \sigma)
    from «ground-on ?V \sigma〉 have (ground-clause ?D) using ground-substs-yield-ground-clause
[of C \sigma] by auto
    then show ?thesis by auto
qed
lemma composition-of-substs:
    shows (subst (subst t \sigma) \eta)
    =(subst t (comp \sigma \eta))
by simp
lemma composition-of-substs-eq :
    shows (subst-equation (subst-equation e \sigma) \eta)
    = (subst-equation e (comp \sigma \eta))
by (metis subst-equation.simps composition-of-substs vars-of-eq.elims)
```

```
lemma composition-of-substs-lit :
    shows (subst-lit (subst-lit l \sigma) \eta)
        =(subst-lit l (comp \sigma \eta))
by (metis subst-lit.simps(1) subst-lit.simps(2)
            composition-of-substs-eq positive-literal.cases)
lemma composition-of-substs-cl :
    shows (subst-cl (subst-cl C \sigma) \eta)
        =(subst-cl C (comp \sigma \eta))
proof -
    let ?f = (\lambdax. (subst-lit (subst-lit x \sigma) \eta))
    let ?f'=(\lambdax.(subst-lit x (comp \sigma \eta)))
    have }\foralll\mathrm{ . (?f l) = (?f'l) using composition-of-substs-lit by auto
    then show ?thesis by auto
qed
lemma substs-preserve-ground-lit :
    assumes ground-clause C
    assumes }y\in
    shows subst-lit y \sigma=y
proof -
            obtain t and s where y=Pos (Eq t s)\vee y = Neg (Eq t s)
                by (metis subst-equation.elims get-sign.elims)
    from this have vars-of t\subseteqvars-of-lit y by auto
    from this and }\langley\inC> have vars-of t\subseteqvars-of-cl C by aut
        from this and assms(1) have ground-term t unfolding ground-term-def by
auto
    then have subst t \sigma=t using substs-preserve-ground-terms by auto
    from <y = Pos (Eqts)\veey=Neg(Eqts)\rangle have vars-of s\subseteqvars-of-lit y by
auto
            from this and }\langley\inC\rangle\mathrm{ have vars-of s}\subseteq\mathrm{ vars-of-cl C by auto
            from this and assms(1) have ground-term s unfolding ground-term-def by
auto
            then have subst s \sigma=s using substs-preserve-ground-terms by auto
            from «subst s \sigma=s` and <subst t \sigma = t` and «y = Pos (Eqt s)\veey=Neg
(Eqt s)>
            show subst-lit y \sigma=y by auto
qed
lemma substs-preserve-ground-clause :
    assumes ground-clause C
    shows subst-cl C \sigma = C
proof
    show subst-cl C \sigma\subseteqC
    proof
        fix x assume x fubst-cl C \sigma
        then obtain }y\mathrm{ where }y\inC\mathrm{ and }x=\mathrm{ subst-lit }y\sigma\mathrm{ by auto
        from assms(1) and }\langley\inC\rangle\mathrm{ and }\langlex=subst-lit y \sigma
```

have $x=y$ using substs-preserve-ground-lit by auto
from this and $\langle y \in C\rangle$ show $x \in C$ by auto
qed
next
show $C \subseteq$ subst-cl $C \sigma$
proof
fix $x$ assume $x \in C$
then have subst-lit $x \sigma \in$ subst-cl $C \sigma$ by auto
from $\operatorname{assms}(1)$ and $\langle x \in C\rangle$ have $x=$ subst-lit $x \sigma$
using substs-preserve-ground-lit [of C x] by auto
from this and $\langle x \in C\rangle$ show $x \in$ subst-cl $C \sigma$ by auto
qed
qed
lemma substs-preserve-finiteness :
assumes finite $C$
shows finite (subst-cl C $\sigma$ )
proof -
from assms(1) show ?thesis using Finite-Set.finite-imageI by auto
qed
We prove that two equal substitutions yield the same objects.
lemma subst-eq-eq :
assumes subst-eq $\sigma \eta$
shows subst-equation e $\sigma=$ subst-equation e $\eta$
proof -
obtain $t$ and $s$ where $e=E q t s$ using equation.exhaust by auto
from assms(1) have subst s $\sigma=$ subst $s \eta$ by auto
from $\operatorname{assms}(1)$ have subst $t \sigma=$ subst $t \eta$ by auto
from 〈subst s $\sigma=$ subst s $\eta\rangle\langle s u b s t t \sigma=$ subst $t \eta\rangle$
and $\langle e=E q t s\rangle$ show ?thesis by auto
qed
lemma subst-eq-lit :
assumes subst-eq $\sigma \eta$
shows subst-lit $l \sigma=$ subst-lit $l \eta$
proof -
obtain $e$ where $l=$ Pos e $\vee l=N e g e$ using literal.exhaust by auto
from assms(1) have subst-equation e $\sigma=$ subst-equation e $\eta$ using subst-eq-eq
by auto
from this and $<l=$ Pos $e \vee l=N e g e\rangle$ show ?thesis by auto
qed
lemma subst-eq-cl:
assumes subst-eq $\sigma \eta$
shows subst-cl $C \sigma=$ subst-cl $C \eta$
proof (rule ccontr)
assume subst-cl $C \sigma \neq$ subst-cl $C \eta$
then obtain $L$ where $L \in C$ and subst-lit $L \sigma \neq$ subst-lit $L \eta$

```
    by force
    from assms(1) and «subst-lit L \sigma\not= subst-lit L \eta`
    show False using subst-eq-lit by auto
qed
lemma coincide-on-eq :
    assumes coincide-on \sigma \eta (vars-of-eq e)
    shows subst-equation e \sigma=subst-equation e }
proof -
    obtain t and s where e=Eqts using equation.exhaust by auto
    then have vars-of t\subseteqvars-of-eq e by simp
    from this and «coincide-on \sigma \eta (vars-of-eq e)> have coincide-on \sigma \eta (vars-of t)
        unfolding coincide-on-def by auto
    from this have subst t \sigma = subst t }\eta\mathrm{ using coincide-on-term by auto
    from }\langlee=Eqts\rangle\mathrm{ have vars-of s}\subseteq\mathrm{ vars-of-eq e by simp
    from this and <coincide-on \sigma \eta (vars-of-eq e)> have coincide-on \sigma \eta (vars-of s)
    unfolding coincide-on-def by auto
    from this have subst s \sigma= subst s \eta using coincide-on-term by auto
    from <subst t \sigma = subst t }\eta\mathrm{ 〉
    and <subst s \sigma = subst s \eta\rangle
    and }\langlee=Eqts\rangle\mathrm{ show ?thesis by auto
qed
lemma coincide-on-lit :
    assumes coincide-on \sigma \eta (vars-of-lit l)
    shows subst-lit l \sigma = subst-lit l \eta
proof -
    obtain e where l=Pos e\veel=Neg e using literal.exhaust by auto
    then have vars-of-eq e\subseteqvars-of-lit l by auto
    from this and <coincide-on \sigma \eta (vars-of-lit l)〉 have coincide-on \sigma \eta (vars-of-eq
e)
    unfolding coincide-on-def by auto
    from this have subst-equation e \sigma= subst-equation e }
    using coincide-on-eq by auto
    from this and }<l=Pose\veel=Neg e\rangle\mathrm{ show ?thesis by auto
qed
lemma coincide-on-cl :
    assumes coincide-on \sigma \eta (vars-of-cl C)
    shows subst-cl C \sigma = subst-cl C \eta
proof (rule ccontr)
    assume subst-cl C \sigma}\not=\mathrm{ subst-cl C }
    then obtain L where L\inC and subst-lit L \sigma\not= subst-lit L \eta
    by force
    from <L CC> have vars-of-lit L}\subseteq\mathrm{ vars-of-cl C by auto
    from this and assms have coincide-on \sigma }\eta\mathrm{ (vars-of-lit L) unfolding coin-
cide-on-def by auto
    from this and «subst-lit L \sigma\not= subst-lit L \eta>
    show False using coincide-on-lit by auto
```


## qed

### 3.2 Semantics

Interpretations are congruences on the set of terms.
definition fo-interpretation :: 'a binary-relation-on-trms $\Rightarrow$ bool where
$($ fo-interpretation $x)=($ congruence $x)$
fun validate-ground-eq :: 'a binary-relation-on-trms $\Rightarrow{ }^{\prime}$ 'a equation $\Rightarrow$ bool where
(validate-ground-eq $I(E q t s)=(I t s))$
fun validate-ground-lit :: 'a binary-relation-on-trms $\Rightarrow{ }^{\prime}$ ' literal $\Rightarrow$ bool where
validate-ground-lit $I($ Pos $E)=($ validate-ground-eq I E) $\mid$
validate-ground-lit I (Neg $E)=(\neg($ validate-ground-eq I E) $)$
fun validate-ground-clause :: 'a binary-relation-on-trms $\Rightarrow{ }^{\prime}$ 'a clause $\Rightarrow$ bool where
validate-ground-clause $I C=(\exists L .(L \in C) \wedge($ validate-ground-lit I L) $)$
fun validate-clause :: 'a binary-relation-on-trms $\Rightarrow{ }^{\prime} a$ clause $\Rightarrow$ bool where validate-clause $I C=(\forall s$. (ground-clause (subst-cl C s)) $\longrightarrow($ validate-ground-clause I (subst-cl C s) ))
fun validate-clause-set :: 'a binary-relation-on-trms $\Rightarrow$ 'a clause set $\Rightarrow$ bool where
validate-clause-set $I S=(\forall C .(C \in S \longrightarrow($ validate-clause $I C)))$
definition clause-entails-clause $::$ 'a clause $\Rightarrow$ 'a clause $\Rightarrow$ bool where
clause-entails-clause $C D=(\forall I$. (fo-interpretation $I \longrightarrow$ validate-clause $I C \longrightarrow$ validate-clause I D) )
definition set-entails-clause :: 'a clause set $\Rightarrow{ }^{\prime}$ 'a clause $\Rightarrow$ bool where
set-entails-clause $S C=(\forall I$. (fo-interpretation $I \longrightarrow$ validate-clause-set $I S \longrightarrow$ validate-clause I C))
definition satisfiable-clause-set $::$ ' $a$ clause set $\Rightarrow$ bool where
$($ satisfiable-clause-set $S)=(\exists I .($ fo-interpretation $I) \wedge($ validate-clause-set $I S))$
We state basic properties of the entailment relation.
lemma set-entails-clause-member:
assumes $C \in S$
shows set-entails-clause $S C$

```
proof (rule ccontr)
    assume \neg ?thesis
    from this obtain I where fo-interpretation I validate-clause-set I S ᄀvali-
date-clause I C
    unfolding set-entails-clause-def by auto
    from〈validate-clause-set I S` and assms(1) \prec\neg validate-clause I C` show False
by auto
qed
lemma instances-are-entailed :
    assumes validate-clause I C
    shows validate-clause I (subst-cl C \sigma)
proof (rule ccontr)
    assume \negvalidate-clause I (subst-cl C \sigma)
    then obtain }
        where \negvalidate-ground-clause I (subst-cl (subst-cl C \sigma) \eta)
            and ground-clause (subst-cl (subst-cl C \sigma) \eta)
    by auto
    then have i: \negvalidate-ground-clause I (subst-cl C (comp \sigma \eta))
    using composition-of-substs-cl by metis
    from <ground-clause (subst-cl (subst-cl C \sigma) \eta)>
        have ii: ground-clause (subst-cl C (comp \sigma \eta))
        using composition-of-substs-cl by metis
    from i and ii have \negvalidate-clause I C by auto
    from «\negvalidate-clause I C` and <validate-clause I C` show False by blast
qed
```

We prove that two equivalent substitutions yield equivalent objects.
lemma equivalent-on-eq :
assumes equivalent-on $\sigma \eta$ (vars-of-eq e) I
assumes fo-interpretation I
shows (validate-ground-eq I (subst-equation e $\sigma$ ) $)=($ validate-ground-eq $I$ (subst-equation
e $\eta$ ))
proof -
obtain $t$ and $s$ where $e=E q t s$ using equation.exhaust by auto
then have vars-of $t \subseteq$ vars-of-eq e by simp
from this and assms(1) have equivalent-on $\sigma \eta$ (vars-of $t) I$
unfolding equivalent-on-def by auto
from this assms(2)
have $I$ (subst $t \sigma$ ) (subst $t \eta$ ) using equivalent-on-term
unfolding fo-interpretation-def by auto
from $\langle e=E q t s\rangle$ have vars-of $s \subseteq$ vars-of-eq e by simp
from this and «equivalent-on $\sigma \eta$ (vars-of-eq e) $I\rangle$ have equivalent-on $\sigma \eta$ (vars-of
s) $I$
unfolding equivalent-on-def by auto
from this assms(2) have $I$ (subst $s \sigma$ ) (subst s $\eta$ )
using equivalent-on-term unfolding fo-interpretation-def by auto
from $\operatorname{assms}(2)\langle I$ (subst $t \sigma$ ) (subst $t \eta$ ) 〉
and $\langle I$ (subst $s \sigma)($ subst $s \eta)\rangle$
and $\langle e=E q t s\rangle$ show ?thesis unfolding fo-interpretation-def congruence-def equivalence-relation-def
transitive-def symmetric-def reflexive-def
by (metis (full-types) subst-equation.simps validate-ground-eq.simps)
qed
lemma equivalent-on-lit :
assumes equivalent-on $\sigma \eta$ (vars-of-lit l) I
assumes fo-interpretation I
shows (validate-ground-lit I (subst-lit l $\sigma$ ))
$=($ validate-ground-lit I (subst-lit l $\eta))$
proof -
obtain $e$ where $l=$ Pos $e \vee l=N e g$ e using literal.exhaust by auto
then have vars-of-eq $e \subseteq$ vars-of-lit $l$ by auto
from this and 〈equivalent-on $\sigma \eta$ (vars-of-lit $l$ ) $I\rangle$ have equivalent-on $\sigma \eta$
(vars-of-eq e) I
unfolding equivalent-on-def by auto
from this assms(2) have (validate-ground-eq I (subst-equation e $\sigma$ )) $=($ validate-ground-eq $I$ (subst-equation e $\eta$ ))
using equivalent-on-eq by auto
from this and $\langle l=$ Pos $e \vee l=N e g e\rangle$ show ?thesis by auto
qed
lemma equivalent-on-cl :
assumes equivalent-on $\sigma \eta(v a r s-o f-c l C) I$
assumes fo-interpretation I
shows (validate-ground-clause I (subst-cl C $\sigma$ ))
$=($ validate-ground-clause $I$ (subst-cl C $\eta$ ) $)$
proof (rule ccontr)
assume (validate-ground-clause $I$ (subst-cl C $\sigma$ ))
$\neq($ validate-ground-clause I (subst-cl C $\eta$ ) )
then obtain $L$ where $L \in C$ and (validate-ground-lit $I$ (subst-lit $L \sigma$ ))
$\neq($ validate-ground-lit I (subst-lit $L \eta)$ )
by force
from $\langle L \in C\rangle$ have vars-of-lit $L \subseteq$ vars-of-cl $C$ by auto
from this and assms have equivalent-on $\sigma \eta$ (vars-of-lit $L$ ) $I$ unfolding equiv-alent-on-def by auto
from this assms(2) and 〈(validate-ground-lit I (subst-lit $L \sigma$ ))
$\neq($ validate-ground-lit I (subst-lit L $\eta$ ) ) >
show False using equivalent-on-lit by metis
qed
end
theory superposition
imports Main terms equational-clausal-logic well-founded-continued HOL-Library.Multiset multisets-continued

## begin

## 4 Definition of the Superposition Calculus

### 4.1 Extended Clauses

An extended clause is a clause associated with a set of terms. The intended meaning is that the terms occurring in the attached set are assumed to be in normal form: any application of the superposition rule on these terms is therefore useless and can be blocked. Initially the set of irreducible terms attached to each clause is empty. At each inference step, new terms can be added or deleted from this set.

```
datatype 'a eclause \(=E c l\) 'a clause 'a trm set
fun subst-ecl
where
    (subst-ecl \((\) Ecl C S \() \sigma)=\)
    \((E c l(\) subst-cl C \(\sigma)\{s .(\exists t .(s=(\) subst \(t \sigma) \wedge t \in S))\})\)
fun cl-ecl
where
    \((c l-e c l(E c l C X))=C\)
fun trms-ecl
where
    (trms-ecl \((E c l C X))=X\)
definition renaming-cl
    where renaming-cl \(C D=(\exists \eta\). (renaming \(\eta(\) vars-of-cl \((\) cl-ecl \(C))) \wedge D=\)
(subst-ecl \(C \eta\) ))
definition closed-under-renaming
    where closed-under-renaming \(S=(\forall C D\).
        \((C \in S) \longrightarrow(\) renaming-cl \(C D) \longrightarrow(D \in S))\)
definition variable-disjoint
where (variable-disjoint CD) \(=((\) vars-of-cl \((\) cl-ecl \(C)) \cap(\) vars-of-cl \((c l-e c l ~ D))\)
\(=\{ \}\) )
```


### 4.2 Orders and Selection Functions

We assume that the set of variables is infinite (so that shared variables can be renamed away) and that the following objects are given:
(i) A term ordering that is total on ground terms, well-founded and closed under contextual embedding and substitutions. This ordering is used as
usual to orient equations and to restrict the application of the replacement rule.
(ii) A selection function, mapping each clause to a (possibly empty) set of negative literals. We assume that this selection function is closed under renamings.
(iii) A function mapping every extended clause to an order on positions, which contains the (reversed) prefix ordering. This order allows one to control the order in which the subterms are rewritten (terms occurring at minimal positions are considered with the highest priority).
(iv) A function filter-trms that allows one to filter away some of the terms attached to a given extended clause (it can be used for instance to remove terms if the set becomes too big). The standard superposition calculus corresponds to the case where this function always returns the empty set.

```
locale basic-superposition \(=\)
    fixes trm-ord :: ('a trm \(\times\) 'a trm) set
    fixes sel :: 'a clause \(\Rightarrow\) 'a clause
    fixes pos-ord \(::\) 'a eclause \(\Rightarrow\) 'a trm \(\Rightarrow\) (position \(\times\) position) set
    fixes vars :: ' \(a\) set
    fixes filter-trms :: 'a clause \(\Rightarrow{ }^{\prime}\) 'a trm set \(\Rightarrow\) 'a trm set
    assumes
        trm-ord-wf : (wf trm-ord)
    and trm-ord-ground-total :
        \((\forall x y\). \(((\) ground-term \(x) \longrightarrow(\) ground-term \(y) \longrightarrow x \neq y\)
        \(\longrightarrow((x, y) \in\) trm-ord \(\vee(y, x) \in\) trm-ord \()))\)
    and trm-ord-trans : trans trm-ord
    and trm-ord-irrefl : irrefl trm-ord
    and trm-ord-reduction-left : \(\forall x 1\) x2 \(y .(x 1, x 2) \in\) trm-ord
        \(\longrightarrow((\) Comb \(x 1 y),(\) Comb \(x 2 y)) \in\) trm-ord
    and trm-ord-reduction-right \(: \forall x 1\) x2 \(y .(x 1, x 2) \in\) trm-ord
        \(\longrightarrow((\) Comb y x1),(Comb y x2) ) \(\in\) trm-ord
    and trm-ord-subterm : \(\forall x 1\) x2. \((x 1,(\) Comb \(x 1\) x2 \()) \in\) trm-ord
        \(\wedge(x 2,(\) Comb \(x 1\) x2 \()) \in\) trm-ord
    and trm-ord-subst:
    \(\forall s x y .(x, y) \in\) trm-ord \(\longrightarrow((\) subst \(x\) s \(),(\) subst \(y\) s \()) \in\) trm-ord
    and pos-ord-irrefl : \((\forall x y\). (irrefl (pos-ord \(x y))\) )
    and pos-ord-trans : \((\forall x\). (trans (pos-ord \(x y))\) )
    and pos-ord-prefix : \(\forall x y\) p qr. \(((q, p) \in(\) pos-ord \(x y) \longrightarrow((\) append \(q r), p) \in\)
(pos-ord \(x y\) ))
    and pos-ord-nil : \(\forall x y p .(p \neq\) Nil \() \longrightarrow(p\), Nil \() \in(\) pos-ord \(x y)\)
    and sel-neg: \((\forall x .((\) sel \((\) cl-ecl \(x)) \subseteq(\) cl-ecl \(x))\)
        \(\wedge(\forall y \in \operatorname{sel}(\operatorname{cl-ecl} x) .(\) negative-literal \(y)))\)
    and sel-renaming: \(\forall \eta C\). \(((\) renaming \(\eta(\) vars-of-cl \(C)) \longrightarrow\) sel \(C=\{ \} \longrightarrow\) sel
\((\) subst-cl \(C \eta)=\{ \}\) )
    and infinite-vars: \(\neg\) (finite vars)
    and filter-trms-inclusion: filter-trms \(C E \subseteq E\)
begin
```

We provide some functions to decompose a literal in a way that is compatible with the ordering and establish some basic properties.

```
definition orient-lit \(::\) 'a literal \(\Rightarrow{ }^{\prime}\) 'a trm \(\Rightarrow\) 'a trm \(\Rightarrow\) sign \(\Rightarrow\) bool
where
    (orient-lit Luvs) =
        \(((((L=(\operatorname{Pos}(E q u v))) \vee(L=(\operatorname{Pos}(E q v u)))) \wedge((u, v) \notin \operatorname{trm}\)-ord \() \wedge(s=\)
pos))
    \((((L=(N e g(E q u v))) \vee(L=(\operatorname{Neg}(E q v u)))) \wedge((u, v) \notin\) trm-ord \() \wedge(s=\)
\(n e g))\) )
definition orient-lit-inst \(::\) ' \(a\) literal \(\Rightarrow{ }^{\prime} a \operatorname{trm} \Rightarrow{ }^{\prime} a \operatorname{trm} \Rightarrow\) sign \(\Rightarrow\) ' \(a\) subst \(\Rightarrow\)
bool
where
    (orient-lit-inst Luvs \(\begin{gathered}\text { ) })= \\ \end{gathered}\)
        \(((((L=(\operatorname{Pos}(E q u v))) \vee(L=(\operatorname{Pos}(E q v u))))\)
        \(\wedge(((\) subst \(u \sigma),(\) subst \(v \sigma)) \notin\) trm-ord \() \wedge(s=\) pos \())\)
    V
    \((((L=(N e g(E q u v))) \vee(L=(N e g(E q v u)))) \wedge(((\) subst \(u \sigma),(\) subst \(v \sigma))\)
    \(\notin\) trm-ord \() \wedge(s=\) neg \()))\)
lemma lift-orient-lit:
    assumes orient-lit-inst Ltsp \(\sigma\)
    shows orient-lit (subst-lit \(L \sigma\) ) (subst \(t \sigma\) ) (subst s \(\sigma\) ) \(p\)
unfolding orient-lit-inst-def orient-lit-def using assms orient-lit-inst-def by auto
```

lemma orient-lit-vars:
assumes orient-lit Ltsp
shows vars-of $t \subseteq$ vars-of-lit $L \wedge$ vars-of $s \subseteq$ vars-of-lit $L$
proof -
have $p=n e g \vee p=$ pos using sign.exhaust by auto
then show ?thesis
proof
assume $p=n e g$
from this and assms(1) have $(L=N e g(E q t s)) \vee(L=(N e g(E q s t)))$
unfolding orient-lit-def by auto
then show ?thesis
proof
assume $L=\operatorname{Neg}(E q t s)$
then have vars-of-lit $L=$ vars-of $t \cup$ vars-of $s$ by $\operatorname{simp}$
from this show?thesis by simp
next
assume $L=\operatorname{Neg}(E q s t)$
then have vars-of-lit $L=$ vars-of $s \cup$ vars-of $t$ by $\operatorname{simp}$
from this show? ?thesis by simp
qed
next assume $p=$ pos
from this and $\operatorname{assms}(1)$ have $(L=\operatorname{Pos}(E q t s)) \vee(L=(\operatorname{Pos}(E q s t)))$

```
        unfolding orient-lit-def by auto
        then show ?thesis
        proof
        assume L = Pos (Eqt s)
        then have vars-of-lit L = vars-of t U vars-of s by simp
        from this show ?thesis by simp
    next
        assume L = Pos (Eq s t)
        then have vars-of-lit L = vars-of s U vars-of t by simp
        from this show ?thesis by simp
    qed
    qed
qed
lemma orient-lit-inst-vars:
    assumes orient-lit-inst L tsp\sigma
    shows vars-of t\subseteqvars-of-lit L ^vars-of s\subseteq vars-of-lit L
proof -
    have }p=neg\veep=pos using sign.exhaust by aut
    then show ?thesis
    proof
        assume p = neg
        from this and assms(1) have (L=Neg (Eq t s))\vee (L=(Neg (Eq st)))
            unfolding orient-lit-inst-def by auto
        then show ?thesis
        proof
            assume L = Neg(Eqts)
            then have vars-of-lit L = vars-of t U vars-of s by simp
            from this show ?thesis by simp
        next
            assume L =Neg(Eqst)
            then have vars-of-lit L = vars-of s U vars-of t by simp
            from this show ?thesis by simp
        qed
        next assume p = pos
        from this and assms(1) have (L=Pos (Eq t s))\vee (L=(Pos (Eqst)))
            unfolding orient-lit-inst-def by auto
        then show ?thesis
        proof
            assume L = Pos (Eqts)
            then have vars-of-lit L = vars-of t U vars-of s by simp
            from this show?thesis by simp
        next
            assume L = Pos (Eq s t)
            then have vars-of-lit L= vars-of s U vars-of t by simp
            from this show ?thesis by simp
        qed
    qed
qed
```

```
lemma orient-lit-inst-coincide:
    assumes orient-lit-inst L1 t s polarity \sigma
    assumes coincide-on \sigma \eta (vars-of-lit L1)
    shows orient-lit-inst L1 t s polarity \eta
proof -
    have polarity = pos \vee polarity = neg using sign.exhaust by auto
    then show ?thesis
    proof
    assume polarity = pos
    from this and assms(1) have L1 = Pos (Eqts)\veeL1 = Pos (Eqst)
        and ((subst t \sigma), (subst s \sigma)) # trm-ord
        unfolding orient-lit-inst-def by auto
    from}\langleL1=\operatorname{Pos}(Eqts)\veeL1=\operatorname{Pos}(Eqst)
        have vars-of t\subseteqvars-of-lit L1 and vars-of s\subseteqvars-of-lit L1 by auto
    from «vars-of t\subseteqvars-of-lit L1> and assms(2) have coincide-on \sigma \eta (vars-of
t)
            unfolding coincide-on-def by auto
    from «vars-of s \subseteqvars-of-lit L1〉 and assms(2) have coincide-on \sigma \eta (vars-of
s)
        unfolding coincide-on-def by auto
    from «( (subst t \sigma), (subst s \sigma)) \not\intrm-ord>
        and <coincide-on \sigma \eta (vars-of t)\rangle and <coincide-on \sigma \eta (vars-of s)\rangle assms(2)
    have ( (subst t \eta), (subst s \eta)) & trm-ord
        using coincide-on-term by metis
    from this and <polarity = pos\rangle and }\langleL1=Pos(Eqts)\veeL1=Pos (Eqst)
show ?thesis
        unfolding orient-lit-inst-def by auto
    next assume polarity = neg
    from this and assms(1) have L1 = Neg (Eqts)\veeL1 = Neg (Eqst)
        and ((subst t \sigma), (subst s \sigma)) # trm-ord
        unfolding orient-lit-inst-def by auto
    from}\langleL1=Neg(Eqts)\veeL1=Neg(Eqst)
        have vars-of t\subseteqvars-of-lit L1 and vars-of s\subseteqvars-of-lit L1 by auto
    from <vars-of t\subseteqvars-of-lit L1> and assms(2) have coincide-on \sigma \eta (vars-of
t)
        unfolding coincide-on-def by auto
    from <vars-of s\subseteqvars-of-lit L1〉 and assms(2) have coincide-on \sigma }\eta\mathrm{ (vars-of
s)
        unfolding coincide-on-def by auto
    from <( (subst t \sigma), (subst s \sigma)) \not\intrm-ord>
        and <coincide-on \sigma \eta (vars-of t)\rangle and <coincide-on \sigma \eta (vars-of s)\rangle assms(2)
    have ( (subst t \eta), (subst s \eta)) # trm-ord
        using coincide-on-term by metis
    from this and <polarity = neg> and <L1 =Neg (Eq t s)\veeL1 =Neg (Eq st)\rangle
show ?thesis
        unfolding orient-lit-inst-def by auto
```

```
    qed
qed
lemma orient-lit-inst-subterms:
    assumes orient-lit-inst L t s polarity \sigma
    assumes }u\in\mathrm{ subterms-of t
    shows u\in subterms-of-lit L
proof -
    have polarity = pos \vee polarity = neg using sign.exhaust by auto
    then show ?thesis
    proof
        assume polarity = pos
        from this and assms(1) have L=(Pos (Eqts))\veeL=(Pos (Eqst))
            unfolding orient-lit-inst-def by auto
        then show ?thesis
        proof
            assume L =(Pos (Eqts))
            from this and assms(2) show ?thesis by simp
            next assume L = (Pos (Eqst))
            from this and assms(2) show ?thesis by simp
        qed
    next
        assume polarity = neg
        from this and assms(1) have L=(Neg (Eq t s))\veeL=(Neg (Eq st))
            unfolding orient-lit-inst-def by auto
        then show ?thesis
        proof
            assume L = (Neg (Eqt s))
            from this and assms(2) show ?thesis by simp
            next assume L = (Neg (Eqst))
            from this and assms(2) show ?thesis by simp
        qed
    qed
qed
```


### 4.3 Clause Ordering

Clauses and extended clauses are ordered by transforming them into multisets of multisets of terms. To avoid any problem with the merging of identical literals, the multiset is assigned to a pair clause-substitution rather than to an instantiated clause.

We first map every literal to a multiset of terms, using the usual conventions and then define the multisets associated with clauses and extended clauses.

```
fun mset-lit :: 'a literal => 'a trm multiset
    where mset-lit (Pos (Eq| s)) ={#t,s#}|
        mset-lit (Neg (Eq t s))={#t,t,s,s#}
```

fun mset-cl
where $\operatorname{mset}-\mathrm{cl}(C, \sigma)=\{\#($ mset-lit $($ subst-lit $x \sigma)) . x \in \#($ mset-set $C) \#\}$
fun mset-ecl
where mset-ecl $(C, \sigma)=\{\#($ mset-lit (subst-lit $x \sigma)) . x \in \#$ (mset-set (cl-ecl C)) \#\}

```
lemma mset-ecl-conv: mset-ecl \((C, \sigma)=m s e t-c l(c l-e c l ~ C, \sigma)\)
    by \(\operatorname{simp}\)
lemma mset-ecl-and-mset-lit:
    assumes \(L \in(\) cl-ecl \(C)\)
    assumes finite (cl-ecl C)
    shows (mset-lit (subst-lit \(L \sigma)) \in \#(\) mset-ecl \((C, \sigma))\)
proof -
    from \(\operatorname{assms}(1) \operatorname{assms}(2)\) have \(L \in \#(\) mset-set (cl-ecl C)) by (simp)
    then show ?thesis
    proof -
    have f1: mset-set (cl-ecl C) \(-\{\# L \#\}+\{\# L \#\}=\) mset-set (cl-ecl C)
    by (meson \(\langle L \in \#\) mset-set (cl-ecl C) > insert-DiffM2)
    have count \(\{\#\) mset-lit (subst-lit L \(\sigma\) ) \#\} (mset-lit \((\) subst-lit \(L \sigma))=1\)
        by \(\operatorname{simp}\)
    then show? ?hesis
        by (metis (no-types, lifting) f1 image-mset-add-mset insert-iff mset-ecl.simps
set-mset-add-mset-insert union-mset-add-mset-right)
    qed
qed
lemma ecl-ord-coincide:
    assumes coincide-on \(\sigma \sigma^{\prime}(\) vars-of-cl \((\) cl-ecl \(C))\)
    shows mset-ecl \((C, \sigma)=\) mset-ecl \(\left(C, \sigma^{\prime}\right)\)
proof -
    have \(\forall x .\left(x \in(\right.\) cl-ecl \(C) \longrightarrow\left((\right.\) subst-lit \(x \sigma)=\left(\right.\) subst-lit \(\left.\left.\left.x \sigma^{\prime}\right)\right)\right)\)
    proof \(((\) rule allI \(),(\) rule impI \())\)
    fix \(x\) assume \(x \in(\) cl-ecl \(C)\)
    from this have vars-of-lit \(x \subseteq\) (vars-of-cl (cl-ecl \(C\) )) by auto
    from this and assms(1) have coincide-on \(\sigma \sigma^{\prime}\) (vars-of-lit \(\left.x\right)\) unfolding coin-
cide-on-def by auto
    from this show \(\left((\right.\) subst-lit \(x \sigma)=\left(\right.\) subst-lit \(\left.\left.x \sigma^{\prime}\right)\right)\)
            by (simp add: coincide-on-lit)
    qed
    then show ?thesis using equal-image-mset
    [of cl-ecl C \(\lambda\) x. (mset-lit (subst-lit \(x \sigma\) )) \(\lambda x\). (mset-lit (subst-lit \(\left.x \sigma^{\prime}\right)\) )]
        by (metis mset-ecl.simps)
qed
```

Literal and clause orderings are obtained as usual as the multiset extensions of the term ordering.

```
definition lit-ord :: ('a literal > 'a literal) set
    where
```

$$
\begin{aligned}
& \text { lit-ord }= \\
& \quad\{(x, y) \cdot(((\text { mset-lit } x),(\text { mset-lit } y)) \in(\text { mult trm-ord }))\}
\end{aligned}
$$

lemma mult-trm-ord-trans: shows trans (mult trm-ord)
by (metis (no-types, lifting) mult-def transI transitive-closure-trans(2))
lemma lit-ord-trans:
shows trans lit-ord
by (metis (no-types, lifting) basic-superposition.lit-ord-def basic-superposition-axioms case-prodD case-prodI mem-Collect-eq mult-def transI transitive-closure-trans(2))

## lemma lit-ord-wf: <br> shows wf lit-ord <br> proof -

from trm-ord-wf have wf (mult trm-ord) using wf-mult by auto
then show ?thesis
using lit-ord-def
and measure-wf [of (mult trm-ord) lit-ord mset-lit]
by blast
qed
definition ecl-ord $::\left(\left({ }^{\prime} a\right.\right.$ eclause $\times$ 'a subst $) \times\left({ }^{\prime} a\right.$ eclause $\times$ 'a subst $\left.)\right)$ set where
ecl-ord $=$

$$
\{(x, y) \cdot(((\text { mset-ecl } x),(\text { mset-ecl } y)) \in(\text { mult }(\text { mult trm-ord })))\}
$$

definition ecl-ord-eq :: (('a eclause $\times$ 'a subst $) \times\left({ }^{\prime} a\right.$ eclause $\times$ 'a subst $\left.)\right)$ set where
ecl-ord-eq $=$
ecl-ord $\cup\{(x, y) .(($ mset-ecl $x)=($ mset-ecl $y))\}$
definition cl-ord $::\left(\left({ }^{\prime} a\right.\right.$ clause $\times$ 'a subst $) \times($ 'a clause $\times$ 'a subst $\left.)\right)$ set where
cl-ord $=$
$\{(x, y) .((($ mset-cl $x),($ mset-cl $y)) \in($ mult $($ mult trm-ord $)))\}$
definition cl-ord-eq :: (('a clause $\times$ 'a subst $) \times\left({ }^{\prime} a\right.$ clause $\times$ 'a subst $\left.)\right)$ set where
cl-ord-eq $=$ cl-ord $\cup$
$\{(x, y) \cdot($ mset-cl $x)=($ mset-cl $y)\}$
lemma member-ecl-ord-iff:
$\left(\left(C, \sigma_{C}\right),\left(D, \sigma_{D}\right)\right) \in$ ecl-ord $\longleftrightarrow\left(\left(\right.\right.$ cl-ecl $\left.C, \sigma_{C}\right),\left(\right.$ cl-ecl $\left.\left.D, \sigma_{D}\right)\right) \in$ cl-ord by (simp add: ecl-ord-def cl-ord-def)
lemma mult-mult-trm-ord-trans:
shows trans (mult (mult trm-ord))
by (metis (no-types, lifting) mult-def transI transitive-closure-trans(2))
lemma ecl-ord-trans:
shows trans ecl-ord
by (metis (no-types, lifting) basic-superposition.ecl-ord-def basic-superposition-axioms case-prodD
case-prodI mem-Collect-eq mult-def transI transitive-closure-trans(2))
lemma cl-ord-trans:
shows trans cl-ord
by (metis (no-types, lifting) basic-superposition.cl-ord-def basic-superposition-axioms case-prodD
case-prodI mem-Collect-eq mult-def transI transitive-closure-trans(2))
lemma cl-ord-eq-trans:
shows trans cl-ord-eq
proof -
have $\forall r$. trans $r=\left(\forall p\right.$ pa pb. $\left(\left(p::^{\prime} a\right.\right.$ literal set $\times\left({ }^{\prime} a \times{ }^{\prime} a\right.$ trm $)$ list, $\left.p a\right) \notin r \vee$ $(p a, p b) \notin r)$
$\vee(p, p b) \in r)$
by (simp add: trans-def)
then obtain $p p::\left(\left({ }^{\prime} a\right.\right.$ literal set $\times\left({ }^{\prime} a \times{ }^{\prime} a\right.$ trm $)$ list $) \times{ }^{\prime} a$ literal set $\times\left({ }^{\prime} a \times{ }^{\prime} a\right.$ trm) list) set $\Rightarrow{ }^{\prime}$ a literal set $\times\left({ }^{\prime} a \times\right.$ ' $a$ trm $)$ list and ppa $::\left(\left({ }^{\prime} a\right.\right.$ literal set $\times\left({ }^{\prime} a\right.$ $\times{ }^{\prime} a$ trm $)$ list $) \times{ }^{\prime} a$ literal set $\times\left({ }^{\prime} a \times{ }^{\prime} a\right.$ trm $)$ list $)$ set $\Rightarrow{ }^{\prime} a$ literal set $\times\left({ }^{\prime} a \times{ }^{\prime} a\right.$ trm) list and ppb :: (('a literal set $\times\left({ }^{\prime} a \times{ }^{\prime} a\right.$ trm $)$ list $) \times{ }^{\prime} a$ literal set $\times\left({ }^{\prime} a \times{ }^{\prime} a\right.$ trm) list) set $\Rightarrow{ }^{\prime}$ a literal set $\times\left({ }^{\prime} a \times\right.$ 'a trm) list where
f1: $\forall r .(\neg$ trans $r \vee(\forall p$ pa $p b .(p, p a) \notin r \vee(p a, p b) \notin r \vee(p, p b) \in r)) \wedge$ (trans $r \vee(p p r, p p a r) \in r \wedge(p p a r, p p b r) \in r \wedge(p p r, p p b r) \notin r)$
by (metis (no-types))
have f2: trans $\{(p, p a) .($ mset-cl $p$, mset-cl pa) $\in$ mult (mult trm-ord $)\}$
using cl-ord-def cl-ord-trans by presburger
\{ assume $\neg$ (case (ppa (cl-ord $\cup\{(p, p a)$. mset-cl $p=m s e t-c l p a\})$, ppb (cl-ord $\cup\{(p, p a) . m s e t-c l p=m s e t-c l p a\}))$ of $(p, p a) \Rightarrow$ mset-cl $p=m s e t-c l p a)$
\{ assume (ppa (cl-ord $\cup\{(p a, p)$. mset-cl pa $=$ mset-cl $p\})$, ppb $(c l-o r d \cup$ $\{(p a, p)$. mset-cl $p a=$ mset-cl $p\})) \in\{(p a, p) .($ mset-cl pa, mset-cl $p) \in$ mult $($ mult trm-ord) $\}$

## moreover

\{ assume (ppa (cl-ord $\cup\{(p a, p)$. mset-cl pa $=$ mset-cl $p\})$, ppb (cl-ord $\cup$ $\{(p a, p)$. mset-cl pa $=$ mset-cl $p\})) \in\{(p a, p) .($ mset-cl pa, mset-cl $p) \in$ mult $($ mult trm-ord $)\} \wedge($ mset-cl $(p p(c l-o r d \cup\{(p a, p)$. mset-cl $p a=m s e t-c l p\}))$, mset-cl $(p p b(c l-o r d \cup\{(p a, p)$. mset-cl $p a=$ mset-cl $p\}))) \notin$ mult (mult trm-ord $)$
then have $(p p a(c l-o r d \cup\{(p a, p) . m s e t-c l ~ p a=m s e t-c l p\})$, ppb $(c l-o r d \cup$ $\{(p a, p)$. mset-cl $p a=$ mset-cl $p\})) \in\{(p a, p) .($ mset-cl pa, mset-cl $p) \in$ mult $($ mult trm-ord $)\} \wedge \operatorname{mset}-c l(p p($ cl-ord $\cup\{(p a, p)$. mset-cl $p a=m s e t-c l p\})) \neq$ mset-cl $(p p a($ cl-ord $\cup\{(p a, p)$. mset-cl $p a=$ mset-cl $p\}))$

## by force

then have $((p p)(c l-o r d \cup\{(p a, p)$. mset-cl $p a=m s e t-c l p\})$, ppb $(c l-o r d \cup$ $\{(p a, p)$. mset-cl $p a=$ mset-cl $p\})) \in\{(p a, p) .($ mset-cl pa, mset-cl $p) \in$ mult (mult trm-ord $)\} \vee(p p(c l-o r d \cup\{(p a, p)$. mset-cl $p a=m s e t-c l p\})$, ppb $(c l-o r d \cup\{(p a$,
p). mset-cl $p a=m s e t-c l p\})) \in\{(p a, p)$. mset-cl $p a=$ mset-cl $p\}) \vee(p p(c l-o r d$ $\cup\{(p a, p)$. mset-cl pa $=$ mset-cl $p\})$, ppa $($ cl-ord $\cup\{(p a, p)$. mset-cl $p a=$ mset-cl $p\})) \notin\{(p a, p) .($ mset-cl pa, mset-cl $p) \in$ mult (mult trm-ord $)\} \wedge(p p$ (cl-ord $\cup$ $\{(p a, p)$. mset-cl $p a=$ mset-cl $p\}), p p a(c l-o r d \cup\{(p a, p)$. mset-cl $p a=$ mset-cl $p\})) \notin\{(p a, p)$. mset-cl $p a=$ mset-cl $p\}$
using f2 f1 by blast \}
ultimately have (mset-cl $(p p(c l-o r d \cup\{(p a, p)$. mset-cl $p a=m s e t-c l p\}))$, mset-cl $(p p b($ cl-ord $\cup\{(p a, p)$. mset-cl pa $=$ mset-cl $p\}))) \in$ mult $($ mult trm-ord $)$ $\vee((p p(c l-o r d \cup\{(p a, p)$. mset-cl $p a=m s e t-c l p\}), p p b(c l-o r d \cup\{(p a, p) . m s e t-c l$ $p a=$ mset-cl $p\})) \in\{(p a, p) .($ mset-cl pa, mset-cl $p) \in$ mult (mult trm-ord $)\} \vee(p p$ $(c l-o r d \cup\{(p a, p) . m s e t-c l p a=m s e t-c l p\})$, ppb $(c l-o r d \cup\{(p a, p)$. mset-cl pa $=$ mset-cl $p\})) \in\{(p a, p)$. mset-cl $p a=m s e t-c l p\}) \vee(p p(c l-o r d \cup\{(p a, p) . m s e t-c l$ $p a=m s e t-c l p\}), p p a(c l-o r d \cup\{(p a, p)$. mset-cl $p a=$ mset-cl $p\})) \notin\{(p a, p)$. (mset-cl pa, mset-cl $p) \in$ mult (mult trm-ord $)\} \wedge(p p(c l-o r d \cup\{(p a, p)$. mset-cl pa $=$ mset-cl $p\})$, ppa $(c l-o r d \cup\{(p a, p)$. mset-cl $p a=$ mset-cl $p\})) \notin\{(p a, p)$. mset-cl $p a=m s e t-c l p\}$

## by fastforce $\}$

then have ( $m s e t-c l(p p(c l-o r d \cup\{(p a, p)$. mset-cl $p a=m s e t-c l p\}))$, mset-cl $(p p b(c l-o r d \cup\{(p a, p)$. mset-cl pa $=$ mset-cl $p\}))) \in$ mult $($ mult trm-ord $) \vee((p p$ $(c l-o r d \cup\{(p a, p) . m s e t-c l p a=m s e t-c l p\})$, ppb $(c l-o r d \cup\{(p a, p) . m s e t-c l p a$ $=$ mset-cl $p\})) \in\{(p a, p) .($ mset-cl pa, mset-cl $p) \in$ mult (mult trm-ord $)\} \vee(p p$ $(c l-o r d \cup\{(p a, p) . m s e t-c l p a=m s e t-c l p\}), p p b(c l-o r d \cup\{(p a, p)$. mset-cl $p a=$ $m s e t-c l p\})) \in\{(p a, p)$. mset-cl $p a=m s e t-c l p\}) \vee(p p(c l-o r d \cup\{(p a, p) . m s e t-c l$ $p a=$ mset-cl $p\}), p p a(c l-o r d \cup\{(p a, p)$. mset-cl $p a=$ mset-cl $p\})) \notin\{(p a, p)$. (mset-cl pa, mset-cl $p) \in$ mult (mult trm-ord $)\} \wedge(p p(c l-o r d \cup\{(p a, p)$. mset-cl pa $=m s e t-c l p\}), p p a(c l-o r d \cup\{(p a, p)$. mset-cl $p a=$ mset-cl $p\})) \notin\{(p a, p)$. mset-cl $p a=m s e t-c l p\} \vee(p p(c l-o r d \cup\{(p a, p) . m s e t-c l p a=m s e t-c l p\})$, ppa $(c l-o r d \cup$ $\{(p a, p)$. mset-cl $p a=$ mset-cl $p\})) \notin$ cl-ord $\cup\{(p a, p)$. mset-cl pa $=$ mset-cl $p\} \vee$ $(p p a(c l-o r d \cup\{(p a, p) . m s e t-c l p a=m s e t-c l p\}), p p b(c l-o r d \cup\{(p a, p) . m s e t-c l p a$ $=$ mset-cl $p\})) \notin$ cl-ord $\cup\{(p a, p)$. mset-cl pa= mset-cl $p\} \vee(p p(c l-o r d \cup\{(p a$, $p)$. mset-cl $p a=$ mset-cl $p\}), p p b(c l-o r d \cup\{(p a, p)$. mset-cl $p a=m s e t-c l p\})) \in$ cl-ord $\cup\{(p a, p)$. mset-cl pa $=$ mset-cl $p\}$
using cl-ord-def by auto \}
then have $(p p(c l-o r d \cup\{(p a, p)$. mset-cl $p a=$ mset-cl $p\})$, ppa $(c l$-ord $\cup\{(p a$, $p)$. mset-cl $p a=m s e t-c l p\})) \notin$ cl-ord $\cup\{(p a, p)$. mset-cl pa=mset-cl $p\} \vee(p p a$ $(c l-o r d \cup\{(p a, p)$. mset-cl $p a=m s e t-c l p\})$, ppb $(c l-o r d \cup\{(p a, p)$. mset-cl $p a=$ mset-cl $p\})) \notin$ cl-ord $\cup\{(p a, p)$. mset-cl pa $=$ mset-cl $p\} \vee(p p(c l-o r d \cup\{(p a, p)$. $m s e t-c l p a=m s e t-c l p\}), p p b(c l-o r d \cup\{(p a, p)$. mset-cl pa=mset-cl $p\})) \in$ cl-ord $\cup\{(p a, p)$. mset-cl pa $=$ mset-cl $p\}$
using cl-ord-def by force
then have trans (cl-ord $\cup\{(p a, p)$. mset-cl pa=mset-cl $p\})$
using $f 1$ by meson
from this show ?thesis unfolding cl-ord-eq-def by auto
qed
lemma wf-ecl-ord:
shows wf ecl-ord
proof -
have $w f$ (mult trm-ord) using trm-ord-wf and wf-mult by auto

```
then have wf (mult (mult trm-ord)) using wf-mult by auto
thus?thesis
    using ecl-ord-def
    and measure-wf [of (mult (mult trm-ord)) ecl-ord mset-ecl]
    by blast
qed
```

definition maximal-literal $::$ 'a literal $\Rightarrow$ 'a clause $\Rightarrow$ bool
where
$($ maximal-literal $L C)=(\forall x .(x \in C \longrightarrow(L, x) \notin$ lit-ord $))$
definition eligible-literal
where
$($ eligible-literal $L C \sigma)=(L \in$ sel $($ cl-ecl $C) \vee$
$($ sel $($ cl-ecl $C)=\{ \}$
$\wedge($ maximal-literal $($ subst-lit L $\sigma)($ subst-cl $(c l-e c l ~ C) \sigma)))$
definition strictly-maximal-literal
where strictly-maximal-literal $C L \sigma=$
$(\forall x \in(c l-e c l C)-\{L\} .(($ subst-lit $x \sigma),($ subst-lit $L \sigma))$
$\in$ lit-ord)
definition lower-or-eq
where lower-or-eq $t s=((t=s) \vee((t, s) \in$ trm-ord $))$
lemma eligible-literal-coincide:
assumes coincide-on $\sigma \sigma^{\prime}($ vars-of-cl $(c l-e c l ~ C))$
assumes eligible-literal LC $\sigma$
assumes $L \in($ cl-ecl $C)$
shows eligible-literal $L C \sigma^{\prime}$
proof -
from assms(2) have
$L \in$ sel $($ cl-ecl $C) \vee($ sel $($ cl-ecl $C)=\{ \} \wedge$ maximal-literal $($ subst-lit $L \sigma)$
(subst-cl (cl-ecl C) $\sigma$ ))
unfolding eligible-literal-def by auto
then show ?thesis
proof
assume $L \in \operatorname{sel}(c l-e c l C)$
then show? ?thesis unfolding eligible-literal-def by auto
next
assume sel $($ cl-ecl $C)=\{ \} \wedge$ maximal-literal (subst-lit L $\sigma$ ) (subst-cl (cl-ecl
C) $\sigma$ )
then have sel (cl-ecl C) $=\{ \}$ and maximal-literal (subst-lit $L \sigma$ ) (subst-cl
(cl-ecl C) $\sigma$ )
by auto
from $\operatorname{assms}(1)$ have (subst-cl (cl-ecl C) $\sigma)=\left(\right.$ subst-cl (cl-ecl C) $\left.\sigma^{\prime}\right)$
using coincide-on-cl by blast
from $\operatorname{assms}(3)$ and $\operatorname{assms}(1)$ have coincide-on $\sigma \sigma^{\prime}$ (vars-of-lit L) unfolding
coincide-on-def

```
        by auto
    from this have (subst-lit L \sigma)}=(\mathrm{ subst-lit L }\mp@subsup{\sigma}{}{\prime}
            using coincide-on-lit by auto
    from this and <(subst-cl (cl-ecl C) \sigma) = (subst-cl (cl-ecl C) \sigma')>
        and <maximal-literal (subst-lit L \sigma) (subst-cl (cl-ecl C) \sigma)>
        have maximal-literal (subst-lit L \sigma') (subst-cl (cl-ecl C) 䭫)
        by auto
    from this and «sel (cl-ecl C) = {}` show ?thesis unfolding eligible-literal-def
by auto
    qed
qed
```

The next definition extends the ordering to substitutions．

```
definition lower-on
where lower-on \sigma \eta V = ( }\forallx\inV\mathrm{ .
    (lower-or-eq (subst (Var x) \sigma) ((subst (Var x ) \eta))))
```

We now establish some properties of the ordering relations．

```
lemma lower-or-eq-monotonic:
    assumes lower-or-eq t1 s1
    assumes lower-or-eq t2 s2
    shows lower-or-eq (Comb t1 t2) (Comb s1 s2)
unfolding lower-or-eq-def using trm-ord-reduction-left trm-ord-reduction-right
    by (metis assms(1) assms(2) lower-or-eq-def trm-ord-trans transD)
lemma lower-on-term:
    shows \(\wedge \sigma \eta\). lower-on \(\sigma \eta(\) vars-of \(t) \Longrightarrow\)
        (lower-or-eq (subst \(t \sigma\) ) (subst \(t \eta)\) )
proof (induction \(t\) )
    case (Var \(x\) )
        from this show ?case
            unfolding lower-on-def by auto
    next case (Const \(x\) )
        show ?case
            unfolding lower-or-eq-def by auto
    next case (Comb t1 t2)
        show \(\wedge \sigma \eta\). lower-on \(\sigma \eta(\) vars-of \((\) Comb t1 t2) \() \Longrightarrow\)
        (lower-or-eq (subst (Comb t1 t2) \(\sigma\) ) (subst (Comb t1 t2) \(\eta\) ))
    proof -
            fix \(\sigma \eta\) assume lower-on \(\sigma \eta\) (vars-of (Comb t1 t2))
            from this have lower-on \(\sigma \eta\) (vars-of t1) and lower-on \(\sigma \eta\) (vars-of t2)
                unfolding lower-on-def by auto
            from 〈lower-on \(\sigma \eta\) (vars-of t1) 〉 have lower-or-eq (subst t1 \(\sigma\) ) (subst t1 \(\eta\) )
                using Comb.IH by auto
            from 〈lower-on \(\sigma \eta\) (vars-of t2)〉 have lower-or-eq (subst t2 \(\sigma\) ) (subst t2 \(\eta\) )
                using Comb.IH by auto
            from 〈lower-or-eq (subst t1 \(\sigma\) ) (subst t1 \(\eta\) )〉〈lower-or-eq (subst t2 \(\sigma\) ) (subst
t2 \(\eta\) ) >
            show lower-or-eq (subst (Comb t1 t2) \(\sigma\) ) (subst (Comb t1 t2) \(\eta\) )
```

using lower-or-eq-monotonic by auto
qed
qed
lemma diff-substs-yield-diff-trms:
assumes $($ subst $(\operatorname{Var} x) \sigma) \neq($ subst $(\operatorname{Var} x) \eta)$
shows ( $x \in$ vars-of $t$ )
$\Longrightarrow($ subst $t \sigma) \neq($ subst $t \eta)$
proof (induction $t$ )
case (Var y)
show $(x \in \operatorname{vars}-$ of $(\operatorname{Var} y)) \Longrightarrow($ subst $(\operatorname{Var} y) \sigma) \neq($ subst $(\operatorname{Var} y) \eta)$
proof -
assume ( $x \in$ vars-of (Var $y)$ )
from $\langle(x \in \operatorname{vars}-o f(\operatorname{Var} y))\rangle$ have $x=y$ by auto
from this and $\operatorname{assms}(1)$
show (subst (Var y) $\sigma) \neq($ subst $(\operatorname{Var} y) \eta)$
by auto
qed
next case (Const $y$ )
show ( $x \in$ vars-of (Const $y$ ))
$\Longrightarrow($ subst (Const $y) \sigma) \neq($ subst (Const y) $\eta)$
proof (rule ccontr)
from 〈( $x \in$ vars-of (Const $y)$ ) show False by auto
qed
next case (Comb t1 t2)
show ( $x \in$ vars-of (Comb t1 t2))
$\Longrightarrow($ subst (Comb t1 t2) $\sigma) \neq($ subst (Comb t1 t2) $\eta)$
proof -
assume ( $x \in$ vars-of (Comb t1 t2))
from $\langle x \in$ vars-of (Comb t1 t2) $\rangle$ have $x \in$ vars-of $t 1 \vee x \in$ vars-of t2 by
auto
then show (subst (Comb t1 t2) $\sigma) \neq($ subst $(\operatorname{Comb} t 1$ t2) $\eta)$
proof
assume $x \in$ vars-of t1
from this have (subst t1 $\sigma$ ) $\neq($ subst t1 $\eta)$
using Comb.IH by auto
then show ?thesis by auto
next
assume $x \in$ vars-of t2
from this have (subst t2 $\sigma$ ) $\neq($ subst t2 $\eta)$
using Comb.IH by auto
then show ?thesis by auto
qed
qed
qed
lemma lower-subst-yields-lower-trms:
assumes lower-on $\sigma \eta$ (vars-of $t$ )
assumes $(($ subst $(\operatorname{Var} x) \sigma),($ subst $(\operatorname{Var} x) \eta)) \in$ trm-ord

```
    assumes (x \in vars-of t)
    shows}((\mathrm{ subst t }\sigma),(\mathrm{ subst t }\eta))\in\mathrm{ trm-ord
proof -
    from assms(1) have lower-or-eq (subst t \sigma) (subst t \eta)
        using lower-on-term by auto
    from assms(2) have (subst (Var x) \sigma) }=(\mathrm{ (subst (Var x) }\eta\mathrm{ )
        using trm-ord-irrefl irrefl-def by fastforce
    from this and assms(3) have (subst t \sigma)\not=(subst t \eta)
        using diff-substs-yield-diff-trms by fastforce
    from this and «lower-or-eq (subst t \sigma) (subst t \eta)>
        show ?thesis unfolding lower-or-eq-def by auto
qed
lemma lower-on-lit:
    assumes lower-on \sigma \eta (vars-of-lit L)
    assumes ((subst (Var x) \sigma),(subst (Var x) \eta)) \in trm-ord
    assumes x vars-of-lit L
    shows ((subst-lit L \sigma), (subst-lit L \eta)) \in lit-ord
proof -
    obtain ts where def-l: L = Pos (Eqt s)| L = (Neg (Eq t s))
        by (metis mset-lit.cases)
    from this have vars-of t\subseteqvars-of-lit L and vars-of s\subseteqvars-of-lit L by auto
    from <vars-of s\subseteqvars-of-lit L> and assms(1) have lower-on \sigma }\eta\mathrm{ (vars-of s)
        unfolding lower-on-def by auto
    from def-l have def-ms-l: mset-lit L}={#t,s#}\vee mset-lit L ={#t,t,s,s#
by auto
    from this have t\in# (mset-lit L) and s\in# (mset-lit L) by auto
    from def-l have mset-lit (subst-lit L \sigma)={# (subst u \sigma). u \in# (mset-lit L)
#} by auto
    from def-l have mset-lit (subst-lit L \eta)={# (subst u \eta). u\in# (mset-lit L)#}
by auto
    from〈lower-on \sigma \eta (vars-of s)> have lower-or-eq (subst s \sigma) (subst s \eta)
        using lower-on-term by auto
    let ?L = mset-lit L
    let ?M1 = mset-lit (subst-lit L \sigma)
    let ?M2 = mset-lit (subst-lit L \eta)
    from 〈vars-of t\subseteqvars-of-lit L〉 and assms(1) have lower-on \sigma }\eta\mathrm{ (vars-of t)
        unfolding lower-on-def by auto
    from <vars-of s\subseteqvars-of-lit L> and assms(1) have lower-on \sigma }\eta\mathrm{ (vars-of s)
        unfolding lower-on-def by auto
    have all-lower: }\forallu.(u\in#(\mathrm{ mset-lit L) }\longrightarrow(((subst u \sigma),(subst u \eta)) \in trm-ord
    \vee ( \text { subst u } \sigma ) = ( \text { subst u ף))} )
    proof (rule allI,rule impI)
        fix }u\mathrm{ assume }u\in#(mset-lit L)
    have }u=t\veeu=
    proof (cases)
            assume mset-lit L = {#t,s#}
            from this and }\langleu\in# (mset-lit L)\rangle show ?thesi
```

```
            by (simp add: count-single insert-DiffM2 insert-noteq-member not-gr0)
    next
    assume }\neg\mathrm{ mset-lit L ={#t,s#}
    from this and def-ms-l have mset-lit L = {#t,t,s,s #}
        by auto
    from this and }\langleu\in# (mset-lit L)\rangle show ?thesi
        using not-gr0 by fastforce
    qed
    then show (((subst u\sigma), (subst u \eta)) \in trm-ord
\vee ( \text { subst u } ) = ( \text { subst u } \eta ) )
    proof
        assume u=t
        from〈lower-on \sigma \eta (vars-of t)> have lower-or-eq (subst t \sigma) (subst t \eta)
            using lower-on-term by auto
        from this show ?thesis unfolding lower-or-eq-def using < }u=t\rangle\mathrm{ by auto
    next
        assume u=s
        from〈lower-on \sigma \eta (vars-of s)> have lower-or-eq (subst s \sigma) (subst s \eta)
        using lower-on-term by auto
    from this show ?thesis unfolding lower-or-eq-def using <u = s` by auto
    qed
qed
have sl-exists: \existsu. (u \in# (mset-lit L) ^((subst u \sigma), (subst u \eta)) \in trm-ord)
proof -
    from <x < vars-of-lit L> and def-l have
        x\in vars-of t\vee }\vee\in\mathrm{ vars-of s by auto
    then show ?thesis
    proof
        assume x vars-of t
        from this and <lower-on \sigma \eta (vars-of t)>assms(1) assms(2)
            have ((subst t \sigma),(subst t \eta)) \in trm-ord
            using lower-subst-yields-lower-trms by auto
        from this and <t\in# (mset-lit L)\rangle show ?thesis by auto
    next
        assume }x\in\mathrm{ vars-of s
        from this and <lower-on \sigma \eta (vars-of s)>assms(1) assms(2)
            have ((subst s \sigma),(subst s \eta)) \in trm-ord
            using lower-subst-yields-lower-trms by auto
        from this and «s\in# (mset-lit L)\rangle show ?thesis by auto
    qed
qed
from all-lower sl-exists and
<mset-lit (subst-lit L \sigma) = {# (subst u \sigma).u\in# (mset-lit L) #}>
<mset-lit (subst-lit L \eta) = {# (subst u \eta). u\in# (mset-lit L) #}>
have (?M1,?M2) \in (mult trm-ord)
using trm-ord-irrefl image-mset-ordering
                                    [of ?M1 \lambdax. (subst x \sigma) ?L ?M2 \lambdax. (subst x \eta) trm-ord]
by blast
from this show ?thesis unfolding lit-ord-def by auto
```


## qed

lemma lower-on-lit-eq:
assumes lower-on $\sigma \eta$ (vars-of-lit $L$ )
shows $(($ subst-lit $L \sigma)=($ subst-lit $L \eta)) \vee(($ subst-lit L $\sigma),($ subst-lit L $\eta)) \in$ lit-ord
proof (cases)
assume coincide-on $\sigma \eta$ (vars-of-lit L)
then show ?thesis using coincide-on-lit by auto
next
assume $\neg$ coincide-on $\sigma \eta$ (vars-of-lit $L)$
then obtain $x$ where $x \in$ vars-of-lit $L$
and (subst $($ Var $x) \sigma) \neq($ subst $(\operatorname{Var} x) \eta)$
unfolding coincide-on-def by auto
from $\langle x \in$ vars-of-lit $L\rangle \operatorname{assms}(1)$
$\langle($ subst $(\operatorname{Var} x) \sigma) \neq($ subst $(\operatorname{Var} x) \eta)$ )and $\operatorname{assms}(1)$
have $(($ subst $(\operatorname{Var} x) \sigma),($ subst $(\operatorname{Var} x) \eta)) \in$ trm-ord
unfolding lower-on-def lower-or-eq-def by auto
from this assms(1) $\langle x \in$ vars-of-lit $L\rangle$ show ?thesis using lower-on-lit by auto qed
lemma lower-on-cl:
assumes lower-on $\sigma \eta($ vars-of-cl $(c l-e c l C))$
assumes $((\operatorname{subst}(\operatorname{Var} x) \sigma),(\operatorname{subst}(\operatorname{Var} x) \eta)) \in$ trm-ord
assumes $x \in$ vars-of-cl (cl-ecl C)
assumes finite (cl-ecl C)
shows $((C, \sigma),(C, \eta)) \in$ ecl-ord
proof -
let ?M1 $=$ mset-ecl $(C, \sigma)$
let ?M2 $=$ mset-ecl $(C, \eta)$
let $? M=($ mset-set $($ cl-ecl $C))$
let ? $\mathrm{f} 1=\lambda$. . mset-lit ( subst-lit $x \sigma$ ) $)$
let ?f2 $=\lambda x$. $($ mset-lit $($ subst-lit $x \eta))$
have ?M1 $=\{\#(? f 1 u) . u \in \# ? M \#\}$ using mset-ecl.simps by blast
have ? $\mathrm{M}_{2}=\{\#($ ?f2 $u) . u \in \#$ ?M \# $\}$ using mset-ecl.simps by blast
have $i: \forall u .(u \in \# ? M \longrightarrow(((? f 1 u),($ ?f2 $u)) \in($ mult trm-ord $) \vee(? f 1 u)=($ ? f2 u)))
proof $(($ rule allI $),($ rule impI $))$
fix $u$ assume $u \in \#$ ? $M$
from this have $u \in($ cl-ecl C) using count-mset-set(3) by (simp add: assms(4))
from this and assms(1) have lower-on $\sigma \eta$ (vars-of-lit u) unfolding lower-on-def by auto
then have $(($ subst-lit $u \sigma)=($ subst-lit $u \quad \eta))$
$\vee(($ subst-lit $u \sigma)$, (subst-lit u $\eta)) \in$ lit-ord
using lower-on-lit-eq by blast
from this show $(((? f 1 u),(? f 2 u)) \in($ mult trm-ord $) \vee(? f 1 u)=(? f 2 u))$
unfolding lit-ord-def by auto
qed
have irrefl (mult trm-ord) by (simp add: irreflI trm-ord-wf wf-mult)

```
    have \(i i: \exists u .(u \in \# ? M \wedge((? f 1 u),(? f 2 u)) \in(\) mult trm-ord \())\)
    proof -
    from \(\langle x \in\) vars-of-cl (cl-ecl \(C)\rangle\) obtain \(u\) where \(u \in(\) cl-ecl \(C)\) and \(x \in\)
vars-of-lit u
    by auto
    from \(\operatorname{assms}(4)\langle u \in(\) cl-ecl \(C)\rangle\) have \(u \in \#\) ?M by auto
    from \(\langle u \in(\) cl-ecl \(C)\rangle\) and assms(1) have lower-on \(\sigma \eta\) (vars-of-lit \(u\) )
        unfolding lower-on-def by auto
    from \(\langle x \in\) vars-of-lit \(u\rangle\langle l o w e r-o n ~ \sigma ~ \eta(v a r s-o f-l i t ~ u)\rangle \operatorname{assms}(2)\)
        have \(((\) subst-lit u \(\sigma)\), (subst-lit u \(\eta)) \in\) lit-ord
        using lower-on-lit by blast
    from this \(\langle u \in \#\) ? \(M>\) have \((u \in \#\) ? \(M \wedge((? f 1 u),(\) ?f2 \(u)) \in(\) mult trm-ord \())\)
        unfolding lit-ord-def by auto
    then show ?thesis by auto
    qed
    from \(i\) ii 〈?M1 \(=\{\#(? f 1 u) . u \in \# ? M \#\}\rangle\langle ? M 2=\{\#(? f 2 u) . u \in \# ? M\)
\#\}〉〈irrefl (mult trm-ord) 〉
    have \((? M 1, ? M 2) \in(\) mult ( mult trm-ord \())\)
    using image-mset-ordering [of ?M1 ?f1 ?M ?M2 ?f2 (mult trm-ord)] by auto
    then show ?thesis unfolding ecl-ord-def by auto
qed
lemma subterm-trm-ord :
    shows \(\bigwedge t s\).
                subterm t \(p s \Longrightarrow p \neq[]\)
        \(\Longrightarrow(s, t) \in\) trm-ord
proof (induction \(p\) )
    case Nil
        from 〈Nil \(\neq[]\rangle\) show ?case by auto
    next case (Cons i q)
        from \(\langle\) subterm \(t(i \# q) s\rangle\) obtain \(t 1\) t2 where
            \(t=\) (Comb t1 t2) using subterm.elims(2) by blast
    have \(i=\) Left \(\vee i=\) Right using indices.exhaust by blast
    then show \((s, t) \in\) trm-ord
    proof
            assume \(i=\) Left
            from this and \(\langle t=\) Comb \(t 1\) t2 \(\rangle\) and \(\langle\operatorname{subterm} t(i \# q) s\rangle\)
                have subterm t1 \(q\) s by auto
            show ?thesis
            proof (cases)
                assume \(q=\) Nil
                from this and \(\langle s u b t e r m ~ t 1 ~ q ~ s\rangle\) have \(t 1=s\) by auto
                from this and \(\langle t=\) Comb t1 t2〉 show ?case using trm-ord-subterm by
auto
            next
                assume \(q \neq\) Nil
                from this and \(\langle\) subterm t1 \(q s\rangle\) have \((s, t 1) \in\) trm-ord using Cons.IH by
auto
                from \(\langle t=\) Comb t1 t2 \(\rangle\) have \((t 1, t) \in\) trm-ord using trm-ord-subterm by
```

```
auto
            from this and }\langle(s,t1)\in\mathrm{ trm-ord }\rangle\mathrm{ show ?case
                using trm-ord-trans unfolding trans-def by metis
        qed
    next
        assume i= Right
        from this and <t=Comb t1 t2\rangle and <subterm t (i# q) s`
            have subterm t2 q s by auto
        show ?thesis
        proof (cases)
            assume q=Nil
            from this and <subterm t2 q s` have t2 =s by auto
            from this and 〈t=Comb t1 t2> show ?case using trm-ord-subterm by
auto
            next
            assume q}\not=\mathrm{ Nil
            from this and «subterm t2 q s` have (s,t2) \in trm-ord using Cons.IH by
auto
            from <t = Comb t1 t2> have ( t2,t) \in trm-ord using trm-ord-subterm by
auto
            from this and <(s,t2) \in trm-ord\rangle show ?case
                using trm-ord-trans unfolding trans-def by metis
            qed
    qed
qed
lemma subterm-trm-ord-eq :
    assumes subterm t p s
    shows s=t\vee(s,t)\intrm-ord
proof (cases)
    assume p=Nil
        from this and assms(1) show ?thesis by auto
    next assume p}\not=N\mathrm{ Nil
        from this and assms(1) show ?thesis using subterm-trm-ord by auto
qed
lemma subterms-of-trm-ord-eq :
    assumes s\in subterms-of t
    shows s=t\vee(s,t)\intrm-ord
proof -
    from assms(1) obtain p}\mathrm{ where subterm t p s using occurs-in-def by auto
    from this show ?thesis using subterm-trm-ord-eq by auto
qed
lemma subt-trm-ord:
    shows }t\precs\longrightarrow(t,s)\in\mathrm{ trm-ord
proof (induction s)
    case (Var x)
    show ?case
```

```
    proof
            assume t}\prec Var x
            then show (t,Var x)\in trm-ord by auto
    qed
    case (Const x)
    show ?case
    proof
        assume }t\prec\mathrm{ Const }
        then show (t,Const x) \in trm-ord by auto
    qed
    case (Comb t1 t2)
    show ?case
proof
    assume t}\prec\mathrm{ Comb t1 t2
    show (t, Comb t1 t2) \in trm-ord
    proof (rule ccontr)
            assume (t, Comb t1 t2) & trm-ord
            then have i: t\not= t1 using trm-ord-subterm by auto
            from}\langle(t,Comb t1 t2) &trm-ord\rangle have ii: t\not= t2 using trm-ord-subterm
by auto
            from i ii and <t < Comb t1 t2> have t \prec t1\vee t\prec t2 by auto
            from this and «(t, Comb t1 t2) & trm-ord 
                show False using Comb.IH trm-ord-subterm trm-ord-trans trans-def by
metis
        qed
    qed
qed
lemma trm-ord-vars:
    assumes (t,s)\intrm-ord
    shows vars-of t\subseteqvars-of s
proof (rule ccontr)
    assume \negvars-of t\subseteqvars-of s
    then obtain }x\mathrm{ where }x\in\mathrm{ vars-of t and }x\not\in\mathrm{ vars-of s by auto
    let ? }\sigma=[(x,s)
    from assms have ((subst t ?\sigma),(subst s ?\sigma)) \in trm-ord
        using trm-ord-subst by auto
    let ?\vartheta = []
    let ?V = vars-of s
    have subst s ?\vartheta = s by simp
    have subst (Var x) ? \sigma=s by simp
    have coincide-on ?\sigma ?\vartheta ?V
    proof (rule ccontr)
    assume \neg coincide-on ?\sigma ?\vartheta ?V
    then obtain }y\mathrm{ where }y\in?V\mathrm{ subst (Var y) ? }\sigma\not=\mathrm{ subst (Var y) ?v
        unfolding coincide-on-def by auto
    from «subst (Var y) ?\sigma \not= subst (Var y) ?\vartheta` have y = 人
        by (metis assoc.simps(2) subst.simps(1))
    from this and }\langlex\not\in\mathrm{ vars-of }s\rangle\langley\in?V\rangle\mathrm{ show False by auto
```


## qed

from this and «subst $s ? \vartheta=s$ 〉 have subst $s ? \sigma=s$
using coincide－on－term by metis
from $\langle x \in$ vars－of $t\rangle$ have $(\operatorname{Var} x) \prec t$ using 〈（subst $t[(x, s)]$ ，subst $s[(x, s)]) \in$ trm－ord〉 ＜subst $s[(x, s)]=s\rangle$ trm－ord－wf vars－iff－occseq by fastforce
from this have $((\operatorname{Var} x), t) \in$ trm－ord using subt－trm－ord by auto
from this and assms（1）have（Var $x, s) \in$ trm－ord using trm－ord－trans trans－def
by metis
from this have $(($ subst $(\operatorname{Var} x) ? \sigma),($ subst $s ? \sigma)) \in$ trm－ord using trm－ord－subst by metis
from this and «subst $s ? \sigma=s\rangle\langle$ subst $(\operatorname{Var} x) ? \sigma=s\rangle$ have $(s, s) \in$ trm－ord by auto
from this show False using trm－ord－irrefl irrefl－def by metis
qed
lemma lower－on－ground：
assumes lower－on $\sigma \eta V$
assumes ground－on $V \eta$
shows ground－on $V \sigma$
proof（rule ccontr）
assume $\neg$ ground－on $V \sigma$
from this obtain $x$ where $x \in V$ and vars－of（subst（Var $x) \sigma$ ）$\neq\{ \}$ unfolding ground－on－def ground－term－def by metis
from $\operatorname{assms}(1)\langle x \in V\rangle$ have $(\operatorname{subst}(\operatorname{Var} x) \sigma)=($ subst $(\operatorname{Var} x) \eta)$ $\vee(($ subst $(\operatorname{Var} x) \sigma),($ subst $(\operatorname{Var} x) \eta)) \in$ trm－ord unfolding lower－on－def lower－or－eq－def by metis
from this have vars－of（subst（Var $x) \sigma) \subseteq \operatorname{vars}$－of $(\operatorname{subst}(\operatorname{Var} x) \eta$ ） using trm－ord－vars by auto
from this and＜vars－of（subst（Var $x) \sigma) \neq\{ \}>$ have vars－of（subst（Var $x) \eta) \neq\{ \}$ by auto
from this and $\langle x \in V\rangle$ and assms（2）show False unfolding ground－on－def ground－term－def by metis
qed
lemma replacement－monotonic ：
shows $\Lambda t$ s．$(($ subst $v \sigma),($ subst $u \sigma)) \in$ trm－ord $\Longrightarrow$ subterm t $p u \Longrightarrow$ replace－subterm $t p v s$ $\Longrightarrow(($ subst s $\sigma),($ subst $t \sigma)) \in$ trm－ord
proof（induction $p$ ）
case Nil
from $\langle$ subterm $t$ Nil $u\rangle$ have $t=u$ by auto
from 〈replace－subterm $t$ Nil $v s\rangle$ have $s=v$ by auto
from $\langle t=u\rangle$ and $\langle s=v\rangle$ and $\langle(($ subst $v \sigma)$ ，（subst $u \sigma)) \in$ trm－ord $\rangle$
show ？case by auto
next case（Cons i q）
from＜subterm $t(i \# q) u\rangle$ obtain 11 t2 where $t=$（Comb t1 t2）using subterm．elims（2）by blast
have $i=$ Left $\vee i=$ Right using indices．exhaust by blast

```
        then show ((subst s \sigma),(subst t \sigma)) \in trm-ord
        proof
    assume i= Left
    from this and «t = Comb t1 t2\rangle and <subterm t (i# q) u`
        have subterm t1 qu by auto
    from <i = Left> and <t = Comb t1 t2\rangle and 〈replace-subterm t (i# q) v s\rangle
        obtain t1' where replace-subterm t1 qv t1' and s=Comb t1't2 by auto
    from <((subst v \sigma), (subst u \sigma)) \in trm-ord〉
        and «subterm t1 qu` and <replace-subterm t1 qvt1'〉 have
        ((subst t1'\sigma),(subst t1 \sigma)) \in trm-ord
        using Cons.IH Cons.prems(1) by blast
    from this and <t = (Comb t1 t2)\rangle and <s=(Comb t1' t2)>
        show ((subst s \sigma),(subst t \sigma)) \in trm-ord
        by (simp add: trm-ord-reduction-left)
    next
    assume i= Right
    from this and <t = Comb t1 t2\rangle and <subterm t (i# q) u`
        have subterm t2 q u by auto
    from <i = Right> and 〈t= Comb t1 t2> and <replace-subterm t (i#q) v s>
        obtain t2' where replace-subterm t2 qv t2' and s=Comb t1 t2' by auto
    from <((subst v \sigma), (subst u \sigma)) \in trm-ord〉
        and «subterm t2 qu` and <replace-subterm t2 qv t2'〉 have
        ((subst t2' \sigma), (subst t2 \sigma)) \in trm-ord
        using Cons.IH Cons.prems(2) by blast
    from this and <t =(Comb t1 t2)> and <s=(Comb t1 t2 ')>
        show ((subst s \sigma), (subst t \sigma)) \in trm-ord
        by (simp add: trm-ord-reduction-right)
    qed
qed
lemma mset-lit-subst:
    shows (mset-lit (subst-lit L \sigma)) =
        {# (subst x \sigma). x \in# (mset-lit L) #}
proof -
    have positive-literal L \vee negative-literal L
    using negative-literal.simps(2) positive-literal.elims(3) by blast
then show ?thesis
proof
    assume positive-literal L
    then obtain ts where L = Pos (Eqts)
            by (metis equation.exhaust positive-literal.elims(2))
    from this show ?thesis by auto
next
    assume negative-literal L
    then obtain ts where L = Neg (Eqt s)
        by (metis equation.exhaust negative-literal.elims(2))
    from this show ?thesis by auto
qed
```


## qed

lemma lit-ord-irrefl:
shows $(L, L) \notin$ lit-ord
by (simp add: lit-ord-wf)
lemma lit-ord-subst:
assumes $(L, M) \in$ lit-ord
shows $(($ subst-lit $L \sigma),($ subst-lit $M \sigma)) \in$ lit-ord
proof -
let ?f $=\lambda x$. (subst $x \sigma)$
have $i: \wedge t s .((t, s) \in$ trm-ord $\Longrightarrow(($ ?f $t),($ ?f $s)) \in$ trm-ord $)$
using trm-ord-subst by auto
from $\operatorname{assms}(1)$ have $i i:(($ mset-lit $L),($ mset-lit $M)) \in($ mult trm-ord $)$
unfolding lit-ord-def by auto
let ? $L=\{\#($ ?f $x) . x \in \#($ mset-lit $L) \#\}$
let $? M=\{\#($ ?f $x), x \in \#($ mset-lit $M) \#\}$
from $i$ and $i i$ have $i i i:(? L, ? M) \in($ mult trm-ord) using monotonic-fun-mult by metis
have $l: ? L=($ mset-lit (subst-lit L $\sigma$ ) $)$
using mset-lit-subst by auto
have $m: ? M=($ mset-lit $($ subst-lit $M \sigma))$
using mset-lit-subst by auto
from $l m$ iii show ?thesis unfolding lit-ord-def by auto
qed
lemma args-are-strictly-lower:
assumes is-compound $t$
shows (lhs $t, t) \in$ trm-ord $\wedge(r h s t, t) \in$ trm-ord
by (metis assms is-compound.elims(2) lhs.simps(1) rhs.simps(1) trm-ord-subterm)

```
lemma mset-subst:
    assumes \(C^{\prime}=\) subst-cl \(D \vartheta\)
    assumes \(\sigma \doteq \vartheta \diamond \eta\)
    assumes finite \(D\)
    shows mset-cl \(\left(C^{\prime}, \eta\right)=\) mset-cl \((D, \sigma) \vee\left(\right.\) mset-cl \(\left(C^{\prime}, \eta\right)\), mset-cl \(\left.(D, \sigma)\right) \in(\) mult
(mult trm-ord))
proof -
    let ?f \(=\lambda x\). (subst-lit \(x \vartheta)\)
    let \(? g=\lambda x\). \((\) mset-lit \((\) subst-lit \(x \eta))\)
    let \(? h=\lambda x\). \((\) mset-lit \((\) subst-lit \(x \sigma))\)
    have \(i: \forall x \in D .((? g(? f x))=(? h x))\)
    proof
    fix \(x\)
    have \((\) subst-lit \((\) subst-lit \(x\) \(\vartheta) \eta)=(\) subst-lit \(x(\vartheta \diamond \eta))\)
        using composition-of-substs-lit by auto
    from \(\operatorname{assms}(2)\) have \((\) subst-lit \(x \quad \sigma)=(\) subst-lit \(x(\vartheta \diamond \eta))\)
        using subst-eq-lit by auto
    from this «(subst-lit (subst-lit \(x\) ソ) \(\eta)=(\) subst-lit \(x(\vartheta \diamond \eta))\rangle\)
```

```
    show \((? g(? f x))=(? h x)\) by auto
    qed
    from \(\operatorname{assms}(3)\) have mset-set (?f' \(D) \subseteq \# \quad\{\#(? f x) . x \in \# m s e t-s e t(D) \#\}\)
    using mset-set-mset-image by auto
    from this have ii: \(\{\#(? g x) . x \in \# \operatorname{mset-set}(? f \cdot D) \#\} \subseteq \# \quad\{\#(? g x) . x \in \#\)
\(\{\#(? f x) \cdot x \in \# \operatorname{mset}-\operatorname{set}(D) \#\} \#\}\)
    using image-mset-subseteq-mono by auto
    have \(\{\#(? g x) . x \in \#\{\#(? f x) . x \in \# \operatorname{mset}-\operatorname{set}(D) \#\} \#\}=\{\#(? g(? f x))\).
\(x \in \#\) mset-set \(D \#\}\)
    using mset-image-comp [of ?g ?f ] by auto
    from this and \(i i\) have
    iii: \(\{\#(? g x) . x \in \#\) mset-set \((? f \cdot D) \#\} \subseteq \#\{\#(? g(? f x)) \cdot x \in \#\) mset-set
\(D \#\}\) by auto
    from \(i\) have \(\{\#(? g(? f x)) . x \in \#(\) mset-set \(D) \#\}=\{\#(? h x) . x \in \#(\) mset-set
D) \#\}
    using equal-image-mset \([\) of \(D \lambda x\). (?g (?f \(x)\) )] by auto
    from this and \(i i i\)
    have \(\{\#(? g x) . x \in \#\) mset-set \((? f ' D) \#\} \subseteq \#\{\#(? h x) . x \in \#\) mset-set \(D\)
\#\} by auto
    from this
    have iv: \(\{\#(? g x) . x \in \#\) mset-set \((? f\) ' \(D) \#\} \subseteq \#\) mset-cl \((D, \sigma)\) by auto
    from assms (1) have \(\left((\lambda \text {. subst-lit } x \geqslant)^{\prime} D\right)=C^{\prime}\) by auto
    from this and \(i v\) have \(\left\{\#\right.\) mset-lit (subst-lit \(x \eta\) ). \(x \in \#\) mset-set \(\left.C^{\prime} \#\right\} \subseteq \#\)
mset-cl \((D, \sigma)\)
    by auto
    from this have mset-cl \(\left(C^{\prime}, \eta\right) \subseteq \#\) mset-cl \((D, \sigma)\) by auto
    from this show ?thesis using multiset-order-inclusion-eq mult-trm-ord-trans by
auto
qed
lemma max-lit-exists:
    shows finite \(C \Longrightarrow C \neq\{ \} \longrightarrow\) ground-clause \(C \longrightarrow(\exists L .(L \in C \wedge\) (maximal-literal
\(L C)\) ))
proof (induction rule: finite.induct)
    case emptyI
    show ?case by simp
next
    fix \(A\) :: ' \(a\) clause and \(a::\) 'a literal
    assume finite \(A\)
    assume hyp-ind: \(A \neq\{ \} \longrightarrow\) ground-clause \(A \longrightarrow(\exists L .(L \in A \wedge\) (maximal-literal
\(L A)\) )
    show (insert a \(A) \neq\{ \} \longrightarrow\) ground-clause (insert a \(A\) )
                \(\longrightarrow(\exists L .(L \in(\) insert \(a A) \wedge(\) maximal-literal \(L(\) insert a \(A))))\)
    proof ((rule impI)+)
    assume insert a \(A \neq\{ \}\)
    assume ground-clause (insert a \(A\) )
    show \((\exists L .(L \in(\) insert \(a A) \wedge(\) maximal-literal \(L(\) insert a \(A))))\)
    proof (cases)
            assume \(A=\{ \}\)
```

```
        then have a\in(insert a A)^(maximal-literal a (insert a A))
            unfolding maximal-literal-def using lit-ord-irrefl by auto
    then show ?thesis by auto
    next assume }A\not={
    have insert a A={a}\cupA by auto
    then have vars-of-cl (insert a A) = vars-of-cl A U vars-of-lit a by auto
    from this and <ground-clause (insert a A)〉 have
        vars-of-lit a ={} and ground-clause A by auto
    from 〈ground-clause A` and «A\not={}` and hyp-ind obtain b where
        b\inA and maximal-literal }bA\mathrm{ by auto
    show ?thesis
    proof (cases)
    assume maximal-literal a A
    then have maximal-literal a (insert a A)
        using lit-ord-wf maximal-literal-def by auto
    then show ?thesis by auto
    next
    assume }\neg\mathrm{ maximal-literal a A
    then obtain }\mp@subsup{a}{}{\prime}\mathrm{ where }\mp@subsup{a}{}{\prime}\inA\mathrm{ and (a,a})\inlit-or
        unfolding maximal-literal-def by auto
    from }\langle\mp@subsup{a}{}{\prime}\inA\rangle\mathrm{ and <maximal-literal b A> have (b,a') & lit-ord
        unfolding maximal-literal-def by auto
    from this and «(a,a})\inlit-ord
        have (b,a)\not\in lit-ord unfolding lit-ord-def
        using mult-def trancl-trans by fastforce
        from this and «maximal-literal b A` have maximal-literal b (insert a A)
        unfolding maximal-literal-def by simp
        from this and }\langleb\inA\rangle\mathrm{ show ?thesis by auto
    qed
    qed
qed
qed
```

We deduce that a clause contains at least one eligible literal.

```
lemma eligible-lit-exists:
    assumes finite (cl-ecl C)
    assumes \((\) cl-ecl \(C) \neq\{ \}\)
    assumes (ground-clause (subst-cl (cl-ecl C) \(\sigma\) ))
    shows \(\exists\). \(((\) eligible-literal \(L C \sigma) \wedge(L \in(\) cl-ecl \(C)))\)
proof (cases)
    assume sel (cl-ecl C) \(=\{ \}\)
    let \(? \mathrm{C}=(\) subst-cl \((c l-e c l ~ C) \sigma)\)
    have finite ?C by (simp add: assms(1))
    have \(? C \neq\{ \}\)
    proof -
    from assms(2) obtain \(L\) where \(L \in(\) cl-ecl C) by auto
    from this have (subst-lit \(L \sigma\) ) \(\in\) ?C by auto
    from this show \(? C \neq\{ \}\) by auto
```

qed
from 〈finite ？$C\rangle\langle ? C \neq\{ \}\rangle \operatorname{assms}(3)$ obtain $L$ where $L \in ? C$ maximal－literal $L$ ？C using max－lit－exists by metis
from $\langle L \in ? C\rangle$ obtain $L^{\prime}$ where $L^{\prime} \in($ cl－ecl $C)$ and $L=\left(\right.$ subst－lit $\left.L^{\prime} \sigma\right)$ by auto
from $\left\langle L^{\prime} \in(\right.$ cl－ecl $\left.C)\right\rangle\left\langle L=\left(\right.\right.$ subst－lit $\left.\left.L^{\prime} \sigma\right)\right\rangle\langle m a x i m a l-l i t e r a l ~ L ? C\rangle\langle s e l(c l-e c l$ $C)=\{ \}$＞
show ？thesis unfolding eligible－literal－def by metis
next
assume sel $($ cl－ecl $C) \neq\{ \}$
then obtain $L$ where $L \in$ sel（cl－ecl C）by auto
from this show ？thesis unfolding eligible－literal－def using sel－neg by blast qed

The following lemmata provide various ways of proving that literals are ordered，depending on the relations between the terms they contain．

```
lemma lit-ord-dominating-term:
    assumes \((s 1, s \mathcal{Q}) \in\) trm-ord \(\vee(s 1, t \mathcal{Q}) \in\) trm-ord
    assumes orient-lit x1 s1 t1 p1
    assumes orient-lit x2 s2 t2 p2
    assumes vars-of-lit \(x 1=\{ \}\)
    assumes vars-of-lit x2 \(=\{ \}\)
    shows \((x 1, x 2) \in\) lit-ord
proof -
    from 〈vars-of-lit \(x 1=\{ \}\) and \(\langle\) orient-lit \(x 1\) s1 t1 p1〉 have vars-of t1 \(=\{ \}\) and
vars-of s1 \(=\{ \}\)
    and \(\neg(s 1, t 1) \in\) trm-ord unfolding orient-lit-def by auto
    from \(\operatorname{assms}(5)\) and <orient-lit x2 s2 t2 \(p 2\) 〉 have vars-of t2 \(=\{ \}\) and vars-of s2
\(=\{ \}\)
    and \(\neg(s 2, t 2) \in\) trm-ord unfolding orient-lit-def by auto
    from \(\langle\) vars-of \(t 1=\{ \}\rangle\) and \(\langle\) vars-of \(s 1=\{ \}\rangle\) and \(\langle\neg(s 1, t 1) \in\) trm-ord \(\rangle\)
        have o1: \(t 1=s 1 \vee(t 1, s 1) \in\) trm-ord using trm-ord-ground-total
        unfolding ground-term-def by auto
    from \(\langle\) vars- of t2 \(=\{ \}\rangle\) and \(\langle\) vars-of \(s 2=\{ \}\rangle\) and \(\langle\neg(s 2, t 2) \in\) trm-ord \(\rangle\)
        have o2: \(\operatorname{t2}=s \mathcal{Z} \vee(t \mathcal{Z}, s \mathcal{Z}) \in\) trm-ord using trm-ord-ground-total
        unfolding ground-term-def by auto
    from \(\langle\neg(s 2\), t2 \() \in\) trm-ord \(\rangle\) and \(\operatorname{assms}(1)\) have \((s 1, s 2) \in\) trm-ord
        by (metis assms(1) o2 trm-ord-trans transE)
    let ? \(m 1=\) mset-lit \(x 1\)
    let \(? m 2=\) mset-lit \(x 2\)
    from \(\operatorname{assms}(1)\) and o1 and o2 have \((t 1, s 2) \in\) trm-ord using trm-ord-trans
        trans-def by metis
    from this and \(\langle(s 1, s 2) \in\) trm-ord \(\rangle\) have
        s2max: \(\forall x .(x \in \#\{\# t 1, t 1, s 1, s 1 \#\} \longrightarrow(x, s 2) \in\) trm-ord \()\)
        by auto
    have \(\{\#\) s2 \(\#\} \subset \#\{\# t 2, t 2, s 2, s 2 \#\}\) by simp
```



```
    have ( \(\{\#\) s2 \# \}, \(\{\#\) t2, t2,s2,s2 \#\}) \() \in\) mult trm-ord
        using trm-ord-trans multiset-order-inclusion \([\) of \(\{\#\) s2 \(\#\}\{\#\) t2,t2,s2,s2 \# \}
```

```
trm-ord] by auto
    have p1=neg \vee p1 = pos using sign.exhaust by auto
    then show ?thesis
    proof
    assume p1 = neg
            from this and <orient-lit x1 s1 t1 p1> have x1 = (Neg (Eq t1 s1)) \vee x1 =
(Neg (Eq s1 t1))
            using orient-lit-def by blast
            from this have m1: ?m1 = {# t1,t1,s1,s1#} using mset-lit.simps by auto
    have p2 = neg \vee p2 = pos using sign.exhaust by auto
    then show ?thesis
    proof
            assume p2 = neg
            from this and <orient-lit x2 s2 t2 p2` have x2 = (Neg (Eq t2 s2)) \vee x2 =
(Neg (Eq s2 t2))
            using orient-lit-def by blast
            from this have m2: ?m2 = {# t2,t2,s2,s2 #} using mset-lit.simps by auto
            from s2max have ({#t1,t1,s1,s1#},{# s2 #}) \in mult trm-ord
                    using mult1-def-lemma [of {# s2 #} {#} s2 {# t1,t1,s1,s1 #} {#
t1,t1,s1,s1 #} trm-ord]
            mult-def
            by auto
            from<({# s2 #},{# t2,t2,s2,s2 #} ) \in mult trm-ord` and <({# t1,t1,s1,s1
#},{# s2 #}) \in mult trm-ord>
                have ({# t1,t1,s1,s1 #}, {# t2,t2,s2,s2 #} ) \in mult trm-ord
                using mult-trm-ord-trans unfolding trans-def by blast
            from this and m1 and m2 show ?thesis
                using lit-ord-def by auto
            next assume p2 = pos
            from this and <orient-lit x2 s2 t2 p2` have x2 = (Pos (Eq t2 s2)) \vee x2 =
(Pos (Eq s2 t2))
                using orient-lit-def by blast
            from this have m2: ?m2 = {# t2,s2 #} using mset-lit.simps by auto
            from s2max have ({# t1,t1,s1,s1#},{# s2 #}) \in mult trm-ord
                using mult1-def-lemma [of {# s2 #} {#} s2 {# t1,t1,s1,s1 #} {#
t1,t1,s1,s1 #} trm-ord]
            mult-def
            by auto
            from this and <( {# s2 #}, {# t2,t2,s2,s2 #} ) \in mult trm-ord>
                have ({# t1,t1,s1,s1 #}, {# t2,s2 #}) \in mult trm-ord
            using mset-ordering-add1 [of {# t1,t1,s1,s1 #} {# s2 #} trm-ord t2] by
(auto)
            from this and m1 and m2 show ?thesis
                using lit-ord-def by auto
    qed
    next
```

```
    assume p1 = pos
    from this and <orient-lit x1 s1 t1 p1` have x1 = (Pos (Eq t1 s1)) \vee x1 =
(Pos (Eq s1 t1))
    using orient-lit-def by blast
    from this have m1: ?m1 = {# t1,s1 #} using mset-lit.simps by auto
    have p2 = neg \vee p2 = pos using sign.exhaust by auto
    then show ?thesis
    proof
        assume p2 = neg
            from this and <orient-lit x2 s2 t2 p2` have x2 = (Neg (Eq t2 s2)) \vee x2 =
(Neg (Eq s2 t2))
            using orient-lit-def by blast
    from this have m2: ?m2 = {# t2,t2,s2,s2 #} using mset-lit.simps by auto
        from s2max have ({# t1,s1 #},{# s2 #}) \in mult trm-ord
            using mult1-def-lemma [of {# s2 #} {#} s2 {# t1,s1 #} {# t1,s1 #}
trm-ord]
            mult-def
            by auto
            from}<({# s2 #}},{# t2,t2,s2,s2 #} ) \in mult trm-ord> and <({# t1,s
#}, {# s2 #}) \in mult trm-ord>
            have ( {# t1,s1 #}, {# t2,t2,s2,s2 #} ) \in mult trm-ord
            using mult-trm-ord-trans unfolding trans-def by blast
        from this and m1 and m2 show ?thesis
            using lit-ord-def by auto
            next assume p2 = pos
            from this and <orient-lit x2 s2 t2 p2> have x2 = (Pos (Eq t2 s2)) \vee x2 =
(Pos (Eq s2 t2))
                using orient-lit-def by blast
            from this have m2: ?m2 = {# t2,s2 #} using mset-lit.simps by auto
            from s2max have ({# t1,s1 #}, {# s2 #}) \in mult trm-ord
                using mult1-def-lemma [of {# s2 #} {#} s2 {# t1,s1 #} {# t1,s1 #}
trm-ord]
                mult-def
                by auto
            from this have ({# t1,s1 #},{# t2,s2 #}) \in mult trm-ord
                using mset-ordering-add1 [of {# t1,s1 #} {# s2 #} trm-ord t2] by auto
            from this and m1 and m2 show ?thesis
                using lit-ord-def by auto
    qed
qed
qed
lemma lit-ord-neg-lit-lhs:
assumes orient-lit x1 s t1 pos
assumes orient-lit x2 s t2 neg
assumes vars-of-lit \(x 1=\{ \}\)
assumes vars-of-lit x2 \(=\{ \}\)
shows \((x 1, x 2) \in\) lit-ord
```

```
proof -
    from assms(3) and assms(1) have vars-of t1 ={} and vars-of s={}
        and}\neg(s,t1)\in\mathrm{ trm-ord unfolding orient-lit-def by auto
    from assms(4) and assms(2) have vars-of t2 = {}
        and}\neg(s,t2)\in\mathrm{ trm-ord unfolding orient-lit-def by auto
    from 〈vars-of t1 = {}> and <vars-of s={}> and <\neg(s,t1) \in trm-ord\rangle
        have o1:t1=s\vee(t1,s)\in trm-ord using trm-ord-ground-total
        unfolding ground-term-def by auto
    from 〈vars-of t2 = {}> and <vars-of s={}> and <\neg(s,t2) \in trm-ord\rangle
        have o2: t2 =s\vee (t2,s)\in trm-ord using trm-ord-ground-total
        unfolding ground-term-def by auto
    let ?m1 = mset-lit x1
    let ?m2 = mset-lit x2
    from «orient-lit x1 s t1 pos` have x1 = (Pos (Eq t1 s))\vee x1 =(Pos (Eq s t1))
        using orient-lit-def by blast
    from this have m1:?m1 = {#t1,s#} using mset-lit.simps by auto
    from <orient-lit x2 s t2 neg> have x2 = (Neg (Eq t2 s))\vee x2 = (Neg (Eq s t2))
            using orient-lit-def by blast
    from this have m2: ?m2 = {# t2,t2,s,s#} using mset-lit.simps by auto
    show ?thesis
    proof (cases)
    assume t1 = s
    have ({# s,s#}, {# t2,s,s#}) \in mult trm-ord
            using mult1-def-lemma [of {# t2,s,s#} {# s,s#} t2 {# s,s#} {#}
trm-ord]
            mult-def by auto
        then have ({# s,s#}, {# t2,t2,s,s#}) \in mult trm-ord
            using mset-ordering-add1 [of {# s,s#} {# t2,s,s#} trm-ord t2] by auto
        from this and \langlet1 =s\rangle and m1 and m2 show ?thesis using lit-ord-def by
auto
    next
        assume t1 \not=s
        from this and o1 have (t1,s) \in trm-ord by auto
        from this have smax: }\forallx.(x\in#{#t1#}\longrightarrow(x,s)\in\mathrm{ trm-ord )
            by auto
        from smax have ({#t1,s#},{# s,s#})\in mult trm-ord
            using mult1-def-lemma [of {# s,s#}{# s#} s {# t1,s#}{# t1 #}
trm-ord]
            mult-def by auto
        from this have ({#t1,s#},{# t2,s,s#})\in mult trm-ord
            using mset-ordering-add1 [of {# t1,s#} {# s,s#} trm-ord t2] by auto
    from this have ({# t1,s#},{# t2,t2,s,s#}) \in mult trm-ord
                using mset-ordering-add1 [of {# t1,s #} {# t2,s,s #} trm-ord t2] by
auto
    from this and m1 and m2 show ?thesis using lit-ord-def by auto
        qed
qed
lemma lit-ord-neg-lit-rhs:
```

```
    assumes orient-lit x1 s t1 pos
    assumes orient-lit x2 t2 s neg
    assumes vars-of-lit x1 = {}
    assumes vars-of-lit x2 = {}
    shows (x1,x2) \in lit-ord
proof -
    from assms(3) and assms(1) have vars-of t1 = {} and vars-of s={}
        and}\neg(s,t1)\in\mathrm{ trm-ord unfolding orient-lit-def by auto
    from assms(4) and assms(2) have vars-of t2 = {}
        and }\neg(t2,s)\in\mathrm{ trm-ord unfolding orient-lit-def by auto
    from 〈vars-of t1 = {}> and <vars-of s={}> and <\neg(s,t1) \in trm-ord\rangle
        have o1: t1 =s\vee (t1,s) \in trm-ord using trm-ord-ground-total
        unfolding ground-term-def by auto
    from <vars-of t2 = {}> and <vars-of s = {}> and }\langle\neg(t2,s)\in trm-ord
        have o2: t2 =s\vee (s,t2) \in trm-ord using trm-ord-ground-total
        unfolding ground-term-def by auto
    let ?m1 = mset-lit x1
    let ?m2 = mset-lit x2
    from <orient-lit x1 s t1 pos` have x1 = (Pos (Eq t1 s))\vee x1=(Pos (Eq s t1))
        using orient-lit-def by blast
    from this have m1:?m1 = {# t1,s#} using mset-lit.simps by auto
    from <orient-lit x2 t2 s neg> have x2 = (Neg (Eq t2 s))\vee x2 = (Neg (Eq s t2))
    using orient-lit-def by blast
    from this have m2: ?m2 = {# t2,t2,s,s#} using mset-lit.simps by auto
    show ?thesis
    proof (cases)
    assume t1 = s
    have ({# s,s#}, {# t2,s,s#}) \in mult trm-ord
                    using mult1-def-lemma [of {# t2,s,s#} {# s,s#} t2 {# s,s#} {#}
trm-ord]
            mult-def by auto
    then have ({# s,s#}, {# t2,t2,s,s#}) \in mult trm-ord
                using mset-ordering-add1 [of {# s,s#} {# t2,s,s#} trm-ord t2] by auto
            from this and <t1 =s\rangle and m1 and m2 show ?thesis using lit-ord-def by
auto
    next
        assume t1 \not=s
        from this and o1 have (t1,s)\in trm-ord by auto
    from this have smax:}\forallx.(x\in#{#t1#}\longrightarrow(x,s)\in\mathrm{ trm-ord )
        by auto
    from smax have ({# t1,s#},{# s,s#}) \in mult trm-ord
                using mult1-def-lemma [of {# s,s #} {# s #} s {# t1,s #} {# t1 #}
trm-ord]
        mult-def by auto
    from this have ({# t1,s#},{# t2,s,s#}) \in mult trm-ord
                using mset-ordering-add1 [of {# t1,s#} {# s,s#} trm-ord t2] by auto
    from this have ({# t1,s#},{# t2,t2,s,s#}) \in mult trm-ord
                using mset-ordering-add1 [of {# t1,s #} {# t2,s,s #} trm-ord t2] by
auto
```

from this and $m 1$ and $m 2$ show ？thesis using lit－ord－def by auto qed
qed
lemma lit－ord－rhs：
assumes $(t 1, t 2) \in$ trm－ord
assumes orient－lit x1 st1 p
assumes orient－lit x2 $s$ t2 $p$
assumes vars－of－lit $x 1=\{ \}$
assumes vars－of－lit x2 $=\{ \}$
shows $(x 1, x 2) \in$ lit－ord
proof－
from $\operatorname{assms}(2)$ and $\operatorname{assms}(4)$ have vars－of $t 1=\{ \}$ and vars－of $s=\{ \}$
and $\neg(s, t 1) \in$ trm－ord unfolding orient－lit－def by auto
from $\operatorname{assms}(3)$ and $\operatorname{assms}(5)$ have vars－of t2 $=\{ \}$ and $\neg(s, t 2) \in$ trm－ord unfolding orient－lit－def by auto
from 〈vars－of $t 1=\{ \}\rangle$ and $\langle$ vars－of $s=\{ \}\rangle$ and $\langle\neg(s, t 1) \in$ trm－ord $\rangle$ have o1：$t 1=s \vee(t 1, s) \in$ trm－ord using trm－ord－ground－total unfolding ground－term－def by auto
from 〈vars－of t2 $=\{ \}\rangle$ and $\langle$ vars－of $s=\{ \}\rangle$ and $\langle\neg(s$, t2 $) \in$ trm－ord $\rangle$ have o2： $\mathrm{t2}=s \vee(t 2, s) \in$ trm－ord using trm－ord－ground－total unfolding ground－term－def by auto
let ？$m 1=$ mset－lit $x 1$
let ？$m_{2}=m$ set－lit $x \mathcal{L}$
have $p=p o s \vee p=n e g$ using sign．exhaust by auto
then show？？thesis
proof
assume $p=p o s$
from this and＜orient－lit x1st1 p＞have $x 1=($ Pos $(E q$ t1 s $)) \vee x 1=($ Pos （Eqst1））
using orient－lit－def by blast
from this have $m 1$ ：？$m 1=\{\# t 1, s \#\}$ using mset－lit．simps by auto
from $\langle p=$ pos $\rangle$ and $\langle o r i e n t-l i t ~ x 2 ~ s t 2 ~ p 〉$ have $x 2=(\operatorname{Pos}(E q$ t2 $s)) \vee x 2=$
（ $\operatorname{Pos}(E q s t 2))$
using orient－lit－def by blast
from this have m2：？$m 2=\{\# t 2, s \#\}$ using mset－lit．simps by auto
from $\operatorname{assms}(1)$ have $(\forall b . b \in \#\{\# t 1 \#\} \longrightarrow(b$, t2）$\in$ trm－ord $)$ by auto
then have $(\{\# t 1, s \#\},\{\# t 2, s \#\}) \in$ mult trm－ord
using mult1－def－lemma $[$ of $\{\# t 2, s \#\}\{\# s \#\} t 2\{\# t 1, s \#\}\{\# t 1 \#\}$ trm－ord］
mult－def by auto
from this and $m 1$ and $m 2$ show ？thesis using lit－ord－def by auto
next assume $p=$ neg
from this and＜orient－lit x1 st1 p〉have $x 1=(\operatorname{Neg}(E q t 1 s)) \vee x 1=(N e g$ （Eqst1））
using orient－lit－def by blast
from this have $m 1$ ：？$m 1=\{\# t 1, t 1, s, s \#\}$ using mset－lit．simps by auto
from $\langle p=$ neg〉 and $<o r i e n t-l i t ~ x 2 ~ s t 2 ~ p 〉$ have $x 2=(\operatorname{Neg}(E q$ t2 $s)) \vee x 2=$ （Neg（Eq st2））
using orient-lit-def by blast
from this have m2: ? $m 2=\{\# t 2, t 2, s, s \#\}$ using mset-lit.simps by auto
from $\operatorname{assms}(1)$ have $\max :(\forall b . b \in \#\{\# t 1, t 1 \#\} \longrightarrow(b, t 2) \in$ trm-ord $)$ by auto
have $i$ : $\{\# t 2, s, s \#\}=\{\# s, s, t 2 \#\}$ by (simp add: add.commute add.left-commute)
have $i i$ : $\{\# t 1, t 1, s, s \#\}=\{\# s, s, t 1, t 1 \#\}$ by (simp add: add.commute add.left-commute)
from $i$ and $i i$ and max have $(\{\# t 1, t 1, s, s \#\},\{\# t 2, s, s \#\}) \in$ mult trm-ord
using mult1-def-lemma $[$ of $\{\# t 2, s, s \#\}\{\# s, s \#\}$ t2 $\{\# t 1, t 1, s, s \#\}\{\#$ t1, t1 \#\} trm-ord] mult-def by auto
then have $(\{\# t 1, t 1, s, s \#\},\{\# t 2, t 2, s, s \#\}) \in$ mult trm-ord using mset-ordering-add1 [of $\{\# t 1, t 1, s, s \#\}\{\# t 2, s, s \#\}$ trm-ord t2] by auto
from this and $m 1$ and $m 2$ show ?thesis using lit-ord-def by auto qed
qed
We show that the replacement of a term by an equivalent term preserves the semantics.

```
lemma trm-rep-preserves-eq-semantics:
    assumes fo-interpretation I
    assumes (I (subst t1 \sigma) (subst t2 \sigma))
    assumes (validate-ground-eq I (subst-equation (Eq t1 s) \sigma))
    shows (validate-ground-eq I (subst-equation (Eq t2 s) \sigma))
proof -
    from assms(1) have transitive I and symmetric I unfolding
            fo-interpretation-def congruence-def equivalence-relation-def by auto
    have (subst-equation (Eq t1 s) \sigma)=(Eq(subst t1 \sigma) (subst s \sigma)) by simp
    from this and assms(3) have I (subst t1 \sigma) (subst s \sigma) by simp
    from this and assms(2) and <transitive I〉 and <symmetric I`
        have I (subst t2 \sigma) (subst s \sigma)
        unfolding transitive-def symmetric-def by metis
    have (subst-equation (Eq t2 s) \sigma) = (Eq (subst t2 \sigma) (subst s \sigma)) by simp
    from this and \I (subst t2 \sigma) (subst s \sigma)\rangle show ?thesis by simp
qed
lemma trm-rep-preserves-lit-semantics:
    assumes fo-interpretation I
    assumes (I (subst t1 \sigma) (subst t2 \sigma))
    assumes orient-lit-inst L t1 s polarity \sigma'
    assumes \neg(validate-ground-lit I (subst-lit L \sigma))
    shows \negvalidate-ground-lit I (subst-lit (mk-lit polarity (Eq t2 s))\sigma)
proof -
    from assms(1) have transitive I and symmetric I unfolding
    fo-interpretation-def congruence-def equivalence-relation-def by auto
```

```
have polarity = pos \vee polarity = neg using sign.exhaust by auto
then show ?thesis
proof
    assume polarity = pos
    from this have mk: (mk-lit polarity (Eq t2 s)) = (Pos (Eq t2 s)) by auto
    from<polarity = pos` and assms(3) have L = (Pos (Eq t1 s)) \vee L = (Pos
(Eqs t1))
            unfolding orient-lit-inst-def by auto
    then show ?thesis
    proof
        assume L =(Pos (Eqt1 s))
        from this and assms(4) have }\negI(\mathrm{ subst t1 }\sigma\mathrm{ ) (subst s O) by simp
        from this and assms(2) and <transitive I〉 and <symmetric I\rangle
            have}\negI(\mathrm{ subst t2 }\sigma\mathrm{ ) (subst s }\sigma\mathrm{ )
            unfolding transitive-def symmetric-def by metis
            from this and mk show ?thesis by simp
    next
        assume L =(Pos (Eq s t1))
        from this and assms(4) have }\negI(\mathrm{ subst s }\sigma)(\mathrm{ subst t1 }\sigma\mathrm{ ) by simp
        from this and assms(2) and <transitive I〉 and <symmetric I〉
            have}\negI(subst t2 \sigma) (subst s \sigma
            unfolding transitive-def symmetric-def by metis
        from this and mk show ?thesis by simp
    qed
    next
    assume polarity = neg
    from this have mk: (mk-lit polarity (Eq t2 s)) = (Neg (Eq t2 s)) by auto
    from〈polarity = neg> and assms(3) have L = (Neg (Eq t1 s)) \vee L=(Neg
(Eq s t1))
            unfolding orient-lit-inst-def by auto
    then show ?thesis
    proof
            assume L = (Neg (Eqt1 s))
            from this and assms(4) have I (subst t1 \sigma) (subst s \sigma) by simp
            from this and assms(2) and <transitive I\rangle and <symmetric I>
                have I (subst t2 \sigma) (subst s \sigma)
                    unfolding transitive-def symmetric-def by metis
            from this and mk show ?thesis by simp
    next
            assume L = (Neg(Eq s t1))
            from this and assms(4) have I (subst s \sigma) (subst t1 \sigma) by simp
            from this and assms(2) and <transitive I〉 and <symmetric I\rangle
                    have}I\mathrm{ (subst t2 }\sigma\mathrm{ ) (subst s }\sigma\mathrm{ )
            unfolding transitive-def symmetric-def by metis
            from this and mk show ?thesis by simp
    qed
qed
qed
```


## lemma subterms－dominated ：

assumes maximal－literal $L C$
assumes orient－lit Ltsp
assumes $u \in$ subterms－of－cl $C$
assumes vars－of－lit $L=\{ \}$
assumes vars－of－cl $C=\{ \}$
shows $u=t \vee(u, t) \in$ trm－ord
proof（rule ccontr）
assume neg－$h: \neg(u=t \vee(u, t) \in$ trm－ord $)$
from $\operatorname{assms}(5)$ and $\operatorname{assms}(3)$ have vars－of $u=\{ \}$ using subterm－vars by blast
from 〈vars－of－lit $L=\{ \}\rangle$ and $\langle$ orient－lit $L t s p\rangle$ have vars－of $s=\{ \}$ and vars－of $t=\{ \}$
and $\neg(t, s) \in$ trm－ord unfolding orient－lit－def by auto
from assms（3）obtain $L^{\prime}$ where $u \in$ subterms－of－lit $L^{\prime}$ and $L^{\prime} \in C$ by auto
from $\operatorname{assms}(5)$ and $\left\langle L^{\prime} \in C\right\rangle$ have vars－of－lit $L^{\prime}=\{ \}$ using vars－of－cl．simps by auto
from $\left\langle u \in\right.$ subterms－of－lit $\left.L^{\prime}\right\rangle$ obtain $t^{\prime} s^{\prime} p^{\prime}$ where orient－lit $L^{\prime} t^{\prime} s^{\prime} p^{\prime}$
and $u \in$ subterms－of $t^{\prime} \cup$ subterms－of $s^{\prime}$ unfolding orient－lit－def
by（metis Un－commute mset－lit．cases subterms－of－eq．simps subterms－of－lit．simps（1）
subterms－of－lit．simps（2）trm－ord－wf wf－asym）
from $\left\langle u \in\right.$ subterms－of $t^{\prime} \cup$ subterms－of $\left.s^{\prime}\right\rangle$ have $u \in$ subterms－of $t^{\prime} \vee u \in$
subterms－of $s^{\prime}$ by auto
then show False
proof
assume $u \in$ subterms－of $t^{\prime}$
from this have $u=t^{\prime} \vee\left(u, t^{\prime}\right) \in$ trm－ord
using subterms－of－trm－ord－eq［of $\left.u t^{\prime}\right]$ by auto
from neg－h and $\langle$ vars－of $u=\{ \}\rangle$ and $\langle$ vars－of $t=\{ \}\rangle$ have $(t, u) \in$ trm－ord using trm－ord－ground－total unfolding ground－term－def by auto
from this and $\left\langle u=t^{\prime} \vee\left(u, t^{\prime}\right) \in\right.$ trm－ord $\rangle$ have $\left(t, t^{\prime}\right) \in$ trm－ord using trm－ord－trans unfolding trans－def by metis
from this and 〈vars－of－lit $\left.L^{\prime}=\{ \}\right\rangle$ and $\operatorname{assms}(4)$ and $\langle o r i e n t-l i t L t s p\rangle$ and 〈orient－lit $\left.L^{\prime} t^{\prime} s^{\prime} p^{\prime}\right\rangle$
have $\left(L, L^{\prime}\right) \in$ lit－ord using lit－ord－dominating－term by blast
from this and $\operatorname{assms}(1)$ and $\left\langle L^{\prime} \in C\right\rangle$ show False unfolding maximal－literal－def by auto
next
assume $u \in$ subterms－of $s^{\prime}$
from this have $u=s^{\prime} \vee\left(u, s^{\prime}\right) \in$ trm－ord
using subterms－of－trm－ord－eq［of $\left.u s^{\prime}\right]$ by auto
from neg－h and 〈vars－of $u=\{ \}\rangle$ and $\langle$ vars－of $t=\{ \}\rangle$ have $(t, u) \in$ trm－ord using trm－ord－ground－total unfolding ground－term－def by auto
from this and $\left\langle u=s^{\prime} \vee\left(u, s^{\prime}\right) \in\right.$ trm－ord $\rangle$ have $\left(t, s^{\prime}\right) \in$ trm－ord using trm－ord－trans unfolding trans－def by metis
from this and 〈vars－of－lit $\left.L^{\prime}=\{ \}\right\rangle$ and $\operatorname{assms}(4)$ and $\langle o r i e n t-l i t L t s p\rangle$ and 〈orient－lit $\left.L^{\prime} t^{\prime} s^{\prime} p^{\prime}\right\rangle$
have $\left(L, L^{\prime}\right) \in$ lit－ord using lit－ord－dominating－term by blast
from this and $\operatorname{assms}(1)$ and $\left\langle L^{\prime} \in C\right\rangle$ show False unfolding maximal－literal－def

## by auto

qed
qed
A term dominates an expression if the expression contains no strictly greater subterm:
fun dominate-eq:: 'a trm $\Rightarrow$ 'a equation $\Rightarrow$ bool where $($ dominate-eq $t(E q u v))=((t, u) \notin$ trm-ord $\wedge(t, v) \notin$ trm-ord $)$
fun dominate-lit:: 'a trm $\Rightarrow$ 'a literal $\Rightarrow$ bool
where $($ dominate-lit $t($ Pos e) $)=($ dominate-eq $t e) \mid$ $($ dominate-lit $t($ Neg $e))=($ dominate-eq $t e)$
definition dominate-cl:: 'a trm $\Rightarrow$ 'a clause $\Rightarrow$ bool where $($ dominate-cl $t C)=(\forall x \in C .($ dominate-lit $t x))$
definition no-disequation-in-cl:: 'a trm $\Rightarrow$ 'a clause $\Rightarrow$ bool where (no-disequation-in-cl $t C)=(\forall u v$.
$(\operatorname{Neg}(E q u v) \in C \longrightarrow(u \neq t \wedge v \neq t)))$
definition no-taut-eq-in-cl:: 'a trm $\Rightarrow$ 'a clause $\Rightarrow$ bool where (no-taut-eq-in-cl t C) $=(\operatorname{Pos}(E q t t) \notin C)$
definition eq-occurs-in-cl where
(eq-occurs-in-cl ts $C \sigma)=\left(\exists L t^{\prime} s^{\prime} .(L \in C) \wedge\left(\right.\right.$ orient-lit-inst $L t^{\prime} s^{\prime}$ pos $\left.\sigma\right)$ $\wedge\left(t=\right.$ subst $\left.t^{\prime} \sigma\right) \wedge\left(s=\right.$ subst $\left.\left.s^{\prime} \sigma\right)\right)$

### 4.4 Inference Rules

We now define the rules of the superposition calculus. Standard superposition is a refinement of the paramodulation rule based on the following ideas:
(i) the replacement of a term by a bigger term is forbidden;
(ii) the replacement can be performed only in the maximal term of a maximal (or selected) literal;
(iii) replacement of variables is forbidden.

Our definition imposes additional conditions on the positions on which the replacements are allowed: any superposition inference inside a term occurring in the set attached to the extended clause is blocked.

We consider two different kinds of inferences: ground or first-order. Ground inferences are those needed for completeness, first-order inferences are those actually used by theorem provers. For conciseness, these two notions of inferences are defined simultaneously, and a parameter is added to the corresponding functions to determine whether the inference is ground or firstorder.

```
datatype inferences = Ground | FirstOrder
```

The following function checks whether a given substitution is a unifier of two terms. If the inference is first-order then the unifier must be maximal.

```
definition ck-unifier where
    ck-unifier t s \sigma type \longleftrightarrow (if type = FirstOrder then min-IMGU \sigma t s else Unifier
\sigmats)
lemma ck-unifier-thm:
    assumes ck-unifier t s \sigma k
    shows (subst t \sigma) = (subst s \sigma)
    by (metis assms min-IMGU-def IMGU-iff-Idem-and-MGU MGU-is-Unifier ck-unifier-def
Unifier-def)
lemma subst-preserve-ck-unifier:
    assumes ck-unifier t s \sigma k
    shows ck-unifier t s (comp \sigma \eta) Ground
proof -
    let ? }\mp@subsup{\sigma}{}{\prime}=(\operatorname{comp}\sigma\eta
    from assms have (subst t \sigma)=(subst s \sigma)
        using ck-unifier-thm by auto
    then have (subst t ? , ')}=(\mathrm{ subst s ? }\mp@subsup{\sigma}{}{\prime})\mathrm{ by simp
    then show ?thesis unfolding ck-unifier-def Unifier-def by auto
qed
```

The following function checks whether a given term is allowed to be reduced according to the strategy described above, i.e., that it does not occur in the set of terms associated with the clause (we do not assume that the set of irreducible terms is closed under subterm thus we use the function occurs-in instead of a mere membership test.

```
definition allowed-redex
    where allowed-redex \(t C \sigma=(\neg(\exists s \in(\) trms-ecl \(C)\).
    (occurs-in (subst \(t \sigma\) ) (subst s \(\sigma\) ))))
```

The following function allows one to compute the set of irreducible terms attached to the conclusion of an inference. The computation depends on the type of the considered inference: for ground inferences the entire set of irreducible terms is kept. For first-order inferences, the function filter-trms is called to remove some of the terms (see also the function dom-trms below).

```
definition get-trms
    where
    get-trms C E t = (if ( }t=\mathrm{ FirstOrder) then(filter-trms C E) else E)
```

The following definition provides the conditions that allow one to propagate irreducible terms from the parent clauses to the conclusion. A term can be propagated if it is strictly lower than a term occurring in the derived clause, or if it occurs in a negative literal of the derived clause. Note that
this condition is slightly more restrictive than that of the basic superposition calculus, because maximal terms occurring in maximal positive literals cannot be kept in the set of irreducible terms. However in our definition, terms can be propagated even if they do not occur in the parent clause or in the conclusion. Extended clauses whose set of irreducible terms fulfills this property are called well-constrained.

```
definition dom-trm
    where dom-trm \(t C=\)
        \((\exists L u v p .(L \in C \wedge(\) decompose-literal \(L u v p)\)
            \(\wedge(((p=n e g \wedge t=u) \vee(t, u) \in\) trm-ord \())))\)
lemma dom-trm-lemma:
    assumes dom-trm \(t C\)
    shows \(\exists u\). \((u \in(\) subterms-of-cl \(C) \wedge(u=t \vee(t, u) \in\) trm-ord \())\)
proof -
    from \(\operatorname{assms}(1)\) obtain \(L u v p\) where
            \(L \in C\) decompose-literal \(L\) uvp \((u=t \vee(t, u) \in\) trm-ord \()\)
            unfolding dom-trm-def by blast
    from 〈decompose-literal \(L u v p\rangle\) have \(u \in\) subterms-of-lit \(L\)
            unfolding decompose-literal-def decompose-equation-def using root-subterm by
force
    from this and \(\langle L \in C\rangle\) have \(u \in\) (subterms-of-cl \(C\) ) by auto
    from this and \(\langle(u=t \vee(t, u) \in\) trm-ord \()\rangle\) show ?thesis by auto
qed
definition dom-trms
where
    dom-trms \(C E=\{x .(x \in E) \wedge(\) dom-trm \(x C)\}\)
lemma dom-trms-subset:
    shows (dom-trms \(C E) \subseteq E\)
unfolding dom-trms-def by auto
lemma dom-trm-vars:
    assumes dom-trm \(t C\)
    shows vars-of \(t \subseteq\) vars-of-cl \(C\)
proof -
    from assms obtain \(L u v p\) where \(L \in C\) decompose-literal \(L u v p t=u \vee\)
\((t, u) \in\) trm-ord
            unfolding dom-trm-def by auto
    from \(\langle t=u \vee(t, u) \in\) trm-ord \(\rangle\) have vars-of \(t \subseteq\) vars-of \(u\) using trm-ord-vars
by blast
    from this and 〈decompose-literal \(L u v p\rangle\) have vars-of \(t \subseteq\) vars-of-lit \(L\) using
decompose-literal-vars by blast
    from this show ?thesis using \(\langle L \in C\rangle\) by auto
qed
definition well-constrained
```

$$
\text { where well-constrained } C=(\forall y .(y \in \text { trms-ecl } C \longrightarrow \text { dom-trm } y(\text { cl-ecl } C)))
$$

The next function allows one to check that a set of terms is in normal form. The argument $f$ denotes the function mapping a term to its normal form (we do not assume that $f$ is compatible with the term structure at this point).
definition all-trms-irreducible
where (all-trms-irreducible $E f)=(\forall x y .(x \in E \longrightarrow$ occurs-in y $x \longrightarrow(f y)=$ y))

Superposition We now define the superposition rule. Note that we assume that the parent clauses are variable-disjoint, but we do not explicitly rename them at this point, thus for completeness we will have to assume that the clause sets are closed under renaming. During the application of the rule, all the terms occurring at a position that is lower than that of the reduced term can be added in the set of irreducible terms attached to the conclusion (the intuition is that we assume that the terms occurring at minimal positions are reduced first). In particular, every proper subterm of the reduced term $u^{\prime}$ is added in the set of irreducible terms, thus every application of the superposition rule in a term introduced by unification will be blocked.
Clause P1 is the "into" clause and clause P2 is the "from" clause.

```
definition superposition ::
    'a eclause \(\Rightarrow\) ' \(a\) eclause \(\Rightarrow{ }^{\prime}\) ' eclause \(\Rightarrow\) 'a subst \(\Rightarrow\) inferences \(\Rightarrow\) ' a clause \(\Rightarrow\) bool
where
    (superposition P1 P2 \(C \sigma k C^{\prime}\) ) =
    ( \(\exists\) L t s u v M p Cl-P1 Cl-P2 Cl-C polarity \(t^{\prime} u^{\prime} L^{\prime}\) trms- \(C\).
            \((L \in C l-P 1) \wedge(M \in C l-P 2) \wedge(\) eligible-literal LP1 \(\sigma) \wedge(\) eligible-literal \(M\)
P2 \(\sigma\) )
        \(\wedge\) (variable-disjoint P1 P2)
        \(\wedge(C l-P 1=(\) cl-ecl P1 \()) \wedge(C l-P 2=(\) cl-ecl P2 \())\)
        \(\wedge\left(\neg\right.\) is-a-variable \(\left.u^{\prime}\right)\)
        \(\wedge\) (allowed-redex \(u^{\prime}\) P1 \(\sigma\) )
        \(\wedge\) trms- \(C=\) (get-trms Cl-C (dom-trms Cl-C (subst-set
            \(((\) trms-ecl P1) \(\cup(\) trms-ecl P2 \() \cup\)
                \(\{r . \exists q .(q, p) \in(\) pos-ord P1 t) \(\wedge(\operatorname{subterm} t q r)\}) \sigma)) k)\)
        \(\wedge(C=(E c l C l-C\) trms- \(C))\)
        \(\wedge\) (orient-lit-inst M u v pos \(\sigma\) )
        \(\wedge\) (orient-lit-inst Lts polarity \(\sigma\) )
        \(\wedge((\) subst \(u \sigma) \neq(\) subst \(v \sigma))\)
        \(\wedge\) (subterm t p \(u^{\prime}\) )
        \(\wedge\left(c k\right.\)-unifier \(\left.u^{\prime} u \sigma k\right)\)
        \(\wedge\) (replace-subterm t p v \(t^{\prime}\) )
        \(\wedge((k=\) FirstOrder \() \vee((\) subst-lit \(M \sigma),(\) subst-lit L \(\sigma)) \in\) lit-ord \()\)
        \(\wedge((k=\) FirstOrder \() \vee(\) strictly-maximal-literal P2 M \(\sigma))\)
        \(\wedge\left(L^{\prime}=m k\right.\)-lit polarity \(\left.\left(E q t^{\prime} s\right)\right)\)
        \(\wedge\left(C l-C=\left(\right.\right.\) subst-cl \(\left.\left.C^{\prime} \sigma\right)\right)\)
    \(\left.\wedge\left(C^{\prime}=(C l-P 1-\{L\}) \cup\left((C l-P 2-\{M\}) \cup\left\{L^{\prime}\right\}\right)\right)\right)\)
```

Reflexion We now define the Reflexion rule, which deletes contradictory literals (after unification). All the terms occurring in these literals can be added into the set of irreducible terms (intuitively, we can assume that these terms have been normalized before applying the rule). It is sufficient to add the term $t$, since every term occurring in the considered literal is a subterm of $t$ (after unification).

```
definition reflexion ::
    ' \(a\) eclause \(\Rightarrow\) ' \(a\) eclause \(\Rightarrow\) 'a subst \(\Rightarrow\) inferences \(\Rightarrow{ }^{\prime} a\) clause \(\Rightarrow\) bool
where
    (reflexion \(\left.P C \sigma k C^{\prime}\right)=\)
    ( \(\exists \mathrm{L} 1\) t s Cl-P Cl-C trms-C.
        (eligible-literal L1 P \(\sigma\) )
        \(\wedge(L 1 \in(\) cl-ecl \(P)) \wedge(C l-C=(\) cl-ecl \(C)) \wedge(C l-P=(\) cl-ecl \(P))\)
        \(\wedge\) (orient-lit-inst L1 t s neg \(\sigma\) )
        \(\wedge(c k\)-unifier \(t s \sigma k)\)
        \(\wedge(C=(E c l C l-C\) trms- \(C))\)
        \(\wedge\) trms-C \(=\) (get-trms Cl-C
                (dom-trms Cl-C (subst-set \(((\) trms-ecl \(P) \cup\{t\}) \sigma)) k\) )
        \(\wedge\left(C l-C=\left(\right.\right.\) subst-cl \(\left.\left.C^{\prime} \sigma\right)\right)\)
        \(\left.\wedge\left(C^{\prime}=((C l-P-\{L 1\}))\right)\right)\)
```

Factorization We now define the equational factorization rule, which merges two equations sharing the same left-hand side (after unification), if the right-hand sides are equivalent. Here, contrarily to the previous rule, the term $t$ cannot be added into the set of irreducible terms, because we cannot assume that this term is in normal form (e.g., the application of the equational factorization rule may yield a new rewrite rule of left-hand side $t$ ). However, all proper subterms of $t$ can be added.

```
definition factorization ::
    ' \(a\) eclause \(\Rightarrow\) ' \(a\) eclause \(\Rightarrow\) 'a subst \(\Rightarrow\) inferences \(\Rightarrow{ }^{\prime} a\) clause \(\Rightarrow\) bool
where
    (factorization \(\left.P C \sigma k C^{\prime}\right)=\)
    ( \(\exists\) L1 L2 L't suv Cl-P Cl-C trms-C.
        (eligible-literal L1 P \(\sigma\) )
        \(\wedge(L 1 \in(\) cl-ecl \(P)) \wedge(L 2 \in(\) cl-ecl \(P)-\{L 1\}) \wedge(C l-C=(\) cl-ecl \(C)) \wedge\)
\((C l-P=(\) cl-ecl \(P))\)
    \(\wedge\) (orient-lit-inst L1 t s pos \(\sigma\) )
    \(\wedge\) (orient-lit-inst L2 u v pos \(\sigma\) )
    \(\wedge((\) subst \(t \sigma) \neq(\) subst s \(\sigma))\)
    \(\wedge((\) subst \(t \sigma) \neq(\) subst \(v \sigma))\)
    \(\wedge(c k\)-unifier \(t u \sigma k)\)
    \(\wedge\left(L^{\prime}=\operatorname{Neg}(E q s v)\right)\)
```



```
    \(\wedge t r m s-C=(\) get-trms \(C l-C\)
        (dom-trms Cl-C \((\) subst-set \(((\) trms-ecl \(P) \cup(\) proper-subterms-of \(t)) \sigma)))\)
k)
    \(\wedge\left(C l-C=\left(\right.\right.\) subst-cl \(\left.\left.C^{\prime} \sigma\right)\right)\)
```

$$
\left.\wedge\left(C^{\prime}=\left((C l-P-\{L 2\}) \cup\left\{L^{\prime}\right\}\right)\right)\right)
$$

### 4.5 Derivations

We now define the set of derivable clauses by induction. Note that redundancy criteria are not taken into account at this point. Our definition of derivations also covers renaming.

```
definition derivable :: ' \(a\) eclause \(\Rightarrow\) ' \(a\) eclause set
    \(\Rightarrow\) 'a eclause set \(\Rightarrow\) 'a subst \(\Rightarrow\) inferences \(\Rightarrow\) ' \(a\) clause \(\Rightarrow\) bool
where
    (derivable CPS \(\quad\) k \(C^{\prime}\) ) \(=\)
        \(((\exists P 1 P 2 .(P 1 \in S \wedge P 2 \in S \wedge P=\{P 1, P 2\} \wedge\) superposition P1 P2 \(C\)
\(\left.\left.\sigma k C^{\prime}\right)\right)\)
    \(\vee\left(\exists P 1 .\left(P 1 \in S \wedge P=\{P 1\} \wedge\right.\right.\) factorization P1 C \(\left.\left.\sigma k C^{\prime}\right)\right)\)
    \(\vee\left(\exists P 1 .\left(P 1 \in S \wedge P=\{P 1\} \wedge\right.\right.\) reflexion P1 \(\left.\left.\left.C \sigma k C^{\prime}\right)\right)\right)\)
lemma derivable-premisses:
    assumes derivable \(C P S \sigma k C^{\prime}\)
    shows \(P \subseteq S\)
using assms derivable-def by auto
inductive derivable-ecl :: 'a eclause \(\Rightarrow\) 'a eclause set \(\Rightarrow\) bool
    where
        init [simp, intro!]: \(C \in S \Longrightarrow\) (derivable-ecl \(C\) S \() \mid\)
        \(r n\) [simp, intro!]: (derivable-ecl C S) \(\Longrightarrow\) (renaming-cl C D) \(\Longrightarrow\) (derivable-ecl
D S) |
    deriv [simp, intro!]: \(\forall x .(x \in P \longrightarrow(\) derivable-ecl \(x S)))\)
        \(\Longrightarrow\left(\right.\) derivable \(C P S^{\prime} \sigma\) FirstOrder \(\left.C^{\prime}\right) \Longrightarrow(\) derivable-ecl \(C S)\)
```

We define a notion of instance by associating clauses with ground substitutions.

```
definition instances:: 'a eclause set \(\Rightarrow\) ('a eclause \(\times\) ' \(a\) subst) set
    where instances \(S=\{x . \exists C \sigma .(C \in S \wedge\) (ground-clause (subst-cl (cl-ecl C)
\(\sigma)\) )
\(\wedge x=(C, \sigma))\}\)
definition clset-instances:: ('a eclause \(\times\) ' \(a\) subst) set \(\Rightarrow\) 'a clause set
where
    clset-instances \(S=\{C . \exists x .(x \in S \wedge C=(\) subst-cl \((\operatorname{cl-ecl}(\) fst \(x))(\) snd \(x)))\}\)
definition grounding-set
    where grounding-set \(S \sigma=(\forall x .(x \in S \longrightarrow\) (ground-clause (subst-cl (cl-ecl \(x)\)
\(\sigma))\) ))
```


## 5 Soundness

In this section, we prove that the conclusion of every inference rule is a logical consequence of the premises. Thus a clause set is unsatisfiable if the empty clause is derivable. For each rule, we first prove that all ground instances of the conclusion are entailed by the corresponding instances of the parent clauses, then we lift the result to first-order clauses. The proof is standard and straightforward, but note that we also prove that the derived clauses are finite and well-constrained.

```
lemma cannot-validate-contradictary-literals :
    assumes \(l=N e g(E q t t)\)
    assumes fo-interpretation I
    shows \(\neg\) (validate-ground-lit I l)
proof -
    from assms(2) have congruence I unfolding fo-interpretation-def by auto
    then have It \(t\) unfolding congruence-def reflexive-def equivalence-relation-def
by auto
    from this and assms(1) show ?thesis by auto
qed
lemma ground-reflexion-is-sound :
    assumes finite (cl-ecl C)
    assumes reflexion \(C D \sigma k C^{\prime}\)
    assumes (ground-clause (subst-cl (cl-ecl D) \(\vartheta)\) )
    shows clause-entails-clause (subst-cl (subst-cl (cl-ecl C) \(\sigma\) ) \(\vartheta)\)
            (subst-cl (cl-ecl D) \(\vartheta)\)
proof (rule ccontr)
    let ? \(C=(\) cl-ecl \(C)\)
    let ? \(D=(\) cl-ecl \(D)\)
    let ? \(C^{\prime}=(\) subst-cl \((\) subst-cl \((c l-e c l ~ C) \sigma) ~ \vartheta) ~\)
    let \(? D^{\prime}=(\) subst-cl \((\) cl-ecl D) \(\vartheta)\)
    assume \(\neg\left(\right.\) clause-entails-clause ? \(\left.C^{\prime} ? D^{\prime}\right)\)
    then obtain \(I\) where validate-clause \(I ? C^{\prime}\) and \(\neg\left(\right.\) validate-clause \(\left.I ? D^{\prime}\right)\)
fo-interpretation I
    unfolding clause-entails-clause-def by auto
    from \(\operatorname{assms}\) (2) obtain L1ts where
        \(? D=(\) subst-cl \((? C-\{L 1\}) \sigma)\)
        and orient-lit-inst L1 \(t s\) neg \(\sigma\) and \(c k\)-unifier \(t s \sigma k\)
            using reflexion-def \([\) of \(C D \sigma k]\) by auto
    from assms(1) have finite (subst-cl (subst-cl ?C \(\sigma\) ) \(\vartheta\) ) by auto
    then obtain \(\eta\) where \(i\) : ground-clause (subst-cl
            (subst-cl (subst-cl ?C \(\sigma\) ) খ) \(\eta\) )
        using ground-instance-exists [of (subst-cl (subst-cl ?C \(\sigma\) ) \(\vartheta\) )]
        by auto
    let ?CC \(=(\) subst-cl (subst-cl (subst-cl ?C \(\sigma) \vartheta) \eta)\)
    let \(? \sigma^{\prime \prime}=\operatorname{comp} \sigma \vartheta\)
    let ? \(\sigma^{\prime}=\mathrm{comp}\) ? \(\sigma^{\prime \prime} \eta\)
    have ?CC \(=\left(\right.\) subst-cl (subst-cl ? \(C\) ? \(\left.\sigma^{\prime \prime}\right) \eta\) )
```

using composition－of－substs－cl［of ？C］by auto
then have $? C C=\left(\right.$ subst－cl $? C$ ？$\left.\sigma^{\prime}\right)$
using composition－of－substs－cl［of ？C］by auto
from «validate－clause I（subst－cl（subst－cl（cl－ecl C）$\sigma$ ）$\vartheta$ ）〉
have validate－ground－clause I ？CC using $i$ validate－clause．simps by blast
then obtain $l^{\prime}$ where $l^{\prime} \in ? C C$ and validate－ground－lit I $l^{\prime}$ by auto
from $\left\langle l^{\prime} \in ? C C\right\rangle$ and $\left\langle ? C C=\left(\right.\right.$ subst－cl ？$C$ ？$\left.\left.\sigma^{\prime}\right)\right\rangle$ obtain $l$ where
$l \in ? C$ and $l^{\prime}=\left(\right.$ subst－lit $\left.l ? \sigma^{\prime}\right)$ using subst－cl．simps by blast
have subst－lit $l \sigma \in$ ？$D$
proof（rule ccontr）
assume subst－lit $l \sigma \notin ? D$
from this and $\langle ? D=($ subst－cl $(? C-\{L 1\}) \sigma)\rangle$ and $\langle l \in ? C\rangle$
have $l=L 1$ by auto
from this and＜orient－lit－inst L1 ts neg $\sigma \succ$ have $l=(\operatorname{Neg}(E q t s)) \vee l=(N e g$ （Eqst））
unfolding orient－lit－inst－def by auto
from $\langle c k$－unifier $t s \sigma k\rangle$ have subst $t \sigma=$ subst $s \sigma$ using ck－unifier－thm by auto
then have subst（subst（subst $t \sigma$ ）$\vartheta$ ）$\eta=$ subst（subst（subst s $\sigma$ ）$\vartheta$ ）$\eta$ by auto
then have（subst $t ? \sigma^{\prime}$ ）＝subst $s$ ？$\sigma^{\prime}$ by auto
from this and $<l=(N e g(E q t s)) \vee l=(N e g(E q s t))\rangle$
have $\left(\right.$ subst－lit l ？$\left.\sigma^{\prime}\right)=\left(\right.$ Neg $\left(E q\left(\right.\right.$ subst $\left.t ? \sigma^{\prime}\right)\left(\right.$ subst $\left.\left.\left.t ? \sigma^{\prime}\right)\right)\right)$
by auto
from this and 〈fo－interpretation $I\rangle$ have $\neg$（validate－ground－lit $I$（subst－lit $l$ $\left.? \sigma^{\prime}\right)$ ）
using cannot－validate－contradictary－literals $\left[\right.$ of（subst－lit l ？$\left.\sigma^{\prime}\right)\left(\right.$ subst $\left.\left.t ? \sigma^{\prime}\right) I\right]$
by auto
from this and $\left\langle l^{\prime}=\right.$ subst－lit $\left.l ? \sigma^{\prime}\right\rangle$ and $\left\langle\right.$ validate－ground－lit I $\left.l^{\prime}\right\rangle$ show False
by auto
qed
from 〈subst－lit $l \sigma \in ? D\rangle$ and $\left\langle l^{\prime}=\right.$ subst－lit $l$ ？$\left.\sigma^{\prime}\right\rangle$ have $l^{\prime} \in($ subst－cl（subst－cl ？D খ）$\eta$ ）
using subst－cl．simps composition－of－substs－lit mem－Collect－eq by（metis（mono－tags，lifting））
from this and «validate－ground－lit I $\left.l^{\prime}\right\rangle$ have
validate－ground－clause I（subst－cl（subst－cl ？D খ）$\eta$ ）by auto
from 〈ground－clause（subst－cl ？D $\vartheta$ ）〉 have
$($ subst－cl ？D $\vartheta)=($ subst－cl $($ subst－cl ？$D \vartheta) \eta)$
using substs－preserve－ground－clause［of（subst－cl ？D $\vartheta$ ）$\eta$ ］by blast
from this and＜validate－ground－clause I（subst－cl（subst－cl ？D $\vartheta$ ）$\eta$ ）〉
have validate－ground－clause I（subst－cl ？D $\vartheta$ ）by force
from this and assms（3）and $\langle\neg$ validate－clause $I$（subst－cl（cl－ecl D）$\vartheta$ ）〉show False
using substs－preserve－ground－clause validate－clause．elims（3）by metis
qed
lemma reflexion－is－sound ：
assumes finite（cl－ecl C）

```
    assumes reflexion C D \sigmak C'
    shows clause-entails-clause (cl-ecl C) (cl-ecl D)
proof (rule ccontr)
    let ?C = (cl-ecl C)
    let ?D = (cl-ecl D)
    assume \neg(clause-entails-clause ?C ?D)
    then obtain I where validate-clause I ?C and }\neg(validate-clause I ?D) fo-interpretation
I
    unfolding clause-entails-clause-def by auto
    from <\neg (validate-clause I ?D)> obtain \vartheta
    where D-false: ᄀ(validate-ground-clause I (subst-cl ?D \vartheta))
        and (ground-clause (subst-cl ?D \vartheta)) by auto
    have validate-clause I (subst-cl (subst-cl ?C \sigma) \vartheta)
    using <validate-clause I (cl-ecl C)> instances-are-entailed by blast
    from this and assms(1) and assms(2) have validate-clause I (subst-cl ?D \vartheta)
    using ground-reflexion-is-sound unfolding clause-entails-clause-def
    using <fo-interpretation I〉<ground-clause (subst-cl (cl-ecl D) \vartheta)> by blast
    from this and D-false show False
        by (metis <ground-clause (subst-cl (cl-ecl D) \vartheta)>
        substs-preserve-ground-clause validate-clause.elims(1))
qed
lemma orient-lit-semantics-pos:
    assumes fo-interpretation I
    assumes orient-lit-inst l u v pos \eta
    assumes validate-ground-lit I (subst-lit l \sigma)
    shows I (subst u \sigma) (subst v \sigma)
proof -
    let ?u = subst u \sigma
    let ?v = subst v\sigma
        from assms(2) have l=(Pos (Equv))\veel=(Pos (Eqvu)) using ori-
ent-lit-inst-def by auto
    from this and assms(3) have validate-ground-eq I (Eq ?u ?v) \vee validate-ground-eq
I (Eq ?v ?u)
            by auto
    then have I ?u ?v v I ?v ?u by auto
    from this and <fo-interpretation I〉 show I ?u ?v
            unfolding fo-interpretation-def congruence-def equivalence-relation-def sym-
metric-def by blast
qed
lemma orient-lit-semantics-neg :
    assumes fo-interpretation I
    assumes orient-lit-inst l u v neg \vartheta
    assumes validate-ground-lit I (subst-lit l \sigma)
    shows }\negI(\mathrm{ subst }u\sigma)(\mathrm{ subst v }\sigma\mathrm{ )
proof -
    let ?u = subst }u
    let ?v = subst v\sigma
```

from $\operatorname{assms}(2)$ have $l=(N e g(E q u v)) \vee l=(N e g(E q v u)) \quad$ using ori－ ent－lit－inst－def by auto
from this and assms（3）have $\neg$ validate－ground－eq $I(E q ? u$ ？v）$\vee \neg$ vali－ date－ground－eq I（Eq ？v？u）
by auto
then have $\neg I$ ？$u$ ？$v \vee \neg I$ ？v ？$u$ by auto
from this and 〈fo－interpretation $I\rangle$ show $\neg I$ ？u ？v
unfolding fo－interpretation－def congruence－def equivalence－relation－def sym－
metric－def by blast
qed
lemma orient－lit－semantics－replacement ：
assumes fo－interpretation I
assumes orient－lit－inst l u v polarity $\vartheta$
assumes validate－ground－lit I（subst－lit l $\sigma$ ）
assumes $I$（subst $u \sigma$ ）（subst $u^{\prime} \sigma$ ）
shows validate－ground－lit I（subst－lit（mk－lit polarity（Equ＇v））$\sigma$ ）
proof－
from assms（2）obtain $e$ where $l=P o s e \vee l=N e g e$ and $e=E q u v \vee e=$ Eq v u
unfolding orient－lit－inst－def by auto
have polarity $=$ pos $\vee$ polarity $=$ neg using sign．exhaust by blast
then show ？thesis
proof
assume polarity $=$ pos
from this and $\operatorname{assms}(1)$ and assms（2）and＜validate－ground－lit I（subst－lit l
$\sigma)$ ）have
$I$（subst $u \sigma$ ）（subst $v \sigma$ ）using orient－lit－semantics－pos by auto
from this and assms（1）and $\left\langle I\right.$（subst $u \sigma$ ）（subst $u^{\prime} \sigma$ ）〉
have $I$（subst $u^{\prime} \sigma$ ）（subst $v \sigma$ ）unfolding fo－interpretation－def
congruence－def equivalence－relation－def symmetric－def transitive－def by blast
from this and $<$ polarity $=$ pos〉 show ？thesis by auto
next
assume polarity $=$ neg
from this and assms（1）and assms（2）and＜validate－ground－lit I（subst－lit l
$\sigma)$ ）have
$\neg I$（subst $u \sigma$ ）（subst $v \sigma$ ）using orient－lit－semantics－neg
by blast
from this and assms（1）and $\left\langle I\right.$（subst $u \sigma$ ）（subst $u^{\prime} \sigma$ ）〉
have $\neg I$（subst $u^{\prime} \sigma$ ）（subst $v \sigma$ ）unfolding fo－interpretation－def congruence－def equivalence－relation－def symmetric－def transitive－def by blast
from this and $<$ polarity $=$ neg〉 show ？thesis by auto
qed
qed
lemma ground－factorization－is－sound ：
assumes finite（cl－ecl C）
assumes factorization $C D \sigma k C^{\prime}$
assumes（ground－clause（subst－cl（cl－ecl D）$\vartheta)$ ）

```
    shows clause-entails-clause (subst-cl (subst-cl (cl-ecl C) \sigma) \vartheta)
    (subst-cl (cl-ecl D) \vartheta)
proof (rule ccontr)
    let ?C = (cl-ecl C)
    let ?D = (cl-ecl D)
    assume \neg clause-entails-clause (subst-cl (subst-cl (cl-ecl C) \sigma) \vartheta)
        (subst-cl (cl-ecl D) \vartheta)
    then obtain I where
        validate-clause I (subst-cl (subst-cl (cl-ecl C) \sigma) \vartheta) and
        \neg ( v a l i d a t e - c l a u s e ~ I ~ ( s u b s t - c l ~ ( c l - e c l ~ D ) ~ \vartheta ) ) ~ a n d ~ f o - i n t e r p r e t a t i o n ~ I ~
    unfolding clause-entails-clause-def by auto
    from assms(2) obtain L1 L2 L't s uv where
        orient-lit-inst L1 t s pos \sigma and orient-lit-inst L2 u v pos \sigma and ck-unifier t u
\sigmak
        and }\mp@subsup{L}{}{\prime}=Neg(Eqsv
        and (?D = (subst-cl ((?C - {L2 }) \cup{ L'} )) \sigma)
        and L1 = L2
        and L1 \in?C
    using factorization-def by auto
    from assms(1) have finite (subst-cl (subst-cl ?C \sigma) \vartheta) by auto
    then obtain }\eta\mathrm{ where i:ground-clause (subst-cl
        (subst-cl (subst-cl ?C \sigma) \vartheta) \eta)
    using ground-instance-exists [of (subst-cl (subst-cl ?C \sigma) \vartheta)]
    by auto
    let ?CC = (subst-cl (subst-cl (subst-cl ?C \sigma) \vartheta) \eta)
    let ?\sigma'|}=\operatorname{comp \sigma\vartheta
    let ? }\mp@subsup{\sigma}{}{\prime}=comp ? 棌 
    have ?CC= (subst-cl (subst-cl ?C ? }\mp@subsup{\sigma}{}{\prime\prime})\eta
    using composition-of-substs-cl [of ?C] by auto
    then have ?CC = (subst-cl ?C ? }\mp@subsup{\sigma}{}{\prime
    using composition-of-substs-cl [of ?C] by auto
from 〈validate-clause I (subst-cl (subst-cl (cl-ecl C) \sigma) \vartheta)>
    have validate-ground-clause I ?CC using i validate-clause.simps by blast
then obtain l' where l'}\mp@subsup{l}{}{\prime}\in?CC\mathrm{ and validate-ground-lit I l' by auto
from }\langle\mp@subsup{l}{}{\prime}\in?CC\rangle\mathrm{ and 〈?CC = (subst-cl ?C ? }\mp@subsup{\sigma}{}{\prime})\rangle\mathrm{ obtain l where
    l\in?C and l' = (subst-lit l ? }\mp@subsup{\sigma}{}{\prime})\mathrm{ using subst-cl.simps by blast
    from «\neg validate-clause I (subst-cl (cl-ecl D) \vartheta)〉
    have \neg validate-ground-clause I (subst-cl ?D \vartheta)
    using assms(3) substs-preserve-ground-clause validate-clause.elims(3) by metis
    from <ground-clause (subst-cl ?D \vartheta)> have
        (subst-cl ?D \vartheta) =(subst-cl (subst-cl ?D \vartheta) \eta)
    using substs-preserve-ground-clause [of (subst-cl ?D \vartheta) \eta] by blast
from this and «\neg validate-ground-clause I (subst-cl ?D \vartheta)`
    have \neg validate-ground-clause I (subst-cl (subst-cl ?D \vartheta) \eta) by force
from <(?D = (subst-cl ( (?C - {L2 }) \cup{ L'} )) \sigma)>
    have (subst-lit L'\sigma)\in?D by auto
then have
```

```
    (subst-lit (subst-lit (subst-lit \(\left.\left.\left.L^{\prime} \sigma\right) \vartheta\right) ~ \eta\right)\)
        \(\in(\) subst-cl (subst-cl ?D \() ~ \eta)\)
```

    by auto
    from this and «ᄀ validate-ground-clause I (subst-cl (subst-cl ?D ๆ) \(\eta\) )〉
    have \(\neg\) validate-ground-lit I (subst-lit (subst-lit (subst-lit \(\left.L^{\prime} \sigma\right) \vartheta\) ) \(\eta\) )
    by auto
    from this and $\left\langle L^{\prime}=N e g(E q s v)\right\rangle$ have
$I$ (subst (subst (subst s $\sigma$ ) $\vartheta$ ) $\eta$ )
(subst (subst (subst $v \sigma$ ) $\vartheta) \eta$ ) by auto
from this have $I$ (subst $s$ ? $\sigma^{\prime}$ ) (subst $v ? \sigma^{\prime}$ ) by simp
have subst-lit $l \sigma \in$ ? $D$
proof (rule ccontr)
assume subst-lit $l \sigma \notin ? D$
from this and $\left\langle\left(? D=\left(\right.\right.\right.$ subst-cl $\left.\left.\left.\left((? C-\{L 2\}) \cup\left\{L^{\prime}\right\}\right)\right) \sigma\right)\right\rangle$ and $\langle l \in$
?C>
have $l=L 2$ by auto
from 〈ck-unifier t $u \sigma \quad k\rangle$ have subst $t \sigma=$ subst $u \sigma$
using ck-unifier-thm by auto
then have subst (subst (subst $t \sigma) \vartheta) \eta=$
subst (subst (subst $u \sigma$ ) $\vartheta$ ) $\eta$ by auto
then have (subst $t ? \sigma^{\prime}$ ) $=$ subst $u$ ? $\sigma^{\prime}$ by auto
from $\left\langle\right.$ validate-ground-lit $\left.I l^{\prime}\right\rangle$ and $\left\langle l^{\prime}=\left(\right.\right.$ subst-lit $\left.\left.l ? \sigma^{\prime}\right)\right\rangle$ have
validate-ground-lit I (subst-lit l ? $\sigma^{\prime}$ ) by auto
from this and 〈fo-interpretation $I\rangle$ and $\langle l=L 2\rangle$ and $\langle$ orient-lit-inst L2 $u v$
pos $\sigma$ >
have $I$ (subst $u$ ? $\sigma^{\prime}$ ) (subst $v$ ? $\left.\sigma^{\prime}\right)$ using orient-lit-semantics-pos
by blast
from this and 〈fo-interpretation $I\rangle$ and $\left\langle I\right.$ (subst $\left.s ? \sigma^{\prime}\right)\left(\right.$ subst $\left.\left.v ? \sigma^{\prime}\right)\right\rangle$
have $I$ (subst $u$ ? $\sigma^{\prime}$ ) (subst $s$ ? $\sigma^{\prime}$ )
unfolding fo-interpretation-def congruence-def equivalence-relation-def
symmetric-def transitive-def by blast
from this and $\left\langle\left(\right.\right.$ subst $\left.t ? \sigma^{\prime}\right)=$ subst $\left.u ? \sigma^{\prime}\right\rangle$
have $I$ (subst $t ? \sigma^{\prime}$ ) (subst s? $\sigma^{\prime}$ ) by auto
from this have validate-ground-eq I (subst-equation (Eq ts) ? $\sigma^{\prime}$ )
by auto
from $\left\langle I\right.$ (subst $\left.t ? \sigma^{\prime}\right)\left(\right.$ subst $\left.\left.s ? \sigma^{\prime}\right)\right\rangle$ and $\langle f o$-interpretation $I\rangle$
have $I$ (subst $s ? \sigma^{\prime}$ ) (subst $t$ ? $\sigma^{\prime}$ )
unfolding fo-interpretation-def congruence-def equivalence-relation-def
symmetric-def by auto
from this have validate-ground-eq I (subst-equation (Eq st) ? $\sigma^{\prime}$ )
by auto
from <orient-lit-inst L1 tspos $\sigma\rangle$ have $L 1=(\operatorname{Pos}(E q t s)) \vee L 1=(\operatorname{Pos}(E q$
$s t)$
unfolding orient-lit-inst-def by auto

```
    from this and <validate-ground-eq I (subst-equation (Eq s t) ? }\mp@subsup{\sigma}{}{\prime}\mathrm{ )> and
        <validate-ground-eq I (subst-equation (Eq t s) ? \sigma')>
        have validate-ground-lit I (subst-lit L1 ?\sigma')
    by auto
    from}\langleL1\in?C\rangle\mathrm{ and }\langle?D=(subst-cl ((?C-{L2 })\cup{\mp@subsup{L}{}{\prime}}))\sigma\rangle\mathrm{ and
<L1 = L2`>
    have (subst-lit L1 \sigma)\in?D
    by auto
    then have
    (subst-lit (subst-lit (subst-lit L1 \sigma) \vartheta) \eta)
        (subst-cl (subst-cl ?D \vartheta) \eta) by auto
    then have (subst-lit L1 ? }\mp@subsup{\sigma}{}{\prime})\in(\mathrm{ subst-cl (subst-cl ?D ७) ף)
        using composition-of-substs-lit by metis
    from this and <validate-ground-lit I (subst-lit L1 ? \sigma')> and
        \imath validate-ground-clause I (subst-cl (subst-cl ?D \vartheta) \eta)>
        show False by auto
    qed
    from <subst-lit l \sigma & ?D> and <l' = subst-lit l ? }\mp@subsup{\sigma}{}{\prime}
    have l' }\mp@subsup{l}{}{\prime}\mathrm{ (subst-cl (subst-cl ?D Э) ף)
    using subst-cl.simps composition-of-substs-lit mem-Collect-eq
    by (metis (mono-tags, lifting))
    from this and <validate-ground-lit I l'> have
        validate-ground-clause I (subst-cl (subst-cl ?D \vartheta) \eta) by auto
    from this and «\neg validate-ground-clause I (subst-cl (subst-cl ?D \vartheta) \eta)>
    show False by blast
qed
lemma factorization-is-sound :
    assumes finite (cl-ecl C)
    assumes factorization CD \sigmak C'
    shows clause-entails-clause (cl-ecl C) (cl-ecl D)
proof (rule ccontr)
    let ?C = (cl-ecl C)
    let ?D = (cl-ecl D)
    assume }\neg(\mathrm{ clause-entails-clause ?C ?D)
    then obtain I where validate-clause I ?C and }\neg(validate-clause I ?D) fo-interpretatio
I
    unfolding clause-entails-clause-def by auto
    from <\neg (validate-clause I ?D)\rangle obtain \vartheta
    where D-false: \neg(validate-ground-clause I (subst-cl ?D \vartheta))
        and (ground-clause (subst-cl ?D \vartheta)) by auto
    have validate-clause I (subst-cl (subst-cl ?C \sigma) \vartheta)
    using <validate-clause I (cl-ecl C)> instances-are-entailed by blast
    from this and assms(1) and assms(2) have validate-clause I (subst-cl ?D \vartheta)
        using ground-factorization-is-sound unfolding clause-entails-clause-def
    using <fo-interpretation I〉 <ground-clause (subst-cl (cl-ecl D) \vartheta)〉 by blast
    from this and D-false show False
    by (metis <ground-clause (subst-cl (cl-ecl D) \vartheta)>
```

```
        substs-preserve-ground-clause validate-clause.elims(1))
qed
lemma ground-superposition-is-sound :
    assumes finite (cl-ecl P1)
    assumes finite (cl-ecl P2)
    assumes superposition P1 P2 C \sigmak C'
    assumes (ground-clause (subst-cl (cl-ecl C) \vartheta))
    shows set-entails-clause
    { (subst-cl (subst-cl (cl-ecl P1) \sigma) \vartheta),
    (subst-cl (subst-cl (cl-ecl P2) \sigma) \vartheta) }
            (subst-cl (cl-ecl C) \vartheta)
proof (rule ccontr)
    let ?P1 = (cl-ecl P1)
    let ?P2 = (cl-ecl P2)
    let ?C = (cl-ecl C)
    assume ᄀ set-entails-clause
        {(subst-cl (subst-cl (cl-ecl P1)\sigma)\vartheta),
            (subst-cl (subst-cl (cl-ecl P2) \sigma) \vartheta) }
            (subst-cl (cl-ecl C) \vartheta)
    then obtain I
        where validate-clause I (subst-cl (subst-cl (cl-ecl P1) \sigma) \vartheta)
        and validate-clause I (subst-cl (subst-cl (cl-ecl P2) \sigma) \vartheta)
            and \neg(validate-clause I (subst-cl (cl-ecl C) \vartheta)) and fo-interpretation I
    unfolding set-entails-clause-def by (meson insert-iff validate-clause-set.elims(2))
    from assms(3) obtain ts u v M p polarity t' u'L L' where
        orient-lit-inst M u v pos \sigma
        and orient-lit-inst L t s polarity \sigma
        and subterm t p u'
        and ck-unifier u' u \sigma k
        and replace-subterm t p v t'
        and }\mp@subsup{L}{}{\prime}=mk\mathrm{ -lit polarity (Eq t' s)
        and ?C = (subst-cl ((?P1 - {L })\cup((?P2 - {M })\cup{ L'} ))\sigma)
    using superposition-def by auto
    let ?PP1'=(subst-cl (subst-cl ?P1 \sigma)\vartheta)
    let ?P2' = (subst-cl (subst-cl ?P2 \sigma) \vartheta)
    from assms(1) have finite ?P1' by simp
    from assms(2) have finite ?P2' by simp
    let ?vars = (vars-of-cl ?P1') \cup (vars-of-cl ?P2')
    from〈finite ?P1'〉 have finite (vars-of-cl ?P1')
        using set-of-variables-is-finite-cl [of ?P1 ] by auto
    from〈finite ?P2'> have finite (vars-of-cl ?P2')
        using set-of-variables-is-finite-cl [of ?P2'] by auto
    from 〈finite (vars-of-cl ?P1')〉 and 〈finite (vars-of-cl ?P2')> have finite ?vars
by auto
    then obtain }\eta\mathrm{ where ground-on ?vars }\eta\mathrm{ using ground-subst-exists by blast
```

then have ground－on（vars－of－cl ？P1＇）$\eta$ unfolding ground－on－def by auto
then have ground－clause（subst－cl
（subst－cl（subst－cl ？P1 $\sigma$ ）$\vartheta) ~ \eta) ~$
using ground－substs－yield－ground－clause
$[$ of（subst－cl（subst－cl ？P1 $\sigma$ ）$\vartheta$ ）$\eta$ ］by auto
from 〈ground－on ？vars $\eta\rangle$ have ground－on（vars－of－cl ？P2＇）$\eta$ unfolding ground－on－def by auto
then have ground－clause（subst－cl

$$
(\text { subst-cl }(\text { subst-cl ?P2 } \sigma) \vartheta) \eta)
$$

using ground－substs－yield－ground－clause
$[o f($ subst－cl（subst－cl ？P2 $\sigma$ ）$\vartheta) ~ \eta]$ by auto
let $? P 1^{\prime \prime}=\left(\right.$ subst $\left.-c l ? P 1^{\prime} \eta\right)$
let $? P 2^{\prime \prime}=\left(\right.$ subst－cl ？P2 $\left.{ }^{\prime} \eta\right)$
let $? \sigma^{\prime \prime}=\operatorname{comp} \sigma \vartheta$
let ？$\sigma^{\prime}=\mathrm{comp}$ ？$\sigma^{\prime \prime} \eta$
have ？P1＂$=\left(\right.$ subst－cl $\left(\right.$ subst－cl ？P1 ？$\left.\left.\sigma^{\prime \prime}\right) \eta\right)$
using composition－of－substs－cl［of ？P1］by auto
then have ？$P 1^{\prime \prime}=\left(\right.$ subst－cl ？$\left.P 1 ? \sigma^{\prime}\right)$
using composition－of－substs－cl［of ？P1］by auto
from 〈ground－clause（subst－cl（subst－cl（subst－cl（cl－ecl P1）$\sigma$ ）$\vartheta) ~ \eta) 〉$
and «validate－clause I（subst－cl（subst－cl（cl－ecl P1）$\sigma$ ）$\vartheta$ ）〉
have validate－ground－clause I ？P1＂using validate－clause．simps by blast
then obtain $l 1^{\prime}$ where $l 1^{\prime} \in ? P 1^{\prime \prime}$ and validate－ground－lit I $l 1^{\prime}$ by auto
have ？$P^{\prime \prime}{ }^{\prime \prime}=\left(\right.$ subst－cl（subst－cl ？P2 ？$\left.\sigma^{\prime \prime}\right) \eta$ ）
using composition－of－substs－cl［of ？P2］by auto
then have ${ }^{2} \mathrm{P}^{\prime \prime}=\left(\right.$ subst－cl ？P2 ？$\left.\sigma^{\prime}\right)$
using composition－of－substs－cl［of ？P2］by auto
from 〈ground－clause（subst－cl（subst－cl（subst－cl（cl－ecl P2）$\sigma$ ）$\vartheta) ~ \eta)\rangle\langle v a l i-$ date－clause I（subst－cl（subst－cl（cl－ecl P2）$\sigma$ ）$\vartheta$ ）＞
have validate－ground－clause I ？P2＇＂using validate－clause．simps by blast
then obtain $l 2^{\prime}$ where $l 2^{\prime} \in ? P 2^{\prime \prime}$ and validate－ground－lit I $l 2^{\prime}$ by auto
from $\left\langle l 1^{\prime} \in ? P 1^{\prime \prime}\right\rangle$ and $\left\langle ? P 1^{\prime \prime}=\left(\right.\right.$ subst－cl ？PP1？$\left.\left.\sigma^{\prime}\right)\right\rangle$ obtain $l 1$ where
$l 1 \in ? P 1$ and $l 1^{\prime}=\left(\right.$ subst－lit $\left.l 1 ? \sigma^{\prime}\right)$ using subst－cl．simps by blast
from 〈l2＇$\left.\in ? P^{2}{ }^{\prime \prime}\right\rangle$ and $\left\langle ? P 2^{\prime \prime}=\left(\right.\right.$ subst－cl ？P2 ？$\left.\left.\sigma^{\prime}\right)\right\rangle$ obtain 12 where
$l 2 \in ? P 2$ and $l 2^{\prime}=\left(\right.$ subst－lit l2 ？$\left.\sigma^{\prime}\right)$ using subst－cl．simps by blast
let ${ }^{2} C^{\prime}=($ subst－cl（subst－cl ？C $\left.\vartheta) \eta\right)$
from 〈ground－clause（subst－cl ？C $\vartheta$ ）〉 have
$($ subst－cl ？C $\vartheta)=($ subst－cl（subst－cl ？C $\vartheta) \eta)$
using substs－preserve－ground－clause［of（subst－cl ？C $\vartheta$ ）$\eta$ ］by blast
from 〈ᄀ validate－clause I（subst－cl（cl－ecl C）$\vartheta$ ）〉
have $\neg$ validate－ground－clause $I$ ？$C^{\prime}$
by（metis assms（4）substs－preserve－ground－clause validate－clause．simps）
have $l 1=L$
proof（rule ccontr）

```
    assume l1 \not=L
    from this and <l1 \in?P1> and <?C = (subst-cl ((?P1 - {L }) \cup((?P2 - {
M }) \cup{ L'' } )) \sigma)>
    have (subst-lit l1 \sigma) \in?C by auto
    from this have (subst-lit (subst-lit (subst-lit l1 \sigma) \vartheta) \eta)
        G?C' by auto
    from this and <l1' = (subst-lit l1 ? \sigma')\rangle have l1' \in? 'C'
        by (simp add: composition-of-substs-lit)
    from this and <validate-ground-lit I l1`` have validate-ground-clause I ?C' by
auto
    from this and «\neg validate-ground-clause I (subst-cl (subst-cl (cl-ecl C) \vartheta) \eta)>
        show False by auto
    qed
    have l2 = M
    proof (rule ccontr)
    assume l2 f= M
    from this and <l2 \in?P2> and <?C = (subst-cl ((?P1 - {L }) \cup ((?P2 - {
M }) \cup{ L' ( ) ) \sigma)>
    have (subst-lit l2 \sigma) \in?C by auto
    from this have (subst-lit (subst-lit (subst-lit l2 \sigma) \vartheta) \eta)
        G?C' by auto
    from this and <l2' = (subst-lit l2 ? }\mp@subsup{\sigma}{}{\prime})\rangle\mathrm{ have l2' }\in
        by (simp add: composition-of-substs-lit)
    from this and <validate-ground-lit I l2'` have validate-ground-clause I ?C' by
auto
    from this and «\neg validate-ground-clause I (subst-cl (subst-cl (cl-ecl C) \vartheta) \eta)〉
        show False by auto
    qed
    from <orient-lit-inst M u v pos \sigma〉 and 〈l2 = M〉 and <fo-interpretation I\rangle
    and <validate-ground-lit I l2'> and <l2' = (subst-lit l2 ? }\mp@subsup{\sigma}{}{\prime})
    have I (subst u ? }\mp@subsup{\sigma}{}{\prime}\mathrm{ ) (subst v ? }\mp@subsup{\sigma}{}{\prime}
    using orient-lit-semantics-pos by blast
from <subterm t p u'〉 have
    subterm (subst t ? }\mp@subsup{\sigma}{}{\prime}\mathrm{ ) p (subst u' ? }\mp@subsup{\sigma}{}{\prime}\mathrm{ )
        using substs-preserve-subterms [of t p u ] by metis
    from <ck-unifier }\mp@subsup{u}{}{\prime}u\sigmak\rangle\mathrm{ have (subst u }\sigma\mathrm{ ) = (subst u' }\sigma\mathrm{ )
    using ck-unifier-thm [of u' u \sigma k] by auto
    from this have (subst (subst (subst u \sigma) \vartheta) \eta)
    =(subst (subst (subst u'\sigma)\vartheta ) \eta) by auto
from this have (subst u ? }\mp@subsup{\sigma}{}{\prime}\mathrm{ ) = (subst u' ? }\mp@subsup{\sigma}{}{\prime}
    using composition-of-substs by auto
from <(subst u ? }\mp@subsup{\sigma}{}{\prime})=(\mathrm{ subst }\mp@subsup{u}{}{\prime}?\mp@subsup{\sigma}{}{\prime})
```



```
    have I (subst u' ?}\mp@subsup{\sigma}{}{\prime})(\mathrm{ subst v ? }\mp@subsup{\sigma}{}{\prime}\mathrm{ )
```

```
    by auto
from <subterm t p u'>
    and <I (subst u' ? }\mp@subsup{\sigma}{}{\prime}\mathrm{ ) (subst v ? }\mp@subsup{\sigma}{}{\prime}\mathrm{ )>
    and 〈fo-interpretation I\rangle
    and 〈replace-subterm t p v t'>
    have I (subst t ? \sigma') (subst t' ? }\mp@subsup{\sigma}{}{\prime}\mathrm{ )
        unfolding fo-interpretation-def using replacement-preserves-congruences [of
I u
    by auto
```

    from \(\langle l 1=L\rangle\) and \(\langle f o\)-interpretation \(I\rangle\) and \(\langle\) validate-ground-lit \(I l 1\rangle\)
    and \(\left\langle l 1^{\prime}=\left(\right.\right.\) subst-lit \(\left.\left.l 1 ? \sigma^{\prime}\right)\right\rangle\)
    and <orient-lit-inst \(L t\) s polarity \(\sigma\rangle\)
    and \(\left\langle I\right.\) (subst \(\left.t ? \sigma^{\prime}\right)\left(\right.\) subst \(\left.\left.t^{\prime} ? \sigma^{\prime}\right)\right\rangle\)
    and \(\left\langle L^{\prime}=m k\right.\)-lit polarity \(\left.\left(E q t^{\prime} s\right)\right\rangle\)
    have validate-ground-lit I (subst-lit \(L^{\prime}\) ? \(\sigma^{\prime}\) )
    using orient-lit-semantics-replacement [of I Ltspolarity \(\sigma\) ? \(\sigma^{\prime} t\) ] by blast
    from \(\left\langle ? C=\left(\right.\right.\) subst-cl \(\left.\left.\left((? P 1-\{L\}) \cup\left((? P 2-\{M\}) \cup\left\{L^{\prime}\right\}\right)\right) \sigma\right)\right\rangle\)
    have subst-lit \(L^{\prime} \sigma \in\) ? \(C\) by auto
    then have subst-lit (subst-lit (subst-lit \(L^{\prime} \sigma\) ) \(\left.\vartheta\right) \eta \in\) ? \(C^{\prime}\)
    by auto
    then have subst-lit \(L^{\prime} ? \sigma^{\prime} \in ? C^{\prime}\) by (simp add: composition-of-substs-lit)
    from this and \(\left\langle\right.\) validate-ground-lit I (subst-lit \(\left.\left.L^{\prime} ? \sigma^{\prime}\right)\right\rangle\) and \(\langle\neg\) validate-ground-clause
    $I ? C^{\prime}>$
show False by auto
qed
lemma superposition-is-sound:
assumes finite (cl-ecl P1)
assumes finite (cl-ecl P2)
assumes superposition P1 P2 $C \sigma k C^{\prime}$
shows set-entails-clause \{ cl-ecl P1, cl-ecl P2 \} (cl-ecl C)
proof (rule ccontr)
let ? P1 $=($ cl-ecl P1)
let ? P2 $=($ cl-ecl P2)
let ? $C=(c l-e c l C)$
assume $\neg$ set-entails-clause $\{$ cl-ecl P1, cl-ecl P2 \} (cl-ecl C)
then obtain $I$
where validate-clause I?P1 and validate-clause I ?P2
and $\neg$ (validate-clause I ?C) and fo-interpretation I
unfolding set-entails-clause-def by (meson insert-iff validate-clause-set.elims(2))
from 〈ᄀ (validate-clause I ?C)〉 obtain $\vartheta$
where $\neg$ (validate-ground-clause I (subst-cl ?C $\vartheta)$ )
and (ground-clause (subst-cl ?C $\vartheta$ )) by auto
have P1-true: validate-clause I (subst-cl (subst-cl ?P1 $\sigma$ ) $\vartheta$ )
using 〈validate-clause I (cl-ecl P1)〉 instances-are-entailed by blast

```
    have P2-true: validate-clause I (subst-cl (subst-cl ?P2 \sigma) \vartheta)
    using <validate-clause I (cl-ecl P2)> instances-are-entailed by blast
    have \neg (validate-clause I (subst-cl ?C \vartheta))
    by (metis «\neg validate-ground-clause I (subst-cl (cl-ecl C) \vartheta)`
            \ground-clause (subst-cl (cl-ecl C) \vartheta)>
            substs-preserve-ground-clause validate-clause.elims(1))
    let ?S = {(subst-cl (subst-cl (cl-ecl P1)\sigma) \vartheta),
        (subst-cl (subst-cl (cl-ecl P2) \sigma) \vartheta) }
    from P1-true and P2-true have validate-clause-set I ?S
    by (metis insert-iff singletonD validate-clause-set.elims(3))
    from this and <\neg (validate-clause I (subst-cl ?C \vartheta))\rangle\langlefo-interpretation I〉
        have \neg set-entails-clause ?S (subst-cl (cl-ecl C) \vartheta)
        using set-entails-clause-def by blast
    from this and assms(1) and assms(2) and assms(3) and
        <(ground-clause (subst-cl ?C \vartheta))>
    show False using ground-superposition-is-sound by auto
qed
lemma superposition-preserves-finiteness:
    assumes finite (cl-ecl P1)
    assumes finite (cl-ecl P2)
    assumes superposition P1 P2 C \sigma k C'
    shows finite (cl-ecl C)}\wedge(\mathrm{ finite C')
proof -
    from assms(3) obtain LM L' where
        def-C:(cl-ecl C) = (subst-cl (((cl-ecl P1) - {L }) \cup(((cl-ecl P2) - {M })\cup
{ L'} )) \sigma)
            and def-C': C'}=(((cl-ecl P1) - {L })\cup(((cl-ecl P2) - {M })\cup{ L'} ))
            using superposition-def by auto
    from assms(1) and assms(2) have finite (((cl-ecl P1) - {L}) \cup(((cl-ecl P2)
- {M })\cup{ L'} ))
            by auto
    from this and def-C def-C' show ?thesis using substs-preserve-finiteness by
auto
qed
lemma reflexion-preserves-finiteness:
    assumes finite (cl-ecl P1)
    assumes reflexion P1 C\sigma k C'
    shows finite (cl-ecl C)}\wedge(finite C'
proof -
    from assms(2) obtain L1 where
            def-C:(cl-ecl C) =(subst-cl ((cl-ecl P1) - {L1 }) \sigma)
            and def-C':}\mp@subsup{C}{}{\prime}=((cl-ecl P1) - {L1 })
            using reflexion-def by auto
    from assms(1) have finite ((cl-ecl P1) - {L1 }) by auto
    from this and def-C def-C' show ?thesis using substs-preserve-finiteness by
auto
qed
```

```
lemma factorization-preserves-finiteness:
    assumes finite (cl-ecl P1)
    assumes factorization P1 C \sigmak C'
    shows finite (cl-ecl C)}\wedge(finite C'
proof -
    from assms(2) obtain L2 L' where
        def-C:(cl-ecl C) =(subst-cl (((cl-ecl P1) - {L2 }) \cup{ L'} ) \sigma)
        and def-C': C' = (((cl-ecl P1) - {L2 }) \cup{ L'} )
        using factorization-def by auto
    from assms(1) have (finite (((cl-ecl P1) - {L2 }) \cup { L'})) by auto
    from this and def-C def-C' show ?thesis using substs-preserve-finiteness by
auto
qed
lemma derivable-clauses-are-finite:
    assumes derivable C PS\sigmak C'
    assumes }\forallx\inP.(finite (cl-ecl x))
    shows finite (cl-ecl C)}\wedge(\mathrm{ finite C')
proof (rule ccontr)
    assume hyp:\neg(finite (cl-ecl C) ^(finite C'))
    have not-sup: \neg(\existsP1 P2. (P1 \inP\wedge P2 \inP\wedge superposition P1 P2 C \sigma k C'))
    proof
        assume ( }\exists\mathrm{ P1 P2. (P1 G P^P2 G P ^ superposition P1 P2 C % k C'))
        then obtain P1 P2 where P1 \inPP2 }\inP\mathrm{ superposition P1 P2 C }\sigmak\mp@subsup{C}{}{\prime}\mathrm{ by
auto
    from }\langleP1\inP\rangle\mathrm{ and assms(2) have finite (cl-ecl P1) by auto
    from }\langleP2\mathcal{Z}\inP\rangle\mathrm{ and assms(2) have finite (cl-ecl P2) by auto
    from «(finite (cl-ecl P1))\rangle and «(finite (cl-ecl P2))〉 and «superposition P1 P2
C\sigmak C'>
            have finite (cl-ecl C) ^(finite C') using superposition-preserves-finiteness
[of P1 P2 C \sigma] by auto
    then show False using hyp by auto
    qed
    have not-ref: \neg(\existsP1.(P1 \inP\wedge reflexion P1 C\sigma k C'))
    proof
    assume ( }\exists\mathrm{ P1. (P1 G P ^ reflexion P1 C ok C'))
    then obtain P1 where P1 \inP reflexion P1 C\sigmak 乍 by auto
    from }\langleP1\inP\rangle\mathrm{ and assms(2) have finite (cl-ecl P1) by auto
    from <(finite (cl-ecl P1))\rangle and <reflexion P1 C \sigmak C'>
            have finite (cl-ecl C) ^(finite C') using reflexion-preserves-finiteness [of P1
C \sigma] by auto
    then show False using hyp by auto
    qed
    have not-fact: \neg(\existsP1.(P1 \inP\wedge factorization P1 C \sigmak 鱼))
    proof
    assume ( }\exists\textrm{P}1.(P1\inP\wedge factorization P1 C \sigma k C '))
    then obtain P1 where P1 \inP factorization P1 C \sigmak C' by auto
    from }\langleP1\inP\rangle\mathrm{ and assms(2) have finite (cl-ecl P1) by auto
```

```
    from <(finite (cl-ecl P1))> and < factorization P1 C \sigma k C'>
            have finite (cl-ecl C) ^(finite C') using factorization-preserves-finiteness
[of P1 C \sigma] by auto
    then show False using hyp by auto
    qed
    from not-sup not-ref not-fact and assms(1) show False unfolding derivable-def
by blast
qed
lemma derivable-clauses-lemma:
    assumes derivable C PS\sigmak C'
    shows }((\mathrm{ cl-ecl C) }=(\mathrm{ subst-cl C' }\sigma)
proof (rule ccontr)
    assume hyp:\neg ((cl-ecl C) =(subst-cl C'\sigma))
    have not-sup: \neg(\existsP1 P2. (P1 GS^P2 G S^ superposition P1 P2 C \sigmak C'))
    proof
    assume ( }\exists\mathrm{ P1 P2. (P1 G S ^P2 G S ^ superposition P1 P2 C б k C'))
    then obtain P1 P2 where P1 \inS P2 \inS superposition P1 P2 C \sigmak C' by
auto
    from <superposition P1 P2 C \sigma k C'` obtain Cl-C Cl-P1 L Cl-P2 M L' T
            where Cl-C = (subst-cl ((Cl-P1 - {L }) \cup((Cl-P2 - {M }) \cup{ L'} )) \sigma)
                (C'=(Cl-P1 - {L }) \cup((Cl-P2 - {M }) \cup{ L'} ))
                C=(Ecl Cl-C T)
                unfolding superposition-def by blast
    from <Cl-C = (subst-cl ((Cl-P1 - {L }) \cup ((Cl-P2 - {M }) \cup{ L'} )) \sigma)>
                <(C'}=(Cl-P1 - {L })\cup((Cl-P2 - {M })\cup{ L'} ))\rangle\langleC=(Ecl Cl-C
T)> hyp show False by auto
    qed
    have not-ref: \neg(\existsP1.(P1 \inS ^ reflexion P1 C\sigmak C'))
    proof
    assume ( }\exists\mathrm{ P1. (P1 G S ^ reflexion P1 C %k C'))
    then obtain P1 where P1 \inS reflexion P1 C \sigmak C' by auto
    from <reflexion P1 C \sigma k C'`}\mathrm{ obtain T Cl-C Cl-P L1 where
        C=(Ecl Cl-C T)
        Cl-C=(subst-cl ((Cl-P - {L1}) )) \sigma
        (C'=((Cl-P - {L1 }) )) unfolding reflexion-def by blast
    from <Cl-C = (subst-cl ((Cl-P - {L1 }) )) \sigma\rangle
        <(C'=((Cl-P - {L1 }) ))\rangle\langleC=(Ecl Cl-C T)\rangle hyp show False by auto
    qed
    have not-fact: }\neg(\existsP1.(P1\inS\wedge factorization P1 C \sigmak C')
proof
    assume ( }\exists\mathrm{ P1. (P1 G S ^ factorization P1 C 
    then obtain P1 where P1 \inS factorization P1 C \sigmak C' by auto
    from〈factorization P1 C \sigma k C'夕 obtain T Cl-C Cl-P L' L2 where
        C=(Ecl Cl-C T)
        Cl-C=(subst-cl ( (Cl-P - {L2 }) \cup{ L'} )) \sigma
        C'}=((Cl-P-{L2 })\cup{\mp@subsup{L}{}{\prime}})\mathrm{ unfolding factorization-def by blast
    from 〈Cl-C = (subst-cl ((Cl-P - {L2 }) \cup{ L'} )) \sigma`
        〈C'}=((Cl-P-{L2 })\cup{\mp@subsup{L}{}{\prime}})\rangle\langleC=(Ecl Cl-C T)\rangle hyp show False by
```

```
auto
    qed
    from not-sup not-ref not-fact and assms(1) show False unfolding derivable-def
by blast
qed
lemma substs-preserves-decompose-literal:
    assumes decompose-literal L t s polarity
    shows decompose-literal (subst-lit L \eta) (subst t \eta) (subst s \eta) polarity
proof -
    let ?L = (subst-lit L \eta)
    let ?t = (subst t \eta)
    let ?s =(subst s \eta)
    have polarity = pos \vee polarity = neg using sign.exhaust by auto
    then show ?thesis
    proof
        assume polarity = pos
        from this and assms(1) have L=Pos (Eqt s)\veeL=Pos (Eqst)
            unfolding decompose-literal-def decompose-equation-def by auto
            from}\langleL=Pos(Eqts)\veeL=Pos (Eqst)
                have ?L = Pos (Eq ?t ?s) \vee ?L = Pos (Eq ?s ?t) by auto
    from this <polarity = pos` show ?thesis unfolding decompose-literal-def
                decompose-equation-def by auto
    next
        assume polarity = neg
        from this and assms(1) have L=Neg (Eq t s)\veeL=Neg (Eq s t)
            unfolding decompose-literal-def decompose-equation-def by auto
    from this «polarity = neg` show ?thesis unfolding decompose-literal-def
                decompose-equation-def by auto
    qed
qed
lemma substs-preserve-dom-trm:
    assumes dom-trm t C
    shows dom-trm (subst t \sigma) (subst-cl C \sigma)
proof -
    let ?t = (subst t \sigma)
    from assms(1) have ( }\exists\textrm{L}\mathrm{ uv p. (L GC^(decompose-literal L uvp)
                \wedge (((p=neg \wedget=u)\vee (t,u)\intrm-ord )))) unfolding dom-trm-def by
auto
    from this obtain Luvp where L\inC
        decompose-literal Luvp((( 
        unfolding dom-trm-def by blast
    let ?u = (subst u \sigma)
    from}\langleL\inC\rangle\mathrm{ have (subst-lit L }\sigma)\in(\mathrm{ subst-cl C %) by auto
    from <decompose-literal L u v p〉
        have decompose-literal (subst-lit L \sigma) (subst u \sigma) (subst v \sigma) p
```

```
    using substs-preserves-decompose-literal by metis
    from <(( ( p=neg \wedget=u)\vee (t,u) \in trm-ord ) )>
        have}(((p=neg\wedge?t=?u)\vee(?t,?u)\in\mathrm{ trm-ord })
        using trm-ord-subst by auto
    from this «(subst-lit L \sigma) \in(subst-cl C \sigma)>
    <decompose-literal (subst-lit L \sigma) (subst u \sigma) (subst v \sigma) p>
    show dom-trm (subst t \sigma) (subst-cl C \sigma)
    unfolding dom-trm-def by auto
qed
lemma substs-preserve-well-constrainedness:
    assumes well-constrained C
    shows well-constrained (subst-ecl C \sigma)
proof (rule ccontr)
    assume \neg?thesis
    from this obtain y where y\intrms-ecl (subst-ecl C \sigma)
        and \negdom-trm y (cl-ecl (subst-ecl C \sigma)) unfolding well-constrained-def by
auto
    obtain Cl-C T where C=(Ecl Cl-C T) using eclause.exhaust by auto
    from this have (subst-ecl C \sigma)
        =(Ecl (subst-cl Cl-C \sigma) (subst-set T \sigma)) by auto
    from this have (cl-ecl (subst-ecl C \sigma) = (subst-cl Cl-C \sigma))
        and trms-ecl (subst-ecl C \sigma) = (subst-set T \sigma)
        by auto
    from «(cl-ecl (subst-ecl C \sigma) = (subst-cl Cl-C \sigma))\rangle
        <C=(Ecl Cl-C T)> have (cl-ecl (subst-ecl C \sigma) = (subst-cl (cl-ecl C) \sigma)) by
auto
    from \y\in trms-ecl (subst-ecl C \sigma)\rangle\langleC=(Ecl Cl-C T)>
        obtain z where z\inT and y=(subst z\sigma) by auto
    from}\langlez\inT\rangle\operatorname{assms}(1)\langleC=(Ecl Cl-CT)\rangle have dom-trm z (cl-ecl C)
        unfolding well-constrained-def by auto
    from this have dom-trm (subst z \sigma) (subst-cl (cl-ecl C) \sigma)
        using substs-preserve-dom-trm by auto
    from this «y = (subst z \sigma)` have dom-trm y (subst-cl (cl-ecl C) \sigma)
        by auto
    from this «(cl-ecl (subst-ecl C \sigma) = (subst-cl (cl-ecl C) \sigma))>
        \curlywedge dom-trm y (cl-ecl (subst-ecl C \sigma))> show False by auto
qed
lemma ck-trms-sound:
    assumes T= get-trms D (dom-trms C E)k
    shows T\subseteq(dom-trms C E)
proof (cases)
    assume k = FirstOrder
    from this and assms have T= filter-trms D (dom-trms C E)
        unfolding get-trms-def by auto
    from this show ?thesis using filter-trms-inclusion by blast
next
    assume k\not= FirstOrder
```

```
    from this and assms have T=(dom-trms C E)
    unfolding get-trms-def by auto
    from this show ?thesis using filter-trms-inclusion by blast
qed
lemma derivable-clauses-are-well-constrained:
    assumes derivable CPS\sigmak C'
    shows well-constrained C
proof (rule ccontr)
    assume hyp: \neg well-constrained C
    then obtain y where y\intrms-ecl C and \negdom-trm y (cl-ecl C)
    unfolding well-constrained-def by auto
```



```
    proof
    assume ( }\exists\mathrm{ P1 P2. (P1 G S ^ P2 }\inS\wedge superposition P1 P2 C \sigma k C'))
    then obtain P1 P2 where P1 \inS P2 GS superposition P1 P2 C \sigma k C' by
auto
    from<superposition P1 P2 C \sigma k C'` obtain Cl-C T E
            where
                T=(get-trms Cl-C (dom-trms Cl-C (subst-set E \sigma)) k)
                Cl-C=(subst-cl C' \sigma)
                C=(Ecl Cl-C T)
                unfolding superposition-def by blast
    from }\langleT=(\mathrm{ get-trms Cl-C (dom-trms Cl-C (subst-set E O)) k)>
            have T\subseteq(dom-trms Cl-C (subst-set E \sigma))
            using ck-trms-sound by metis
    from this and }\langley\intrms-ecl C\rangle and \langleC=(Ecl Cl-C T)\rangle hav
                y}\in(\mathrm{ dom-trms (cl-ecl C) (subst-set E F)) by auto
    from this and «\neg dom-trm y (cl-ecl C)\rangle show False unfolding dom-trms-def
by auto
    qed
    have not-ref: ᄀ(\existsP1. (P1 \inS ^ reflexion P1 C \sigma k C'})
    proof
        assume ( }\exists\mathrm{ P1. (P1 G S ^ reflexion P1 C % k C'))
        then obtain P1 where P1 \inS reflexion P1 C\sigmak C' by auto
    from 〈reflexion P1 C \sigma k C'` obtain T Cl-C E where
                T=(get-trms Cl-C (dom-trms Cl-C (subst-set E \sigma)) k)
                Cl-C=(subst-cl C' \sigma)
                C=(Ecl Cl-C T)
        unfolding reflexion-def by blast
    from <T = (get-trms Cl-C (dom-trms Cl-C (subst-set E \sigma)) k)>
            have T\subseteq(dom-trms Cl-C (subst-set E \sigma))
            using ck-trms-sound by metis
    from this and }\langley\in\mathrm{ trms-ecl C〉 and }\langleC=(Ecl Cl-C T)\rangle hav
        y\in(dom-trms (cl-ecl C) (subst-set E \sigma)) by auto
    from this and «\neg dom-trm y (cl-ecl C)〉 show False unfolding dom-trms-def
by auto
    qed
    have not-fact: \neg(\existsP1.(P1 \inS ^ factorization P1 C \sigmak C'))
```

```
proof
    assume ( }\exists\textrm{P}1.(P1\inS\wedge\mathrm{ factorization P1 C ok C
    then obtain P1 where P1 \inS factorization P1 C \sigmak C' by auto
    from〈factorization P1 C \sigmak C'夕 obtain T Cl-C E where
                T=(get-trms Cl-C (dom-trms Cl-C (subst-set E \sigma)) k)
                Cl-C = (subst-cl C'\sigma)
                C=(Ecl Cl-C T)
            unfolding factorization-def by blast
    from\T = (get-trms Cl-C (dom-trms Cl-C (subst-set E \sigma)) k)\rangle
            have T\subseteq(dom-trms Cl-C (subst-set E \sigma))
            using ck-trms-sound by metis
    from this and }\langley\intrms-ecl C\rangle and \langleC=(Ecl Cl-C T)\rangle hav
        y}\in(\mathrm{ dom-trms (cl-ecl C) (subst-set E O)) by auto
    from this and «\neg dom-trm y (cl-ecl C)\rangle show False unfolding dom-trms-def
by auto
    qed
    from not-sup not-ref not-fact and assms(1) show False unfolding derivable-def
by blast
qed
lemma derivable-clauses-are-entailed:
    assumes derivable C P S \sigma k C'
    assumes validate-clause-set I (cl-ecl' P)
    assumes fo-interpretation I
    assumes }\forallx\inP.(finite (cl-ecl x)
    shows validate-clause I (cl-ecl C)
proof (rule ccontr)
    assume \negvalidate-clause I (cl-ecl C)
    have not-sup: \neg (\existsP1 P2. (P1 \inS^P2 GS\wedgeP={P1,P2 } ^ superposition
P1 P2 C \sigma k C'))
    proof
        assume (\existsP1 P2. (P1 G S ^P2 \inS ^P={P1,P2 } ^ superposition P1
P2 C \sigma k C'))
    from this obtain P1 P2 where P1 \in P P2 \in P and superposition P1 P2 C
\sigmak C' by auto
    from }\langleP1\inP\rangle\mathrm{ and assms(2) have validate-clause I (cl-ecl P1) by auto
    from }\langleP2\inP\rangle\mathrm{ and assms(2) have validate-clause I (cl-ecl P2) by auto
    from assms(4) and }\langleP1\inP\rangle\mathrm{ have finite (cl-ecl P1) by auto
    from assms(4) and }\langleP2, PP\rangle\mathrm{ have finite (cl-ecl P2) by auto
    from assms(3) and 〈finite (cl-ecl P1)〉 and 〈finite (cl-ecl P2)〉
            and <superposition P1 P2 C \sigmak C'` have set-entails-clause {(cl-ecl P1),
(cl-ecl P2) } (cl-ecl C)
            using superposition-is-sound by blast
    from this and assms(3) and <validate-clause I (cl-ecl P1)〉 and <validate-clause
I (cl-ecl P2)>
            have validate-clause I (cl-ecl C)
            using set-entails-clause-def [of {(cl-ecl P1),(cl-ecl P2) } cl-ecl C] by auto
    from this and «\negvalidate-clause I (cl-ecl C)\rangle show False by auto
    qed
```

```
    have not-fact: }\neg(\existsP1.(P1\inS\wedgeP={P1}\wedge factorization P1 C \sigma k C')
    proof
    assume ( }\existsP1.(P1\inS\wedgeP={P1}\wedge factorization P1 C \sigma k C'))
    from this obtain P1 where P1 \inP and factorization P1 C \sigmak 重 by auto
    from }\langleP1\inP\rangle\mathrm{ and assms(2) have validate-clause I (cl-ecl P1) by auto
    from assms(4) and }\langleP1\inP\rangle\mathrm{ have finite (cl-ecl P1) by auto
    from assms(3) and <finite (cl-ecl P1)〉 and
        <actorization P1 C \sigma k C`> have clause-entails-clause (cl-ecl P1) (cl-ecl C)
        using factorization-is-sound by auto
    from this and assms(3) and «validate-clause I (cl-ecl P1)〉
        have validate-clause I (cl-ecl C) unfolding clause-entails-clause-def by auto
    from this and «\negvalidate-clause I (cl-ecl C)〉 show False by auto
qed
    have not-ref: }\neg(\existsP1.(P1\inS\wedgeP={P1}\wedge reflexion P1 C \sigmak C')
    proof
    assume (\existsP1. (P1 \inS\wedgeP={P1}^ reflexion P1 C \sigma k C'))
    from this obtain P1 where P1 \inP and reflexion P1 C \sigmak C' by auto
    from }\langleP1\inP\rangle\mathrm{ and assms(2) have validate-clause I (cl-ecl P1) by auto
    from assms(4) and }\langleP1\inP\rangle\mathrm{ have finite (cl-ecl P1) by auto
    from assms(3) and <finite (cl-ecl P1)〉 and
        <reflexion P1 C \sigma k C'〉 have clause-entails-clause (cl-ecl P1) (cl-ecl C)
        using reflexion-is-sound by auto
    from this and assms(3) and «validate-clause I (cl-ecl P1)>
        have validate-clause I (cl-ecl C) unfolding clause-entails-clause-def by auto
    from this and «\negvalidate-clause I (cl-ecl C)\rangle show False by auto
qed
from not-sup not-fact not-ref and assms(1) show False unfolding derivable-def
by blast
qed
lemma all-derived-clauses-are-finite:
    shows derivable-ecl C S \Longrightarrow \forallx S S. (finite (cl-ecl x)) \Longrightarrow finite (cl-ecl C)
proof (induction rule: derivable-ecl.induct)
    fix C :: 'a eclause fix }S\mathrm{ assume C }\in
    assume }\forallx\inS.(finite (cl-ecl x)
    from this }\langleC\inS\rangle\mathrm{ show finite (cl-ecl C) by auto
next
    fix C S fix D :: 'a eclause assume derivable-ecl C S
    assume }\forallx\inS.(finite (cl-ecl x)) assume hyp-ind: \forallx\inS.(finite (cl-ecl x))
finite (cl-ecl C)
        (renaming-cl C D)
    from <(renaming-cl C D)` obtain \eta where D = (subst-ecl C \eta)
        unfolding renaming-cl-def by auto
    obtain C-Cl T where C=(Ecl C-Cl T) using eclause.exhaust by auto
    from this and <D = (subst-ecl C \eta)>
        have (cl-ecl D) = (subst-cl (cl-ecl C) \eta) by auto
    from this hyp-ind 〈\forallx\inS.(finite (cl-ecl x))〉 show finite (cl-ecl D)
        using substs-preserve-finiteness by auto
next
```

```
    fix PSC S' \sigma C'
    assume h:\forallx.x\inP\longrightarrow derivable-ecl }xS\wedge((\forallx\inS.finite (cl-ecl x))\longrightarrow finit
(cl-ecl x))
    assume derivable C P S' \sigma FirstOrder C'
    assume }\forallx\inS\mathrm{ . finite (cl-ecl x)
    from h and }\langle\forallx\inS. finite (cl-ecl x)\rangle have \forallx\inP. (finite (cl-ecl x)) by meti
    from this and «derivable CP S'\sigma FirstOrder C'> show finite (cl-ecl C)
    using derivable-clauses-are-finite by auto
qed
lemma all-derived-clauses-are-wellconstrained:
    shows derivable-ecl C S \Longrightarrow}\forallx\inS\mathrm{ . (well-constrained }x\mathrm{ ) }>\mathrm{ well-constrained
C
proof (induction rule: derivable-ecl.induct)
    fix C :: 'a eclause fix }S\mathrm{ assume C }\in
    assume }\forallx\inS\mathrm{ . (well-constrained x)
    from this }\langleC\inS\rangle\mathrm{ show well-constrained C by auto
next
    fix C S fix D :: 'a eclause assume derivable-ecl C S
    assume }\forallx\inS.(\mathrm{ well-constrained x) assume hyp-ind: }\forallx\inS\mathrm{ . (well-constrained
x)\Longrightarrow well-constrained C
    (renaming-cl C D)
    from }\langle\forallx\inS\mathrm{ . (well-constrained x)> and hyp-ind have well-constrained C by
auto
    from «(renaming-cl C D)` obtain \eta where D = (subst-ecl C \eta)
        unfolding renaming-cl-def by auto
    from this and <well-constrained C` show well-constrained D
        using substs-preserve-well-constrainedness by auto
next
    fix PSC S'\sigma C'
    assume }\forallx.x\inP\longrightarrow\mathrm{ derivable-ecl }xS\wedge(\mathrm{ Ball S well-constrained }\longrightarrow\mathrm{ well-constrained
x)
    assume derivable C P S' \sigma FirstOrder C'
    assume Ball S well-constrained
    from <derivable C P S'\sigma FirstOrder C'> show well-constrained C
        using derivable-clauses-are-well-constrained by auto
qed
lemma SOUNDNESS:
    shows derivable-ecl C S \Longrightarrow\forallx\inS.(finite (cl-ecl x))
        set-entails-clause (cl-ecl'S) (cl-ecl C)
proof (induction rule: derivable-ecl.induct)
    fix C :: 'a eclause fix S assume C }\in
    assume }\forallx\inS.(finite (cl-ecl x))
    from }\langleC\inS\rangle\mathrm{ show set-entails-clause (cl-ecl'S) (cl-ecl C)
            unfolding set-entails-clause-def by auto
next
    fix C S fix D :: 'a eclause assume derivable-ecl C S
    assume }\forallx\inS\mathrm{ . (finite (cl-ecl x))
```

```
    assume hyp-ind: \(\forall x \in S\). (finite (cl-ecl \(x)) \Longrightarrow\) set-entails-clause (cl-ecl'S)
(cl-ecl C)
    assume (renaming-cl CD)
    from «(renaming-cl \(C D)\) ) obtain \(\eta\) where \(D=(\) subst-ecl \(C \eta)\)
    unfolding renaming-cl-def by auto
    obtain \(C-C l T\) where \(C=(E c l C-C l T)\) using eclause.exhaust by auto
    from this and \(\langle D=(\) subst-ecl \(C \eta)\rangle\)
    have \((c l-e c l ~ D)=(\) subst-cl \((c l-e c l C) \eta)\) by auto
    show set-entails-clause (cl-ecl'S) (cl-ecl D)
    proof (rule ccontr)
    assume \(\neg\) ?thesis
    from this obtain \(I\) where fo-interpretation \(I\) and \(i\) : validate-clause-set \(I\) (cl-ecl
'S)
        \(\neg\) validate-clause I (cl-ecl D)
        unfolding set-entails-clause-def by auto
    from 〈 \(\neg\) validate-clause \(I(c l-e c l D)\rangle\) and \(\langle(c l-e c l ~ D)=(s u b s t-c l(c l-e c l C) \eta)\rangle\)
        have \(\neg\) validate-clause \(I\) (cl-ecl C) using instances-are-entailed by metis
    from this and 〈fo-interpretation \(I\rangle i\) have \(\neg\) set-entails-clause (cl-ecl' \(S\) ) (cl-ecl
C)
            unfolding set-entails-clause-def by auto
    from this and \(\langle\forall x \in S\). (finite (cl-ecl \(x)\) ) \(\rangle\) hyp-ind show False by auto
    qed
next
    fix \(P S C S^{\prime} \sigma C^{\prime}\)
    assume \(h: \forall x . x \in P \longrightarrow\) derivable-ecl \(x S \wedge((\forall x \in S\). finite (cl-ecl \(x)) \longrightarrow\)
set-entails-clause (cl-ecl'S) (cl-ecl x))
    assume derivable C P \(S^{\prime} \sigma\) FirstOrder \(C^{\prime}\)
    assume \(\forall x \in S\). finite (cl-ecl \(x\) )
    from \(h\) and \(\langle\forall x \in S\). finite (cl-ecl \(x\) )〉 have \(i\) : \(\forall x \in P\). set-entails-clause (cl-ecl
    'S) (cl-ecl \(x)\)
            by metis
    show set-entails-clause (cl-ecl'S) (cl-ecl C)
    proof (rule ccontr)
        assume \(\neg\) ?thesis
            from this obtain \(I\) where fo-interpretation \(I\) and \(i i\) : validate-clause-set \(I\)
(cl-ecl'S)
            \(\neg\) validate-clause I (cl-ecl C)
            unfolding set-entails-clause-def by auto
    from \(h\langle\forall x \in S\). finite (cl-ecl \(x)\rangle\) have \((\forall x \in P\). finite (cl-ecl \(x)\) )
        using all-derived-clauses-are-finite by metis
    from 〈fo-interpretation \(I\rangle i\) and \(i i\)
            have \(\forall x \in P . \quad(\) validate-clause \(I(\) cl-ecl \(x))\) unfolding set-entails-clause-def
by auto
    from this have validate-clause-set \(I(\mathrm{cl-ecl}\) ' \(P\) ) by auto
    from this and \(\langle(\forall x \in P\). finite (cl-ecl \(x))\rangle\langle\) fo-interpretation \(I\rangle\langle\) derivable \(C P\)
\(S^{\prime} \sigma\) FirstOrder \(C^{\prime}{ }^{\prime}\)
            have validate-clause I (cl-ecl C)
            using derivable-clauses-are-entailed [of CP \(S^{\prime} \sigma\) FirstOrder \(C^{\prime} I\) ] by blast
    from this and «ᄀvalidate-clause \(I(\) cl-ecl \(C)\) 〉show False by auto
```

```
qed
qed
lemma REFUTABLE-SETS-ARE-UNSAT:
    assumes }\forallx\inS.(finite (cl-ecl x))
    assumes derivable-ecl C S
    assumes (cl-ecl C={})
    shows \neg(satisfiable-clause-set (cl-ecl'S))
proof
    assume (satisfiable-clause-set (cl-ecl'S))
    then obtain I where fo-interpretation I and model: validate-clause-set I (cl-ecl
    'S)
        unfolding satisfiable-clause-set-def [of cl-ecl'S] by blast
    from assms(1) assms(2) have set-entails-clause (cl-ecl'S) (cl-ecl C)
        using SOUNDNESS by metis
    from this <fo-interpretation I〉 and model have validate-clause I (cl-ecl C)
        unfolding set-entails-clause-def by auto
    from this and assms(3) show False by auto
qed
```


## 6 Redundancy Criteria and Saturated Sets

We define redundancy criteria. We use similar notions as in the Bachmair and Ganzinger paper, the only difference is that we have to handle the sets of irreducible terms associated with the clauses. Indeed, to ensure completeness, we must guarantee that all the terms that are irreducible in the entailing clauses are also irreducible in the entailed one (otherwise some needed inferences could be blocked due the irreducibility condition, as in the basic superposition calculus). Of course, if the attached sets of terms are empty, then this condition trivially holds and the definition collapses to the usual one.
We introduce the following relation:

```
definition subterms-inclusion :: 'a trm set \(\Rightarrow\) 'a trm set \(\Rightarrow\) bool
    where subterms-inclusion E1 E2 \(=(\forall x 1 \in E 1 . \exists x 2 \in E 2 .(\) occurs-in \(x 1\) x2 \())\)
lemma subterms-inclusion-refl:
    shows subterms-inclusion E E
proof (rule ccontr)
    assume \(\neg\) subterms-inclusion E E
    from this obtain \(x 1\) where \(x 1 \in E\) and \(\neg\) occurs-in \(x 1\) x1 unfolding sub-
terms-inclusion-def by force
    from \(\prec \neg\) occurs-in \(x 1\) x1〉 have \(\neg(\exists\) p. subterm x1 px1) unfolding occurs-in-def
by auto
    from this have \(\neg\) subterm x1 Nil x1 by metis
    from this show False by auto
qed
```

```
        subsetD)
```

lemma set-inclusion-preserve-normalization:
assumes all-trms-irreducible E f
assumes $E^{\prime} \subseteq E$
shows all-trms-irreducible $E^{\prime} f$
by (meson all-trms-irreducible-def assms(1) assms(2) subsetD)
lemma subterms-inclusion-preserves-normalization:
assumes all-trms-irreducible E f
assumes subterms-inclusion $E^{\prime} E$
shows all-trms-irreducible $E^{\prime} f$
by (meson all-trms-irreducible-def assms(1) assms(2) occur-in-subterm subterms-inclusion-def)

We define two notions of redundancy, the first one is for inferences: any derivable clause must be entailed by a set of clauses that are strictly smaller than one of the premises.

```
definition redundant-inference :
    'a eclause \(\Rightarrow{ }^{\prime} a\) eclause set \(\Rightarrow{ }^{\prime} a\) eclause set \(\Rightarrow\) 'a subst \(\Rightarrow\) bool
    where redundant-inference \(C S P \sigma \longleftrightarrow\left(\exists S^{\prime} \subseteq\right.\) instances \(S\).
    set-entails-clause (clset-instances \(\left.S^{\prime}\right)(\) cl-ecl \(C) \wedge\)
    ( \(\forall x \in S^{\prime}\). subterms-inclusion (subst-set (trms-ecl (fst x)) (snd x)) (trms-ecl C))
\(\wedge\)
    \(\left(\forall x \in S^{\prime} . \exists D^{\prime} \in \operatorname{cl-ecl}{ }^{\prime} P .\left((\operatorname{cl-ecl}(f s t x)\right.\right.\), snd \(\left.\left.\left.x),\left(D^{\prime}, \sigma\right)\right) \in \operatorname{cl-ord}\right)\right)\)
```

The second one is the usual notion for clauses: a clause is redundant if it is entailed by smaller (or equal) clauses.

```
definition redundant-clause ::
    ' \(a\) eclause \(\Rightarrow\) ' \(a\) eclause set \(\Rightarrow\) 'a subst \(\Rightarrow\) ' \(a\) clause \(\Rightarrow\) bool
    where (redundant-clause \(C S \sigma C^{\prime}\) ) =
        \(\left(\exists S^{\prime} .\left(S^{\prime} \subseteq(\right.\right.\) instances \(S) \wedge\left(\right.\) set-entails-clause \(\left(\right.\) clset-instances \(\left.S^{\prime}\right)(\) cl-ecl \(\left.C)\right)\)
\(\wedge\)
    \(\left(\forall x \in S^{\prime} .(\right.\) subterms-inclusion \((\) subst-set \((\) trms-ecl \((f s t x))(\) snd \(x))\)
    \((\) trms-ecl \(C))) \wedge\)
    \(\left(\forall x \in S^{\prime} .\left(\left((\right.\right.\right.\) mset-ecl \(((\) fst \(x),(\) snd \(x))),\left(\right.\) mset-cl \(\left.\left.\left(C^{\prime}, \sigma\right)\right)\right) \in(\) mult \((\) mult
trm-ord))
    \(\left.\left.\left.\left.\vee(\operatorname{mset}-e c l((f s t x),(\operatorname{snd} x)))=\operatorname{mset}-c l\left(C^{\prime}, \sigma\right)\right)\right)\right)\right)\)
```

Note that according to the definition above, an extended clause is always redundant w.r.t. a clause obtained from the initial one by adding in the attached set of terms a subterm of a term that already occurs in this set. This remark is important because explicitly adding such subterms in the
attached set may prune the search space，due to the fact that the containing term can be removed at some point when calling the function dom－trm． Adding the subterm explicitly is thus useful in this case．In practice，the simplest solution may be to assume that the set of irreducible terms is closed under subterm．
Of course，a clause is also redundant w．r．t．any clause obtained by removing terms in the attached set．In particular，terms can be safely removed from the set of irreducible terms of the entailing clauses if needed to make a given clause redundant．

```
lemma self-redundant-clause:
    assumes \(C \in S\)
    assumes \(C^{\prime}=(\) cl-ecl \(C)\)
    assumes ground-clause (subst-cl (cl-ecl C) \(\sigma\) )
    shows redundant-clause (subst-ecl \(C \sigma\) ) \(S \sigma C^{\prime}\)
proof -
    obtain \(C l-C\) and \(T\) where \(C=E c l C l-C T\) using eclause.exhaust by auto
    from this have cl-ecl \(C=C l-C\) and trms-ecl \(C=T\) by auto
    let ? \(\mathrm{Cl}-\mathrm{C}=\) subst-cl \(\mathrm{Cl}-\mathrm{C} \sigma\)
    let ? \(T=\) subst-set \(T \sigma\)
    let ? \(C=\) subst-ecl \(C \sigma\)
    from \(\langle C=E c l C l-C T\rangle\) have ? \(C=(E c l ? C l-C\) ? \(T)\) by auto
    from this have cl-ecl ? \(C=? C l-C\) and trms-ecl \(? C=\) ? \(T\) by auto
    let \(? S=\{(C, \sigma)\}\)
    from \(\operatorname{assms}(1) \operatorname{assms}(3)\) have \(i: ? S \subseteq\) (instances \(S\) ) unfolding instances-def
by auto
    from \(\langle c l-e c l C=C l-C\rangle\) have clset-instances \(? S=\{\) ?Cl-C \(\}\) unfolding clset-instances-def
    by auto
    from this and 〈cl-ecl ?C \(=\) ?Cl-C〉 have ii: set-entails-clause (clset-instances
?S) (cl-ecl ?C)
    using set-entails-clause-member by force
    have iii: \((\forall x \in\) ?S. ( subterms-inclusion (subst-set (trms-ecl (fst x)) (snd x))
                    (trms-ecl ? C)))
    proof
    fix \(x\) assume \(x \in\) ? \(S\)
    from this have \(x=(C, \sigma)\) by auto
    from this \(\langle C=E c l\) Cl-C T〉
        have subst-set (trms-ecl \((\) fst \(x))(\) snd \(x)=\) ?T by auto
    from this and 〈trms-ecl ?C = ?T〉
        have subst-set (trms-ecl \((f s t x))(\) snd \(x)=(\) trms-ecl ? \(C)\) by auto
    from this show ( subterms-inclusion (subst-set (trms-ecl \((f\) st \(x))(\) snd \(x)\) )
                                    (trms-ecl ?C))
                    using subterms-inclusion-refl by auto
    qed
    have iv: \(\left(\forall x \in\right.\) ?S. \(\left(\left((\right.\right.\) mset-ecl \(((f s t x),(\) snd \(x))),\left(\right.\) mset-cl \(\left.\left.\left(C^{\prime}, \sigma\right)\right)\right) \in(\) mult (mult
trm-ord))
    \(\vee(\) mset-ecl \(((f s t x),(\) snd \(\left.\left.x)))=m s e t-c l\left(C^{\prime}, \sigma\right)\right)\right)\)
    proof
```

```
    fix }x\mathrm{ assume }x\in\mathrm{ ?S
    from this have }x=(C,\sigma)\mathrm{ by auto
        from this <C = Ecl Cl-C T〉 have (mset-ecl ((fst x),(snd x))) = (mset-ecl
(C,\sigma)) by auto
    from this «C' = (cl-ecl C)\rangle have (mset-ecl ((fst x),(snd x))) = mset-cl ( (C',\sigma)
by auto
    from this show ( ((mset-ecl }((fst x),(\mathrm{ snd x ) )),(mset-cl ( }\mp@subsup{C}{}{\prime},\sigma)))\in(\mathrm{ mult (mult
trm-ord))
            \vee (mset-ecl }((fst x),(\mathrm{ snd x ) )) = mset-cl ( (C',}))\mathrm{ by auto
    qed
    from i ii ioi iv show ?thesis unfolding redundant-clause-def by metis
qed
definition trms-subsumes
    where trms-subsumes C D \sigma
    =((subst-cl (cl-ecl C)\sigma) = (cl-ecl D)
    \wedge((subst-set (trms-ecl C)\sigma)\subseteqtrms-ecl D))
definition inference-closed
    where inference-closed S = (\forallP C'D \vartheta.
        (derivable D P S \vartheta FirstOrder C')}\longrightarrow(D\inS)
```

Various notions of saturatedness are defined, depending on the kind of inferences that are considered and on the redundancy criterion.

The first definition is the weakest one: all ground inferences must be redundant (this definition is used for the completeness proof to make it the most general).
definition ground-inference-saturated :: 'a eclause set $\Rightarrow$ bool
where (ground-inference-saturated $S)=\left(\forall C P \sigma C^{\prime}\right.$. (derivable CPS $\sigma$ Ground $\left.C^{\prime}\right) \longrightarrow$
(ground-clause $($ cl-ecl $C)) \longrightarrow$ (grounding-set $P \sigma) \longrightarrow$ (redundant-inference $C S P \sigma)$ )

The second one states that every ground instance of a first-order inference must be redundant.

```
definition inference-saturated :: 'a eclause set \(\Rightarrow\) bool
    where \((\) inference-saturated \(S)=\left(\forall C P \sigma C^{\prime} D \vartheta \eta\right.\).
    (derivable CPS Ground \(\left.C^{\prime}\right) \longrightarrow\) (ground-clause (cl-ecl \(C\) )) \(\longrightarrow\) (grounding-set
\(P \sigma\) )
    \(\longrightarrow\left(\right.\) derivable D P S \(\vartheta\) FirstOrder \(\left.C^{\prime}\right) \longrightarrow(\) trms-subsumes \(D C \eta)\)
    \(\longrightarrow(\sigma \doteq \vartheta \diamond \eta)\)
    \(\longrightarrow(\) redundant-inference (subst-ecl D \(\eta\) ) S P \(\sigma\) )
```

The last definition is the most restrictive one: every derivable clause must be redundant.
definition clause-saturated :: 'a eclause set $\Rightarrow$ bool where (clause-saturated $S)=\left(\begin{array}{llll}\forall & C & \sigma C^{\prime} D \vartheta \eta\end{array}\right.$.

```
(derivable C P S Ground \(\left.C^{\prime}\right) \longrightarrow(\) ground-clause \((\) cl-ecl \(C))\)
    \(\longrightarrow\left(\right.\) derivable D P S \(\vartheta\) FirstOrder \(\left.C^{\prime}\right) \longrightarrow(\) trms-subsumes \(D C \eta)\)
\(\longrightarrow(\sigma \doteq \vartheta \diamond \eta)\)
\(\longrightarrow\left(\right.\) redundant-clause (subst-ecl D \(\eta\) ) \(\left.S \sigma C^{\prime}\right)\) )
```

We now relate these various notions, so that the forthcoming completeness proof applies to all of them. To this purpose, we have to show that the conclusion of a (ground) inference rule is always strictly smaller than one of the premises.
lemma conclusion-is-smaller-than-premisses:
assumes derivable CPS Ground $C^{\prime}$
assumes $\forall x \in S$. (finite (cl-ecl $x$ ))
assumes grounding-set $P \sigma$
shows $\exists D .\left(D \in P \wedge\left(\left(\left(\right.\right.\right.\right.$ mset-cl $\left.\left(C^{\prime}, \sigma\right)\right),($ mset-ecl $\left.(D, \sigma))\right) \in($ mult (mult trm-ord))))
proof (rule ccontr)
assume hyp: $\neg\left(\exists D .\left(D \in P \wedge\left(\left(\left(\right.\right.\right.\right.\right.$ mset-cl $\left.\left(C^{\prime}, \sigma\right)\right),($ mset-ecl $\left.(D, \sigma))\right) \in($ mult (mult trm-ord)))))
from $\operatorname{assms}(1)$ have $P \subseteq S$ unfolding derivable-def by auto
have not-sup: $\neg(\exists P 1 \overline{P 2} .(P 1 \in P \wedge P 2 \in P \wedge$ superposition P1 P2 $C \sigma$ Ground $C^{\prime}$ ))

## proof

assume ( $\exists$ P1 P2. $\left(P 1 \in P \wedge P 2 \in P \wedge\right.$ superposition P1 P2 $C \sigma$ Ground $\left.C^{\prime}\right)$ )
then obtain P1 P2 where P1 $\in P$ P2 $\in P$ superposition P1 P2 $C \sigma$ Ground
$C^{\prime}$ by auto
from «superposition P1 P2 $C \sigma$ Ground $C^{\prime} 〉$ obtain $L t s u v M L^{\prime}$ polarity $u^{\prime}$ $p t^{\prime} C l-C N T$ where
$M \in($ cl-ecl P2) $L \in($ cl-ecl P1)
orient-lit-inst M u v pos $\sigma$
orient-lit-inst $L t s$ polarity $\sigma$
subterm $t p u^{\prime}$
ck-unifier $u^{\prime} u \sigma$ Ground
replace-subterm t p v $t^{\prime}$
$L^{\prime}=m k$-lit polarity $\left(E q t^{\prime} s\right)$
(C = (Ecl Cl-C NT) )
(subst $u \sigma) \neq($ subst $v \sigma)$
( (subst-lit M $\sigma$ ), (subst-lit L $\sigma$ ))
$\in$ lit-ord
strictly-maximal-literal P2 M $\sigma$
$C l-C=\left(\right.$ subst-cl $\left(((c l-e c l\right.$ P1 $)-\{L\}) \cup\left(((\right.$ cl-ecl P2 $\left.\left.)-\{M\}) \cup\left\{L^{\prime}\right\}\right)\right)$
$\sigma$ )
$C^{\prime}=\left(((\right.$ cl-ecl P1 $)-\{L\}) \cup\left(((\right.$ cl-ecl P2 $\left.\left.)-\{M\}) \cup\left\{L^{\prime}\right\}\right)\right)$
unfolding superposition-def by blast
from $\langle P 1 \in P\rangle$ and $\operatorname{assms}(2)$ and $\langle P \subseteq S\rangle$ have finite (cl-ecl P1) by auto
from $\langle P 2 \in P\rangle$ and $\operatorname{assms}(2)$ and $\langle P \subseteq S\rangle$ have finite (cl-ecl P2) by auto
from $\operatorname{assms}(3)$ and $\langle P 2 \in P\rangle$ have ground-clause (subst-cl (cl-ecl P2) $\sigma$ )
unfolding grounding-set-def by auto
from this have vars-of-cl (subst-cl (cl-ecl P2) $\sigma$ ) $=\{ \}$ by auto
from $\langle M \in($ cl－ecl P2）$)$ have（subst－lit $M \sigma) \in($ subst－cl（cl－ecl P2）$\sigma$ ）by auto
from this and 〈vars－of－cl（subst－cl（cl－ecl P2）$\sigma$ ）$=\{ \}$ 〉 have vars－of－lit（subst－lit $M \sigma)=\{ \}$
by auto
from 〈orient－lit－inst Muv pos $\sigma$ 〉 have
orient－lit（subst－lit M $\sigma$ ）（subst $u \sigma)($ subst $v \sigma)$ pos
using lift－orient－lit by auto
from this and 〈vars－of－lit（subst－lit $M \sigma$ ）$=\{ \}$ 〉have vars－of（subst u $\sigma$ ）$=$ \｛\}
using orient－lit－vars by blast
from＜orient－lit（subst－lit M $\sigma$ ）（subst $u \sigma$ ）（subst $v \sigma$ ）pos〉
and $\langle$ vars－of－lit（subst－lit $M \sigma)=\{ \}\rangle$ have vars－of（subst $v \sigma)=\{ \}$ using orient－lit－vars by blast
from＜orient－lit（subst－lit $M \sigma$ ）（subst $u \sigma)($ subst $v \sigma)$ pos〉
have $(($ subst $u \sigma),($ subst $v \sigma)) \notin$ trm－ord
unfolding orient－lit－def by auto
from this and 〈（subst $u \sigma) \neq($ subst $v \sigma)\rangle$ and＜vars－of（subst $u \sigma)=\{ \}\rangle\langle v a r s-o f($ subst $v \sigma)=\{ \}\rangle$
have $(($ subst $v \sigma),($ subst $u \sigma)) \in$ trm－ord using trm－ord－ground－total unfolding ground－term－def by blast
from $\operatorname{assms}(3)$ and $\langle P 1 \in P\rangle$ have ground－clause（subst－cl（cl－ecl P1）$\sigma$ ） unfolding grounding－set－def by auto
from this have vars－of－cl（subst－cl（cl－ecl P1）$\sigma$ ）$=\{ \}$ by auto
from $\langle L \in($ cl－ecl P1）$)$ have（subst－lit $L \sigma) \in($ subst－cl（cl－ecl P1）$\sigma$ ）by auto
from this and «vars－of－cl（subst－cl（cl－ecl P1）$\sigma$ ）$=\{ \}$ 〉 have vars－of－lit（subst－lit $L \sigma)=\{ \}$
by auto
from＜orient－lit－inst $L$ ts polarity $\sigma$ 〉 have
orient－lit（subst－lit $L \sigma$ ）（subst $t \sigma$ ）（subst $s \sigma$ ）polarity
using lift－orient－lit by auto
from this and 〈vars－of－lit（subst－lit $L \sigma$ ）$=\{ \}$＞have vars－of（subst $t \sigma)=\{ \}$ using orient－lit－vars by blast
from «orient－lit（subst－lit L $\sigma$ ）（subst $t \sigma$ ）（subst $s \sigma$ ）polarity〉
and «vars－of－lit（subst－lit $L \sigma)=\{ \}$ have vars－of（subst s $\sigma$ ）$=\{ \}$
using orient－lit－vars by blast
let ？mC1 $=$ mset－ecl $(P 1, \sigma)$
let $? m C 2=\operatorname{mset}-e c l(C, \sigma)$
from $\langle L \in($ cl－ecl P1）〉〈finite（cl－ecl P1）〉
have mset－set（cl－ecl P1）$=$ mset－set（（cl－ecl P1）－\｛L\})+mset-set $\{L\}$ using split－mset－set［of cl－ecl P1 cl－ecl P1－\｛ L \} \{ L \}] by blast
from this have d1：$\{\#$（mset－lit（subst－lit $x \sigma)$ ）．$x \in \#$（mset－set（cl－ecl P1）） \＃\}
$=\{\#($ mset－lit $($ subst－lit $x \sigma)) \cdot x \in \#($ mset－set $(($ cl－ecl P1）$-\{L\})) \#\}$ $+\{\#(m s e t-l i t(s u b s t-l i t ~ x \sigma)) . x \in \#(m s e t-s e t\{L\}) \#\}$ using split－image－mset by auto
let ？$C=\left(\left((\right.\right.$ cl－ecl P1）$-\{L\}) \cup\left(((\right.$ cl－ecl P2 $\left.\left.)-\{M\}) \cup\left\{L^{\prime}\right\}\right)\right)$
from＜finite（cl－ecl P1）〉〈finite（cl－ecl P2）〉 have finite ？C by auto
let ${ }^{2} C^{\prime}=? C-(($ cl－ecl P1 $)-\{L\})$
from 〈finite ？$C$ 〉 have finite ？$C^{\prime}$ by auto
have ？$C=(($ cl－ecl P1 $)-\{L\}) \cup ? C^{\prime}$ by auto
from 〈finite（cl－ecl P1）〉〈finite ？$\left.C^{\prime}\right\rangle$
have mset－set ？$C=$ mset－set $(($ cl－ecl P1 $)-\{L\})+$ mset－set ${ }^{\circ} C^{\prime}$
using split－mset－set［of ？C cl－ecl P1－\｛L\} ?C $]$ by blast
from this have d2：$\{\#$（mset－lit（subst－lit $x \sigma)$ ）．$x \in \#$（mset－set ？C）\＃\}
$=\{\#($ mset－lit $($ subst－lit $x \sigma)) . x \in \#($ mset－set $(($ cl－ecl P1）$)-\{L\})) \#\}$
$+\left\{\#(\right.$ mset－lit $($ subst－lit $x \sigma)) \cdot x \in \#\left(\right.$ mset－set ？$\left.\left.C^{\prime}\right) \#\right\}$
using split－image－mset by auto
have $\{\#($ mset－lit（subst－lit $x \sigma)) . x \in \#($ mset－set $\{L\}) \#\} \neq\{\#\}$
by auto
let $? K=\left\{\#(\right.$ mset－lit $($ subst－lit $x \sigma)) . x \in \#\left(\right.$ mset－set ？$\left.\left.C^{\prime}\right) \#\right\}$
let ？$J=\{\#($ mset－lit $($ subst－lit $x \sigma)) . x \in \#($ mset－set $\{L\}) \#\}$
have $(\forall k \in$ set－mset ？$K . \exists j \in$ set－mset ？$J .(k, j) \in($ mult trm－ord $))$
proof
fix $k$ assume $k \in$ set－mset ？$K$
from this have $k \in \#$ ？K by auto
from this obtain $M^{\prime}$ where $M^{\prime} \in \#\left(\right.$ mset－set ？$\left.C^{\prime}\right)$ and $k=($ mset－lit（subst－lit $\left.M^{\prime} \sigma\right)$ ）
using image－mset－thm［of ？K $\lambda x$ ．（mset－lit（subst－lit $x \sigma)$ ）（mset－set ？$\left.\left.C^{\prime}\right)\right]$ by metis
from $\left\langle M^{\prime} \in \#\left(\right.\right.$ mset－set ？$\left.\left.C^{\prime}\right)\right\rangle$ and $\left\langle\right.$ finite ？$\left.C^{\prime}\right\rangle$ have $M^{\prime} \in ? C^{\prime}$ by auto
have $L \in \#$（ mset－set $\{L\}$ ）by auto
from this have（mset－lit（subst－lit L $\sigma$ ）$\in \#$ ？J）by auto
from this have（mset－lit（subst－lit L $\sigma$ ）$\in$ set－mset ？J）by auto
have $\{\#($ mset－lit $($ subst－lit $x \sigma)) . x \in \#(\operatorname{mset}$－set $\{L\}) \#\} \neq\{\#\}$ by auto

```
show \(\exists j \in\) set-mset ? \(J .(k, j) \in(\) mult trm-ord \()\)
proof (cases)
    assume \(M^{\prime} \in(\) cl-ecl P2) \(-\{M\}\)
    from this and 〈strictly-maximal-literal P2 \(M \sigma\) 〉
        have \(\left(\left(\right.\right.\) subst-lit \(\left.M^{\prime} \sigma\right),(\) subst-lit \(\left.M \sigma)\right) \in\) lit-ord
            unfolding strictly-maximal-literal-def by metis
    from this and \(\langle((\) subst-lit \(M \sigma),(\) subst-lit \(L \sigma)) \in\) lit-ord \(\rangle\)
            have \(\left(\left(\right.\right.\) subst-lit \(\left.M^{\prime} \sigma\right),(\) subst-lit \(\left.L \sigma)\right) \in\) lit-ord
            using lit-ord-trans unfolding trans-def by metis
    from this have ((mset-lit (subst-lit \(\left.M^{\prime} \sigma\right)\) ),
                    (mset-lit \((\) subst-lit \(L \sigma))) \in(\) mult trm-ord \()\)
            unfolding lit-ord-def by auto
    from 〈(mset-lit (subst-lit \(L \sigma) \in\) set-mset ? J) 〉 this \(\left\langle\left(\left(m s e t-l i t\right.\right.\right.\) (subst-lit \(M^{\prime}\)
```

$\sigma)$ ），
$($ mset－lit $($ subst－lit $L \sigma))) \in($ mult trm－ord $)\rangle$
and $\left\langle k=\left(\right.\right.$ mset－lit（subst－lit $\left.\left.\left.M^{\prime} \sigma\right)\right)\right\rangle$ show ？thesis by blast
next assume $M^{\prime} \notin($ cl－ecl P2）$-\{M\}$
from this and $\left\langle M^{\prime} \in ? C^{\prime}\right\rangle$ have $M^{\prime}=L^{\prime}$ by auto
from＜subterm $\left.t p u^{\prime}\right\rangle$ have subterm（subst $t \sigma$ ）$p$（subst $u^{\prime} \sigma$ ）
using substs－preserve－subterms by blast
from 〈ck－unifier $u^{\prime} u \sigma$ Ground〉 have
（subst $u \sigma)=\left(\right.$ subst $\left.u^{\prime} \sigma\right)$ unfolding ck－unifier－def Unifier－def by auto
from this and $\langle(($ subst $v \sigma),($ subst $u \sigma)) \in$ trm－ord $\rangle$
have $\left((\right.$ subst $v \sigma),\left(\right.$ subst $\left.\left.u^{\prime} \sigma\right)\right) \in$ trm－ord by auto
from this «subterm $\left.t p u^{\prime}\right\rangle\left\langle r e p l a c e-s u b t e r m ~ t p v t^{\prime}\right\rangle$
have $\left(\left(\right.\right.$ subst $\left.t^{\prime} \sigma\right),($ subst $\left.t \sigma)\right) \in$ trm－ord
using replacement－monotonic by auto
have polarity $=$ pos $\vee$ polarity $=$ neg using sign．exhaust by auto
then have $\left(\left(\right.\right.$ subst－lit $\left.L^{\prime} \sigma\right),($ subst－lit $\left.L \sigma)\right) \in$ lit－ord
proof
assume polarity $=$ pos
from this and 〈orient－lit－inst $L t$ spolarity $\sigma$ 〉
have $i$ ： mset－lit（subst－lit $L \sigma))=\{\#($ subst s $\sigma) \#\}+\{\#($ subst $t \sigma) \#\}$ unfolding orient－lit－inst－def using add．commute by force
from $\left\langle L^{\prime}=m k\right.$－lit polarity $\left.\left(E q t^{\prime} s\right)\right\rangle\langle p o l a r i t y=p o s\rangle$
have ii：（mset－lit（subst－lit $\left.\left.L^{\prime} \sigma\right)\right)=\{\#($ subst $s \sigma) \#\}$
$+\left\{\#\left(\right.\right.$ subst $\left.\left.t^{\prime} \sigma\right) \#\right\}$
using add．commute by force
have $\{\#($ subst $t \sigma) \#\} \neq\{\#\}$ by auto
have $\left(\forall k^{\prime} \in\right.$ set－mset $\left\{\#\left(\right.\right.$ subst $\left.\left.t^{\prime} \sigma\right) \#\right\} . \exists j^{\prime} \in$ set－mset $\{\#($ subst $t \sigma)$
$\#\} .\left(k^{\prime}, j^{\prime}\right) \in($ trm－ord $\left.)\right)$
proof
fix $k^{\prime}$ assume $k^{\prime} \in$ set－mset $\left\{\#\left(\right.\right.$ subst $\left.\left.t^{\prime} \sigma\right) \#\right\}$
from this have $k^{\prime}=\left(\right.$ subst $\left.t^{\prime} \sigma\right)$ by auto
have $($ subst $t \sigma) \in$ set－mset $\{\#($ subst $t \sigma) \#\}$ by auto
from this $\left\langle k^{\prime}=\left(\right.\right.$ subst $\left.t^{\prime} \sigma\right)$ 〉
and $\left\langle\left(\left(\right.\right.\right.$ subst $\left.t^{\prime} \sigma\right),($ subst $\left.t \sigma)\right) \in$ trm－ord $\rangle$
show $\exists j^{\prime} \in$ set－mset $\{\#($ subst $t \sigma) \#\} .\left(k^{\prime}, j^{\prime}\right) \in($ trm－ord $)$
by auto
qed
from $i$ ii
$\left\langle\left(\left(\right.\right.\right.$ subst $\left.t^{\prime} \sigma\right),($ subst $\left.t \sigma)\right) \in$ trm－ord $\rangle$
have（mset－lit（subst－lit $\left.L^{\prime} \sigma\right),($ mset－lit（subst－lit $\left.L \sigma)\right)$ ） $\in$（mult trm－ord）
by（metis one－step－implies－mult empty－iff insert－iff set－mset－add－mset－insert set－mset－empty）
from this show ？thesis unfolding lit－ord－def by auto
next
assume polarity $=$ neg
from this and＜orient－lit－inst Ltspolarity $\sigma$ 〉

```
        have i:(mset-lit (subst-lit L \sigma))}={#(\mathrm{ subst s }\sigma),(\mathrm{ subst s }\sigma)#
        +{#(subst t \sigma),(subst t \sigma) #}
        unfolding orient-lit-inst-def by auto
            from <L' = mk-lit polarity (Eq t' s)\rangle\langlepolarity = neg> have
                subst-lit L' }\mp@subsup{L}{}{\prime}=(Neg(Eq(\mathrm{ subst t' }\sigma)(\mathrm{ subst s }\sigma)))\mathrm{ ) by auto
            from this have (mset-lit (subst-lit L'\sigma))
        ={# (subst t' \sigma),(subst t'\sigma),(subst s \sigma),(subst s \sigma)#}
        by auto
            from this have ii:(mset-lit (subst-lit L' \sigma))
                = {# (subst s \sigma), (subst s \sigma)#} + {# (subst t'\sigma), (subst t'\sigma) #}
                by (simp add: add.commute add.left-commute)
            have}{#(\mathrm{ subst t }\sigma),(\mathrm{ subst t }\sigma)#
            have ( }\forall\mp@subsup{k}{}{\prime}\in\mathrm{ set-mset {# (subst t' }\sigma),(\mathrm{ subst t' }\sigma)#}\mathrm{ .
                \existsj'\in set-mset {# (subst t \sigma),(subst t \sigma)#}. (k', j') \in(trm-ord))
            proof
            fix }\mp@subsup{k}{}{\prime}\mathrm{ assume }\mp@subsup{k}{}{\prime}\in\mathrm{ set-mset {# (subst t' }\sigma),(\mathrm{ subst t' }\sigma)#
            from this have }\mp@subsup{k}{}{\prime}=(\mathrm{ subst t' }\sigma\mathrm{ ) by auto
            have (subst t \sigma)\in set-mset {# (subst t \sigma),( subst t \sigma) #} by auto
            from this <k'}=(\mathrm{ subst t' }\sigma)\mathrm{ >
                and <((subst t' \sigma),(subst t \sigma)) \in trm-ord>
                show \existsj'\in set-mset {# (subst t \sigma),(subst t \sigma) #}. (k', j') \in (trm-ord)
            by auto
qed
from this i ii «{# (subst t \sigma), (subst t \sigma)#} = {#}`
            have (mset-lit (subst-lit L' \sigma),
                (mset-lit (subst-lit L \sigma))) \in(mult trm-ord)
                using one-step-implies-mult [of {# (subst t \sigma), (subst t \sigma) #}
                {# (subst t' \sigma),(subst t' \sigma)#} trm-ord
                {# (subst s \sigma),(subst s \sigma) #}]
            trm-ord-trans by auto
            from this show ?thesis unfolding lit-ord-def by auto
        qed
    from this and
        <(mset-lit (subst-lit L \sigma)\in set-mset ?J)>
        <k=(mset-lit (subst-lit M'\sigma))\rangle
        <M'}=\mp@subsup{L}{}{\prime}\rangle\mathrm{ show ?thesis unfolding lit-ord-def by auto
    qed
qed
from this d1 d2 have o:
    ({#mset-lit (subst-lit x \sigma).x }x=#\mathrm{ mset-set ?C #},
    {#mset-lit (subst-lit x \sigma). x \in# mset-set (cl-ecl P1)#})
    <mult (mult trm-ord)
using mult-trm-ord-trans one-step-implies-mult [of {# (mset-lit (subst-lit x \sigma)).
x &# (mset-set {L })#}
    {# (mset-lit (subst-lit x \sigma)). x }\in##(mset-set?C')#} mult trm-ord
```

```
    {# (mset-lit (subst-lit x \sigma)). x\in# (mset-set ((cl-ecl P1) - {L })) #} ] by
auto
    from this <C' = (((cl-ecl P1) - { L }) \cup (((cl-ecl P2) - { M }) \cup { L'} ))>
and }\langleP1\inP
    and hyp show False by auto
qed
have not-ref: }\neg(\existsP1.(P1\inP\wedge reflexion P1 C \sigma Ground C'))
proof
    assume ( }\exists>P1.(P1\inP\wedge reflexion P1 C \sigma Ground C'))
    then obtain P1 where P1 \inP reflexion P1 C \sigma Ground C' by auto
    from〈reflexion P1 C \sigma Ground C'`}\mathrm{ obtain L1 ts Cl-C Cl-P where
        (eligible-literal L1 P1 \sigma)
        (L1 \in(cl-ecl P1)) (Cl-C = (cl-ecl C)) (Cl-P = (cl-ecl P1 ))
        (orient-lit-inst L1 t s neg \sigma)
        (ck-unifier t s \sigma Ground)
        (Cl-C = (subst-cl ((Cl-P - {L1 }))) \sigma)
        (C'}=((Cl-P-{L1 }) ))
        unfolding reflexion-def by blast
```

    from \(\langle P 1 \in P\rangle\) and \(\operatorname{assms}(2)\) and \(\langle P \subseteq S\rangle\) have finite (cl-ecl P1) by auto
    let \({ }^{2} m C 1=m s e t-e c l(P 1, \sigma)\)
    let \(? m C 2=\) mset-ecl \((C, \sigma)\)
    from 〈L1 \(\in(\) cl-ecl P1)〉〈finite (cl-ecl P1)〉
        have mset-set \((\) cl-ecl P1) \(=\) mset-set \(((c l-e c l ~ P 1)-\{L 1\})+\) mset-set \(\{L 1\}\)
        using split-mset-set [of cl-ecl P1 cl-ecl P1 - \{ L1 \} \{L1 \}] by blast
    from this have d1: \(\{\#(\) mset-lit (subst-lit \(x \sigma)) . x \in \#(\) mset-set (cl-ecl P1))
    \#\}
$=\{\#($ mset-lit (subst-lit $x \sigma)) \cdot x \in \#($ mset-set $(($ cl-ecl P1) $-\{L 1\})) \#\}$
$+\{\#($ mset-lit (subst-lit $x \sigma)) . x \in \#(m s e t-s e t\{L 1\}) \#\}$
using split-image-mset by auto
let $? C=(($ cl-ecl P1 $)-\{L 1\})$
from 〈finite (cl-ecl P1)〉 have finite ?C by auto
let ? $C^{\prime}=\{ \}$
have finite ? $C^{\prime}$ by auto
have $? C=(($ cl-ecl P1 $)-\{L 1\}) \cup ? C^{\prime}$ by auto
from 〈finite (cl-ecl P1) 〉〈finite ? $\left.C^{\prime}\right\rangle$
have mset-set ? $C=$ mset-set $((c l-e c l ~ P 1)-\{L 1\})+$ mset-set ? $C^{\prime}$
using split-mset-set [of?C cl-ecl P1 - \{L1 \} ?C] by blast
from this have d2: $\{\#($ mset-lit (subst-lit $x \sigma)) . x \in \#($ mset-set ?C) $\#\}$
$=\{\#($ mset-lit $($ subst-lit $x \sigma)) . x \in \#($ mset-set $(($ cl-ecl P1 $)-\{L 1\})) \#\}$
$+\{\#($ mset-lit (subst-lit $x \sigma)) \cdot x \in \#\left(\right.$ mset-set ? $\left.\left.C^{\prime}\right) \#\right\}$
using split-image-mset by auto
have $\{\#($ mset-lit $($ subst-lit $x \sigma)) . x \in \#($ mset-set $\{L 1\}) \#\} \neq\{\#\}$
by auto
let ? $K=\{\#($ mset-lit (subst-lit $x \sigma)) . x \in \#\left(\right.$ mset-set ? $\left.\left.C^{\prime}\right) \#\right\}$
let ? $J=\{\#($ mset-lit $($ subst-lit $x \sigma)) . x \in \#(m s e t-s e t\{L 1\}) \#\}$
have $(\forall k \in$ set-mset ? $K . \exists j \in$ set-mset ? $J .(k, j) \in($ mult trm-ord $))$ by auto
from this d1 d2 $\langle\{\#($ mset-lit $($ subst-lit $x \sigma)) . x \in \#($ mset-set $\{L 1\}) \#\} \neq$ \{\#\},
have $o$ :
(\{\#mset-lit (subst-lit $x \sigma$ ). $x \in \#$ mset-set ? $C \#\}$,
\{\#mset-lit (subst-lit x $\sigma$ ). $x \in \#$ mset-set (cl-ecl P1)\#\})
$\in$ mult (mult trm-ord)
using mult-trm-ord-trans one-step-implies-mult [of
$\{\#($ mset-lit (subst-lit $x \sigma)) . x \in \#($ mset-set $\{L 1\}) \#\}$
$\{\#($ mset-lit (subst-lit $x \sigma)) . x \in \#\left(\right.$ mset-set ? $\left.\left.C^{\prime}\right) \#\right\}$ mult trm-ord
$\{\#($ mset-lit $($ subst-lit $x \sigma)) \cdot x \in \#($ mset-set $((c l-e c l P 1)-\{L 1\})) \#\}]$
by auto
from this $\langle C l-P=($ cl-ecl $P 1)\rangle\left\langle C^{\prime}=((C l-P-\{L 1\}))\right\rangle$ and $\langle P 1 \in P\rangle$ and hyp show False by auto
qed

```
have not-fact: \neg(\existsP1.(P1 \in P ^ factorization P1 C \sigma Ground C')}
proof
    assume ( }\existsP1.(P1\inP\wedge factorization P1 C \sigma Ground C'))
    then obtain P1 where P1 \inP factorization P1 C \sigma Ground C' by auto
    from〈factorization P1 C \sigma Ground C'> obtain L1 L2 L'ts u v Cl-P Cl-C
where
    (eligible-literal L1 P1 \sigma)
    (L1 \in (cl-ecl P1)) (L2 G (cl-ecl P1) - {L1 }) (Cl-C= (cl-ecl C)) (Cl-P=
(cl-ecl P1))
```

    (orient-lit-inst L1 t s pos \(\sigma\) )
    (orient-lit-inst L2 u v pos \(\sigma\) )
    \(((\) subst \(t \sigma) \neq(\) subst \(s \sigma))\)
    \(((\) subst \(t \sigma) \neq(\) subst \(v \sigma))\)
    (ck-unifier t u \(\sigma\) Ground)
        \(\left(L^{\prime}=\operatorname{Neg}(E q s v)\right)\)
        \(\left(C l-C=\left(\right.\right.\) subst-cl \(\left.\left.\left((C l-P-\{L 2\}) \cup\left\{L^{\prime}\right\}\right)\right) \sigma\right)\)
        \(\left(C^{\prime}=\left((C l-P-\{L 2\}) \cup\left\{L^{\prime}\right\}\right)\right)\)
        unfolding factorization-def by blast
    from \(\langle P 1 \in P\rangle\) and \(\operatorname{assms}(2)\) and \(\langle P \subseteq S\rangle\) have finite (cl-ecl P1) by auto
    from \(\operatorname{assms}(3)\) and \(\langle P 1 \in P\rangle\) have ground-clause (subst-cl (cl-ecl P1) \(\sigma\) )
    unfolding grounding-set-def by auto
from this have vars-of-cl (subst-cl (cl-ecl P1) $\sigma$ ) $=\{ \}$ by auto
from $\langle L 1 \in($ cl-ecl P1) $\rangle$ have $($ subst-lit L1 $\sigma) \in($ subst-cl (cl-ecl P1) $\sigma$ ) by auto
from this and «vars-of-cl (subst-cl (cl-ecl P1) $\sigma$ ) $=\{ \}$ 〉 have vars-of-lit (subst-lit
L1 $\sigma$ ) $=\{ \}$
by auto

```
    from <orient-lit-inst L1 t s pos \sigma` have
        orient-lit (subst-lit L1 \sigma) (subst t \sigma)(subst s \sigma) pos
        using lift-orient-lit by auto
    from this and <vars-of-lit (subst-lit L1 \sigma) ={}> have vars-of (subst t \sigma) =
{}
    using orient-lit-vars by blast
    from <orient-lit (subst-lit L1 \sigma) (subst t \sigma) (subst s \sigma) pos`
        and <vars-of-lit (subst-lit L1 \sigma)={}> have vars-of (subst s \sigma)={}
        using orient-lit-vars by blast
    from <(L2 \in (cl-ecl P1) - {L1 })> have L2 \in (cl-ecl P1) by auto
    from <L2 \in (cl-ecl P1)\rangle have (subst-lit L2 \sigma) \in(subst-cl (cl-ecl P1) \sigma) by
auto
    from this and «vars-of-cl (subst-cl (cl-ecl P1) \sigma)={}` have vars-of-lit (subst-lit
L2 \sigma) = {}
    by auto
    from <orient-lit-inst L2 u v pos \sigma〉 have
        orient-lit (subst-lit L2 \sigma) (subst u \sigma) (subst v \sigma) pos
        using lift-orient-lit by auto
    from this and <vars-of-lit (subst-lit L2 \sigma) = {}> have vars-of (subst u \sigma) =
{}
        using orient-lit-vars by blast
    from <orient-lit (subst-lit L2 \sigma) (subst u \sigma) (subst v \sigma) pos`
        and <vars-of-lit (subst-lit L2 \sigma) = {}> have vars-of (subst v \sigma)={}
        using orient-lit-vars by blast
    from <ck-unifier t u \sigma Ground> have (subst t \sigma)=(subst u \sigma)
        unfolding ck-unifier-def Unifier-def by auto
    from <orient-lit (subst-lit L1 \sigma) (subst t \sigma) (subst s \sigma) pos`
        have ((subst t \sigma),(subst s \sigma)) # trm-ord
        unfolding orient-lit-def by auto
    from this and «(subst t \sigma)\not=(subst s \sigma)\rangle
        and «vars-of (subst t \sigma)={}><vars-of (subst s \sigma)={}>
        have ((subst s \sigma),(subst t \sigma)) \in trm-ord using trm-ord-ground-total
        unfolding ground-term-def by blast
    from this and <(subst t \sigma)=( subst u \sigma)> have
        ((subst s \sigma),(subst u \sigma)) \in trm-ord by auto
```

    from <orient-lit (subst-lit L2 \(\sigma\) ) (subst \(u \sigma)(\) subst \(v \sigma)\) pos〉
        have \(((\) subst \(u \sigma),(\) subst \(v \sigma)) \notin\) trm-ord
        unfolding orient-lit-def by auto
    from this and \(\langle(\) subst \(t \sigma) \neq(\) subst \(v \sigma)\rangle\)
        and \(\langle(\) subst \(t \sigma)=(\) subst \(u \sigma)\rangle\)
        and «vars-of (subst \(u \sigma)=\{ \}\rangle\langle\) vars-of (subst v \(\sigma\) ) \(=\{ \}\rangle\)
        have \(((\) subst \(v \sigma),(\) subst \(u \sigma)) \in\) trm-ord using trm-ord-ground-total
        unfolding ground-term-def by metis
    $$
\text { let } ? m C 1=\operatorname{mset-ecl}(P 1, \sigma)
$$

$$
\text { let } ? m C 2=\text { mset-ecl }(C, \sigma)
$$

from 〈L2 $\in($ cl－ecl P1）$\langle\langle$ finite $($ cl－ecl P1）$\rangle$
have mset－set $($ cl－ecl P1）$=$ mset－set $(($ cl－ecl P1）$-\{L 2\})+$ mset－set $\{$ L2 $\}$
using split－mset－set［of cl－ecl P1 cl－ecl P1－\｛ L2 \} \{ L2 \}] by blast
from this have d1：$\{\#($ mset－lit（subst－lit $x \sigma)) . x \in \#($ mset－set（cl－ecl P1）） \＃\}
$=\{\#($ mset－lit $($ subst－lit $x \sigma)) . x \in \#($ mset－set $(($ cl－ecl P1）$-\{$ L2 $\})) \#\}$ $+\{\#($ mset－lit（subst－lit $x \sigma)) \cdot x \in \#($ mset－set $\{L 2\}) \#\}$
using split－image－mset by auto
let $? C=\left(((\right.$ cl－ecl P1 $\left.)-\{L 2\}) \cup\left\{L^{\prime}\right\}\right)$
from 〈finite（cl－ecl P1）〉 have finite ？C by auto
let ${ }^{2} C^{\prime}=? C-(($ cl－ecl P1 $)-\{L 2\})$
from 〈finite？$C$ 〉 have finite ？$C^{\prime}$ by auto
have ？$C=(($ cl－ecl P1 $)-\{L 2\}) \cup ? C^{\prime}$ by auto
from〈finite $\left(\right.$ cl－ecl P1）〉〈finite？$\left.C^{\prime}\right\rangle$
have mset－set ？$C=m$ set－set $(($ cl－ecl P1 $)-\{$ L2 $\})+$ mset－set ？$C^{\prime}$ using split－mset－set［of ？C cl－ecl P1－\｛ L2 \} ? C ] by blast
from this have d2：$\{\#($ mset－lit（subst－lit $x \sigma)) . x \in \#($ mset－set ？$C) \#\}$ $=\{\#($ mset－lit $($ subst－lit $x \sigma)) . x \in \#($ mset－set $(($ cl－ecl P1）$-\{$ L2 $\})) \#\}$ $+\{\#($ mset－lit（subst－lit $x \sigma)) . x \in \#\left(\right.$ mset－set ？$\left.\left.C^{\prime}\right) \#\right\}$ using split－image－mset by auto
have $\{\#($ mset－lit $($ subst－lit $x \sigma)) . x \in \#($ mset－set $\{$ L2 $\}) \#\} \neq\{\#\}$ by auto
let $? K=\left\{\#(\right.$ mset－lit $($ subst－lit $x \sigma)) . x \in \#\left(\right.$ mset－set ？$\left.\left.C^{\prime}\right) \#\right\}$
let ？$J=\{\#($ mset－lit（subst－lit $x \sigma)) . x \in \#($ mset－set $\{L 2\}) \#\}$
have $(\forall k \in$ set－mset ？$K . \exists j \in$ set－mset ？$J .(k, j) \in$（mult trm－ord $)$ ）
proof
fix $k$ assume $k \in$ set－mset？$K$
from this have $k \in \#$ ？$K$ by simp
from this obtain $M^{\prime}$ where $M^{\prime} \in \#\left(\right.$ mset－set ？$\left.C^{\prime}\right)$ and $k=($ mset－lit（subst－lit $\left.M^{\prime} \sigma\right)$ ）
using image－mset－thm［of ？K $\lambda$ ． ．（mset－lit（subst－lit $x \sigma)$ ）（mset－set ？$\left.\left.C^{\prime}\right)\right]$ by metis
from $\left\langle M^{\prime} \in \#\left(\right.\right.$ mset－set ？$\left.\left.C^{\prime}\right)\right\rangle$ and $\left\langle\right.$ finite ？$\left.C^{\prime}\right\rangle$ have $M^{\prime} \in ? C^{\prime}$ by auto
have L2 $\in \#$（mset－set $\{$ L2 \}) by auto
from this have（mset－lit（subst－lit L2 $\sigma$ ）$\in$ \＃？J）by auto
from this have（mset－lit（subst－lit L2 $\sigma$ ）$\in$ set－mset ？J）by auto
have $\{\#($ mset－lit（subst－lit $x \sigma)) . x \in \#($ mset－set $\{L 2\}) \#\} \neq\{\#\}$ by auto
show $\exists j \in$ set－mset ？$J .(k, j) \in($ mult trm－ord $)$
proof－
from $\left\langle M^{\prime} \in ? C^{\prime}\right\rangle$ have $M^{\prime}=L^{\prime}$ by auto
from 〈orient-lit-inst L2 $u$ v pos $\sigma\rangle$
have $i$ : (mset-lit (subst-lit L2 $\sigma$ )) $=\{\#\}+\{\#($ subst $u \sigma),($ subst $v \sigma) \#\}$
unfolding orient-lit-inst-def using add.commute by force
from $\left\langle L^{\prime}=\operatorname{Neg}(E q s v)\right\rangle$
have $i i:\left(\right.$ mset-lit $\left(\right.$ subst-lit $\left.\left.L^{\prime} \sigma\right)\right)=$
$\{\#\}+\{\#($ subst $s \sigma),($ subst $s \sigma),($ subst $v \sigma),($ subst $v \sigma) \#\}$
by force
have $\{\#($ subst $u \sigma),($ subst $v \sigma) \#\} \neq\{\#\}$ by auto
have $\left(\forall k^{\prime} \in\right.$ set-mset $\{\#($ subst $s \sigma),($ subst $s \sigma),($ subst $v \sigma),($ subst $v \sigma) \#\}$.
$\exists j^{\prime} \in$ set-mset $\{\#($ subst $u \sigma),($ subst $v \sigma) \#\} .\left(k^{\prime}, j^{\prime}\right) \in($ trm-ord $\left.)\right)$

## proof

fix $k^{\prime}$ assume $n h: k^{\prime} \in$ set-mset $\{\#($ subst $s \sigma),($ subst $s \sigma),($ subst $v \sigma)$,
(subst $v \sigma$ ) \#\}
have (subst $u \sigma) \in$ set-mset $\{\#($ subst $u \sigma)$, (subst $v \sigma$ ) \#\} by auto
from $n h$ have $k^{\prime}=($ subst $s \sigma) \vee k^{\prime}=($ subst $v \sigma)$ by auto
then show $\exists j^{\prime} \in$ set-mset $\{\#($ subst $u \sigma)$, (subst $\left.v \sigma) \#\right\} .\left(k^{\prime}, j^{\prime}\right) \in$
(trm-ord)
proof
assume $k^{\prime}=($ subst s $\sigma)$
from this and $\langle(($ subst $s \sigma),($ subst $u \sigma)) \in$ trm-ord $\rangle$ and $\langle($ subst $u \sigma) \in$ set-mset $\{\#($ subst $u \sigma)$, (subst $v \sigma) \#\}$ ) show
?thesis by auto
next
assume $k^{\prime}=($ subst $v \sigma)$
from this and $\langle(($ subst $v \sigma),($ subst $u \sigma)) \in$ trm-ord $\rangle$
and $\langle($ subst $u \sigma) \in$ set-mset $\{\#($ subst $u \sigma)$, (subst $v \sigma) \#\}$ show
?thesis by auto
qed
qed
from this $i$ ii $\langle\{\#($ subst $u \sigma),($ subst $v \sigma) \#\} \neq\{\#\} 〉$
have (mset-lit (subst-lit $L^{\prime} \sigma$ ),
(mset-lit (subst-lit L2 $\sigma$ ) )) $\in($ mult trm-ord $)$
using one-step-implies-mult $[$ of $\{\#($ subst $u \sigma)$, (subst $v \sigma) \#\}$ $\{\#($ subst $s \sigma),($ subst $s \sigma),($ subst $v \sigma),($ subst $v \sigma) \#\}$ trm-ord \{\#\}]
trm-ord-trans by metis
from this $\left\langle M^{\prime}=L^{\prime}\right\rangle\left\langle k=\left(\right.\right.$ mset-lit (subst-lit $\left.\left.\left.M^{\prime} \sigma\right)\right)\right\rangle$ show ?thesis by auto qed
qed
from this d1 d2 $\langle\{\#($ mset-lit (subst-lit $x \sigma)$ ). $x \in \#($ mset-set $\{L 2\}) \#\} \neq$ $\{\#\}$ >
have $o$ :
(\{\#mset-lit (subst-lit $x \sigma$ ). $x \in \#$ mset-set ? $C \#\}$,
\{\#mset-lit (subst-lit $x \sigma$ ). $x \in \#$ mset-set (cl-ecl P1)\#\})
$\in$ mult (mult trm-ord)
using mult-trm-ord-trans one-step-implies-mult [of
$\{\#($ mset-lit (subst-lit $x \sigma)) . x \in \#($ mset-set $\{L 2\}) \#\}$
$\{\#($ mset-lit (subst-lit $x \sigma)), x \in \#$ (mset-set ? $\left.\left.C^{\prime}\right) \#\right\}$ mult trm-ord
$\{\#($ mset-lit $($ subst-lit $x \sigma)) \cdot x \in \#($ mset-set $((c l-e c l P 1)-\{L 2\})) \#\}]$
by metis
from this $\langle(C l-P=($ cl-ecl P1 $))\rangle\left\langle C^{\prime}=\left((C l-P-\{L 2\}) \cup\left\{L^{\prime}\right\}\right)\right\rangle$ and $\langle P 1$ $\in P_{\text {> }}$
and hyp show False by auto
qed
from not-sup not-ref not-fact and assms(1) show False unfolding derivable-def by blast
qed
lemma redundant-inference-clause:
assumes redundant-clause E S $\sigma C^{\prime}$
assumes derivable $C P S \sigma$ Ground $C^{\prime}$
assumes grounding-set $P \sigma$
assumes $\forall x \in S$. (finite (cl-ecl $x$ ))
shows redundant-inference ESP $\sigma$
proof -
from $\operatorname{assms}(1)$ obtain $S^{\prime}$ where $S^{\prime} \subseteq($ instances $S)$
(set-entails-clause (clset-instances $\left.S^{\prime}\right)($ cl-ecl $\left.E)\right)$
( $\forall x \in S^{\prime}$. ( subterms-inclusion (subst-set (trms-ecl (fst x)) (snd $\left.x\right)$ )
(trms-ecl E))) and
ball-S'- $C^{\prime}$-le: $\forall x \in S^{\prime} .\left(\right.$ mset-ecl $(f s t ~ x$, snd $x)$, mset-cl $\left.\left(C^{\prime}, \sigma\right)\right) \in$ mult (mult trm-ord) $\vee$
mset-ecl $($ fst $x$, snd $x)=\operatorname{mset}-c l\left(C^{\prime}, \sigma\right)$
unfolding redundant-clause-def by auto
from $\operatorname{assms}(3) \operatorname{assms}(4)\left\langle d e r i v a b l e ~ C P S \sigma\right.$ Ground $C^{\prime} 〉$
obtain $D$ where $D \in P$
$\left(\left(\left(\right.\right.\right.$ mset-cl $\left.\left(C^{\prime}, \sigma\right)\right),($ mset-ecl $\left.(D, \sigma))\right) \in($ mult $($ mult trm-ord $\left.))\right)$
using conclusion-is-smaller-than-premisses by blast
have $\forall x \in S^{\prime} . \exists D^{\prime} \in \operatorname{cl-ecl}$ ' $P .\left((\operatorname{cl-ecl}(f s t x)\right.$, snd $\left.x),\left(D^{\prime}, \sigma\right)\right) \in c l$-ord proof (intro ballI)
fix $x$ assume $x \in S^{\prime}$
have $(($ cl-ecl $(f s t x)$, snd $x),($ cl-ecl $D, \sigma)) \in$ cl-ord
using ball-S $S^{\prime}$ - $C^{\prime}$-le[rule-format, $\left.O F\left\langle x \in S^{\prime}\right\rangle\right]$
using 〈(mset-cl $\left(C^{\prime}, \sigma\right)$, mset-ecl $\left.(D, \sigma)\right) \in$ mult (mult trm-ord) $\rangle$
unfolding cl-ord-def mem-Collect-eq prod.case mset-ecl-conv
by (metis mult-mult-trm-ord-trans[THEN transD])
with $\langle D \in P\rangle$ show $\exists D^{\prime} \in \operatorname{cl-ecl}{ }^{\prime} P .\left((\operatorname{cl-ecl}(f s t x)\right.$, snd $\left.x),\left(D^{\prime}, \sigma\right)\right) \in$ cl-ord
by auto
qed
from this and $\left\langle S^{\prime} \subseteq(\right.$ instances $\left.S)\right\rangle$ and $\left\langle\left(\right.\right.$ set-entails-clause (clset-instances $\left.S^{\prime}\right)$ (cl-ecl E))>
and $\left\langle\left(\forall x \in S^{\prime}\right.\right.$. ( subterms-inclusion (subst-set $($ trms-ecl $(f$ st $x))($ snd $\left.x)\right)$
(trms-ecl E)))>

```
    show ?thesis unfolding redundant-inference-def
    by auto
qed
lemma clause-saturated-and-inference-saturated:
    assumes clause-saturated S
    assumes }\forallx\inS\mathrm{ .(finite (cl-ecl x))
    shows inference-saturated S
proof (rule ccontr)
    assume \neg inference-saturated S
    then obtain CP\sigma C'D \vartheta \eta
    where derivable C P S \sigma Ground C' ground-clause (cl-ecl C)
                derivable D P S \vartheta FirstOrder C'trms-subsumes D C \eta
                \doteq\vartheta\diamond\eta grounding-set P\sigma
                \negredundant-inference (subst-ecl D \eta) S P \sigma
        unfolding inference-saturated-def by blast
    from assms(2)<grounding-set P \sigma〉<derivable C P S \sigma Ground C'〉
        \checkmarkredundant-inference (subst-ecl D \eta) S P \sigma`
        have \negredundant-clause (subst-ecl D \eta) S \sigma C'
        using redundant-inference-clause by blast
    from assms(1) have }\CP\sigma\mp@subsup{C}{}{\prime}D \vartheta \eta
        (derivable C P S \sigma Ground C')}\longrightarrow(\mathrm{ ground-clause (cl-ecl C))
        \longrightarrow ( \text { derivable D P S v FirstOrder C')} \longrightarrow ( \text { trms-subsumes D C } \eta \text { )}
        \longrightarrow ( \sigma \doteq \vartheta \diamond \eta )
        \longrightarrow ( r e d u n d a n t - c l a u s e ~ ( s u b s t - e c l ~ D ~ \eta ) S ~ \sigma ~ C ' ) ~ u n f o l d i n g ~ c l a u s e - s a t u r a t e d - d e f ~
by blast
    from this and «derivable C PS \sigma Ground C'〉<ground-clause (cl-ecl C)〉
        <derivable D P S \vartheta FirstOrder C'`
        <trms-subsumes D C \eta\rangle\langle\sigma\doteq\vartheta\diamond \ assms(1) have redundant-clause (subst-ecl
D \eta) S\sigma C'
        by auto
    from this and «\negredundant-clause (subst-ecl D \eta) S\sigma C'` show False by auto
qed
```


## 7 Refutational Completeness

We prove that our variant of the superposition calculus is complete under the redundancy criteria defined above．This is done as usual，by constructing a model of every saturated set not containing the empty clause．

## 7．1 Model Construction

We associate as usual every set of extended clauses with an interpretation． The interpretation is constructed in such a way that it is a model of the set of clauses if the latter is saturated and does not contain the empty clause．The
interpretation is constructed by defining directly a normalization function mapping every term to its normal form, i.e., to the minimal equivalent term. Note that we do not consider sets of rewrite rules explicitly.

The next function associates every normalization function with the corresponding interpretation (two terms are in relation if they share the same normal form). The obtained relation is an interpretation if the normalization function is compatible with the term combination operator.

```
definition same-values : \(:\left({ }^{\prime}\right.\) a trm \(\Rightarrow{ }^{\prime}\) 'a trm) \(\Rightarrow{ }^{\prime}\) 'a trm \(\Rightarrow{ }^{\prime}\) a trm \(\Rightarrow\) bool
    where (same-values \(f\) ) \(=\)
                \((\lambda x y .(f x)=(f y))\)
definition value-is-compatible-with-structure :: ('a trm \(\Rightarrow{ }^{\prime}\) 'a trm) \(\Rightarrow\) bool
    where (value-is-compatible-with-structure \(f)=(\forall t s .(f(\operatorname{Comb} t s))=(f(C o m b\)
\((f t)(f s)))\) )
```

```
lemma same-values-fo-int:
    assumes value-is-compatible-with-structure \(f\)
    shows fo-interpretation (same-values \(f\) )
proof -
    let \(? I=(\) same-values \(f)\)
    have ref: reflexive ?I unfolding same-values-def reflexive-def by simp
    have sym: symmetric ?I unfolding same-values-def symmetric-def by auto
    have trans: transitive ?I unfolding same-values-def transitive-def by auto
    from \(\operatorname{assms}(1)\) have comp: compatible-with-structure ?I
        unfolding same-values-def
            compatible-with-structure-def value-is-compatible-with-structure-def [of f]
    by metis
    from ref trans sym comp have congruence ?I unfolding congruence-def equiv-
alence-relation-def
    by auto
    then show ?thesis unfolding fo-interpretation-def by auto
qed
```

The normalization function is defined by mapping each term to a set of pairs. Intuitively, the second element of each pair represents the right hand side of a rule that can be used to rewrite the considered term, and the first element of the pair denotes its normal form. The value of the term is the first component of the pair with the smallest second component.

The following function returns the set of values for which the second component is minimal. We then prove that this set is non-empty and define a function returning an arbitrary chosen element.

```
definition min-trms :: ('a trm × 'a trm) set }=>\mathrm{ ' 'a trm set
    where (min-trms E) = ({ x. (\exists pair. (pair }\in
    \wedge(\forall\mp@subsup{pair}{}{\prime}\inE.(\mathrm{ snd pair'},\mathrm{ snd pair })\not\in\mathrm{ trm-ord }))\wedgex=\mathrm{ fst pair })})
```

lemma min-trms-not-empty:

```
    assumes \(E \neq\{ \}\)
    shows min-trms \(E \neq\{ \}\)
proof -
    from \(\operatorname{assms}(1)\) obtain \(x\) where \(x \in E\) by auto
    let ?pair-ordering \(=\{(x, y) .((\) snd \(x),(\) snd \(y)) \in\) trm-ord \(\}\)
    from trm-ord-wf have wf ?pair-ordering using measure-wf by auto
    from this \(\langle x \in E\rangle\)
        obtain \(y\) where \(y \in E\) and \(\forall z .(z, y) \in\) ?pair-ordering \(\longrightarrow(z \notin E)\)
        using wfE-min [of ?pair-ordering]
        by metis
    from this have fst \(y \in\) min-trms \(E\) unfolding min-trms-def by blast
    then show ?thesis by auto
qed
definition get-min :: 'a trm \(\Rightarrow\left({ }^{\prime}\right.\) a trm \(\times\) 'a trm) set \(\Rightarrow{ }^{\prime}\) a trm
    where \((\) get-min \(t E)=\)
        (if \(((\) min-trms \(E)=\{ \})\) then \(t\) else \((S O M E x .(x \in\) min-trms \(E)))\)
```

We now define the normalization function. The definition is tuned to make the termination proof straightforward. We will reformulate it afterward to get a simpler definition.
We first test whether a subterm of the considered term is reducible. If this is the case then the value can be obtained by applying recursively the function on each subterm, and then again on the term obtained by combining the obtained normal forms. If not, then we collect all possible pairs (as explained above), and we use the one with the minimal second component. These pairs can be interpreted as rewrite rules, giving the value of the considered term: the second component is the right-hand side of the rule and the first component is the normal form of the right-hand side. As usual, such rewrite rules are obtained from ground clauses that have a strictly positive maximal literal, no selected literals, and that are not validated by the constructed interpretation.

```
function trm-rep:: 'a trm \(\Rightarrow\left({ }^{\prime} a\right.\) eclause set \(\Rightarrow{ }^{\prime} a\) trm \()\)
    where
        (trm-rep \(t)=\)
            ( \(\lambda\) S. (if \(((\) is-compound \(t) \wedge((\) lhs \(t), t) \in\) trm-ord \(\wedge((r h s t), t) \in\) trm-ord
            \(\wedge(((\) lhs \(t, t) \in\) trm-ord \(\longrightarrow\) (trm-rep (lhs \(t) S) \neq(\) lhs \(t))\)
            \(\vee((\) rhs \(t, t) \in\) trm-ord \(\longrightarrow(\) trm-rep \((\) rhs \(t) S) \neq(\) rhs \(t))))\)
            then (if \(((\) Comb (trm-rep (lhs t) S) (trm-rep (rhs t) S)), t) \(\in\) trm-ord
                    then
                    (trm-rep (Comb (trm-rep (lhst)S)(trm-rep (rhst)S)) S)
                    else t)
            else (get-min \(t\)
            \(\left\{\right.\) pair. \(\exists z C C C^{\prime} C\) s \(L L^{\prime} \sigma t^{\prime} s^{\prime}\).
                pair \(=(z, s)\)
                    \(\wedge C C \in S \wedge(t \notin(\) subst-set (trms-ecl CC) \(\sigma))\)
                    \(\wedge\left(\forall x .\left(\exists x^{\prime} \in(\right.\right.\) trms-ecl \(C C)\). occurs-in \(x\left(\right.\) subst \(\left.\left.x^{\prime} \sigma\right)\right)\)
```

```
        \(\longrightarrow((x, t) \in\) trm-ord \(\longrightarrow(\) trm-rep \(x S)=x))\)
        \(\wedge\left(C^{\prime}=(\right.\) cl-ecl \(\left.C C)\right) \wedge(s, t) \in\) trm-ord \(\wedge((s, t) \in\) trm-ord \(\longrightarrow(z=\) trm-rep
\(s S)\) )
    \(\wedge\left(\right.\) orient-lit-inst \(L^{\prime} t^{\prime} s^{\prime}\) pos \(\left.\sigma\right) \wedge\left(\right.\) sel \(\left.C^{\prime}\right)=\{ \} \wedge\left(L^{\prime} \in C^{\prime}\right)\)
    \(\wedge(\) maximal-literal \(L C) \wedge\left(L=\left(\right.\right.\) subst-lit \(\left.\left.L^{\prime} \sigma\right)\right) \wedge\left(C=\left(\right.\right.\) subst-cl \(\left.\left.C^{\prime} \sigma\right)\right)\)
    \(\wedge(\) ground-clause \(C) \wedge\left(t=\left(\right.\right.\) subst \(\left.\left.t^{\prime} \sigma\right)\right) \wedge\left(s=\left(\right.\right.\) subst \(\left.\left.s^{\prime} \sigma\right)\right) \wedge\left(\right.\) finite \(\left.C^{\prime}\right)\)
    \(\wedge\)
    ( \(\forall L u v\).
        ( \(L \in C \longrightarrow\) orient-lit \(L u v\) pos
            \(\longrightarrow(u, t) \in\) trm-ord \(\longrightarrow(v, t) \in\) trm-ord
            \(\longrightarrow(\) trm-rep u \(S) \neq(\) trm-rep \(v S))\)
    \(\wedge\)
        \((L \in C \longrightarrow\) orient-lit \(L\) u v neg \(\longrightarrow(u, t) \in\) trm-ord \(\longrightarrow(v, t) \in\) trm-ord
            \(\longrightarrow(\) trm-rep u \(S)=(\) trm-rep \(v S))\) )
    \(\wedge\left(\forall s^{\prime \prime}\right.\). \((\)
        (eq-occurs-in-cl t s" \(\left.\left(C^{\prime}-\left\{L^{\prime}\right\}\right) \sigma\right) \longrightarrow\left(s^{\prime \prime}, t\right) \in\) trm-ord \(\longrightarrow(s, t) \in\)
trm-ord
\[
\left.\left.\left.\left.\left.\left.\longrightarrow(\text { trm-rep } s S) \neq\left(\text { trm-rep } s^{\prime \prime} S\right)\right)\right)\right\}\right)\right)\right)
\]
by auto
termination apply (relation trm-ord)
by auto (simp add: trm-ord-wf)
```

We now introduce a few shorthands and rewrite the previous definition into an equivalent simpler form. The key point is to prove that a term is always greater than its normal form.
definition subterm-reduction-aux:: 'a eclause set $\Rightarrow$ ' $a$ trm $\Rightarrow$ 'a trm where
subterm-reduction-aux $S t=$
(if $(($ Comb (trm-rep (lhs t) $S)($ trm-rep $($ rhs $t) S)), t) \in$ trm-ord then (trm-rep (Comb (trm-rep (lhst) $S$ ) (trm-rep (rhst) $S$ )) $S$ ) else $t$ )
definition subterm-reduction:: 'a eclause set $\Rightarrow{ }^{\prime}$ ' trm $\Rightarrow$ ' $a$ trm where
subterm-reduction $S t=$
(trm-rep (Comb (trm-rep (lhst) S) (trm-rep (rhst) S)) S)
definition maximal-literal-is-unique
where (maximal-literal-is-unique $t s C^{\prime} L^{\prime} S \sigma$ ) $=$
( $\forall s^{\prime \prime}$. (
(eq-occurs-in-clt $\left.s^{\prime \prime}\left(C^{\prime}-\left\{L^{\prime}\right\}\right) \sigma\right) \longrightarrow\left(s^{\prime \prime}, t\right) \in$ trm-ord $\longrightarrow(s, t) \in$
trm-ord

$$
\left.\left.\longrightarrow(\text { trm-rep } s S) \neq\left(\text { trm-rep } s^{\prime \prime} S\right)\right)\right)
$$

definition smaller-lits-are-false
where (smaller-lits-are-false $t C S$ ) $=$
( $\forall L u v$.

$$
\begin{aligned}
& (L \in C \longrightarrow \text { orient-lit } L u v \text { pos } \\
& \quad \longrightarrow(u, t) \in \text { trm-ord } \longrightarrow(v, t) \in \text { trm-ord }
\end{aligned}
$$

$$
\begin{aligned}
& \longrightarrow(\text { trm-rep } u S) \neq(\text { trm-rep } v S)) \\
& \\
(L \in & C \text { orient-lit } L u \text { v neg } \longrightarrow(u, t) \in \text { trm-ord } \longrightarrow(v, t) \in \text { trm-ord } \\
& \longrightarrow(\text { trm-rep } u S)=(\text { trm-rep } v S)))
\end{aligned}
$$

```
definition int-clset
    where int-clset \(S=(\) same-values \((\lambda x\). \((\) trm-rep \(x S)))\)
lemma smaller-lits-are-false-if-cl-not-valid:
    assumes \(\neg\) (validate-ground-clause (int-clset S) C)
    shows smaller-lits-are-false t CS
proof (rule ccontr)
    let ?I \(=\) int-clset \(S\)
    assume \(\neg\) smaller-lits-are-false \(t C S\)
    from this obtain \(L u v\) where \(L \in C\)
        and (orient-lit L u v pos \(\wedge(\) trm-rep u \(S)=(\) trm-rep v \(S\) ) )
            \(\vee(\) orient-lit L u v neg \(\wedge(\) trm-rep u \(S) \neq(\) trm-rep \(v S))\)
        unfolding smaller-lits-are-false-def by blast
    then have (orient-lit \(L u\) v pos \(\wedge(\) trm-rep \(u S)=(\) trm-rep v S) )
        \(\vee(\) orient-lit \(L\) u v neg \(\wedge(\) trm-rep \(u S) \neq(\) trm-rep \(v S))\) by blast
    then show False
    proof
        assume c-pos: (orient-lit Luvpos \(\wedge(\) trm-rep u \(S)=(\) trm-rep \(v S))\)
    then have orient-lit \(L u v\) pos by blast
    from c-pos have (trm-rep u \(S\) ) \(=(\) trm-rep \(v S)\) by blast
    from <orient-lit \(L u v\) pos \(\rangle\) have \(L=(\operatorname{Pos}(E q u v)) \vee L=(\operatorname{Pos}(E q v u))\)
        unfolding orient-lit-def by auto
    from this and 〈(trm-rep u \(S)=(\) trm-rep \(v S)\rangle\) have validate-ground-lit ?I \(L\)
        using validate-ground-lit.simps(1) validate-ground-eq.simps
        unfolding same-values-def int-clset-def
        by (metis (mono-tags, lifting))
    from this and \(\langle L \in C\rangle\) and assms show False unfolding int-clset-def
        using validate-ground-clause.simps by blast
    next
        assume c-neg: (orient-lit Luv neg \(\wedge(\) trm-rep u \(S) \neq(\) trm-rep \(v S))\)
        then have orient-lit \(L u v\) neg by blast
    from \(c\)-neg have (trm-rep \(u S) \neq(\) trm-rep \(v S)\) by blast
    from <orient-lit Luvneg〉 have \(L=(N e g(E q u v)) \vee L=(N e g(E q v u))\)
        unfolding orient-lit-def by auto
    from this and \(\langle(\) trm-rep \(u S) \neq(\) trm-rep \(v S)\) 〉 have validate-ground-lit ?I \(L\)
        using validate-ground-lit.simps(2) validate-ground-eq.simps
        unfolding same-values-def int-clset-def
        by (metis (mono-tags, lifting))
    from this and \(\langle L \in C\rangle\) and assms show False unfolding int-clset-def
        using validate-ground-clause.simps by blast
    qed
qed
```

The following function states that all instances of the terms attached to a
clause are in normal form w.r.t. the interpretation associated with $S$, up to some maximal term $t$

```
definition trms-irreducible
    where trms-irreducible CC \(\sigma S t=\)
                \(\left(\forall x .\left(\exists x^{\prime} \in\left(\right.\right.\right.\) trms-ecl CC). occurs-in \(x\left(\right.\) subst \(\left.\left.x^{\prime} \sigma\right)\right) \longrightarrow\)
                        \(((x, t) \in\) trm-ord \(\longrightarrow(\) trm-rep \(x S)=x))\)
```

lemma trms-irreducible-lemma:
assumes all-trms-irreducible (subst-set (trms-ecl C) $\sigma$ ) ( $\lambda$ t. trm-rep $t S)$
shows trms-irreducible $C \sigma S t$
proof (rule ccontr)
assume $\neg$ trms-irreducible $C \sigma S t$
from this obtain $x$ where $\exists x^{\prime} \in$ trms-ecl C. occurs-in $x$ (subst $x^{\prime} \sigma$ ) and
trm-rep $x S \neq x$ unfolding trms-irreducible-def by blast
from $\left\langle\exists x^{\prime} \in\right.$ trms-ecl $C$. occurs-in $x\left(\right.$ subst $x^{\prime} \sigma$ )〉 obtain $x^{\prime}$ where
$x^{\prime} \in$ trms-ecl $C$ and occurs-in $x\left(\right.$ subst $x^{\prime} \sigma$ ) by blast
from $\left\langle x^{\prime} \in\right.$ trms-ecl $\left.C\right\rangle$
have (subst $\left.x^{\prime} \sigma\right) \in($ subst-set (trms-ecl C) $\sigma)$
by auto
from this and assms(1) <occurs-in $x\left(\right.$ subst $x^{\prime} \sigma$ )〉
have trm-rep $x S=x$ unfolding all-trms-irreducible-def by metis
from this and $\langle$ trm-rep $x S \neq x\rangle$ show False by blast
qed

The following predicate states that a term $z$ is the normal form of the righthand side of a rule of left-hand side $t$. It is used to define the set of possible values for term $t$. The actual value is that corresponding to the smallest right-hand side.

```
definition candidate-values
    where (candidate-values z CC C' \(C\) s L L \(\left.L^{\prime} \sigma t^{\prime} s^{\prime} t S\right)=\)
            \((C C \in S \wedge(t \notin(\) subst-set \((\) trms-ecl \(C C) \sigma)) \wedge(\) trms-irreducible \(C C \sigma S\)
t)
    \(\wedge\left(C^{\prime}=(\right.\) cl-ecl \(\left.C C)\right) \wedge(s, t) \in\) trm-ord \(\wedge((s, t) \in\) trm-ord \(\longrightarrow(z=\) trm-rep
\(s\) S)
    \(\wedge\left(\right.\) orient-lit-inst \(L^{\prime} t^{\prime} s^{\prime}\) pos \(\left.\sigma\right) \wedge\left(\right.\) sel \(\left.C^{\prime}=\{ \}\right) \wedge\left(L^{\prime} \in C^{\prime}\right) \wedge(\) maximal-literal
LC)
    \(\wedge\left(L=\left(\right.\right.\) subst-lit \(\left.\left.L^{\prime} \sigma\right)\right) \wedge\left(C=\left(\right.\right.\) subst-cl \(\left.\left.C^{\prime} \sigma\right)\right) \wedge(\) ground-clause \(C)\)
    \(\wedge\left(t=\left(\right.\right.\) subst \(\left.\left.t^{\prime} \sigma\right)\right) \wedge\left(s=\left(\right.\right.\) subst \(\left.\left.s^{\prime} \sigma\right)\right) \wedge\left(\right.\) finite \(\left.C^{\prime}\right)\)
    \(\wedge\) (smaller-lits-are-false t C S)
    \(\wedge\left(\right.\) maximal-literal-is-unique \(\left.\left.t s C^{\prime} L^{\prime} S \sigma\right)\right)\)
```

definition set-of-candidate-values:: 'a eclause set $\Rightarrow{ }^{\prime} a \operatorname{trm} \Rightarrow\left({ }^{\prime} a \operatorname{trm} \times\right.$ 'a trm) set
where set-of-candidate-values $S t=$
$\left\{\right.$ pair. $\exists z C C C^{\prime} C s L L^{\prime} \sigma t^{\prime} s^{\prime}$. pair $=(z, s) \wedge\left(\right.$ candidate-values $\left.\left.z C C C^{\prime} C s L L^{\prime} \sigma t^{\prime} s^{\prime} t S\right)\right\}$
definition subterm-reduction-applicable-aux:: 'a eclause set $\Rightarrow$ 'a trm $\Rightarrow$ bool
where subterm－reduction－applicable－aux $S t=$

$$
\begin{aligned}
& \text { (is-compound } t \wedge(\text { lhs } t, t) \in \text { trm-ord } \wedge(\text { rhs } t, t) \in \text { trm-ord } \\
& \wedge(((\text { lhs } t, t) \in \text { trm-ord } \longrightarrow(\text { trm-rep }(\text { lhs } t) S) \neq(\text { lhs } t)) \\
& \vee((\text { rhs } t, t) \in \text { trm-ord } \longrightarrow(\text { trm-rep }(\text { rhs } t) S) \neq(\text { rhs } t))))
\end{aligned}
$$

definition subterm－reduction－applicable：：＇a eclause set $\Rightarrow$＇a trm $\Rightarrow$ bool
where subterm－reduction－applicable $S t=$
（is－compound $t \wedge(($ trm－rep $($ lhs $t) S) \neq($ lhs $t) \vee($ trm－rep $($ rhs $t) S) \neq($ rhs $t)))$
lemma trm－rep－is－lower－aux：
assumes $\forall y .(y, t) \in$ trm－ord $\longrightarrow$
$(y \neq($ trm－rep $y S) \longrightarrow(($ trm－rep $y S), y) \in$ trm－ord $)$
assumes（subterm－reduction－applicable $S t$ ）
shows $(($ Comb（trm－rep（lhs t）$S)($ trm－rep（rhst）$S)), t) \in$ trm－ord
proof－
have（lhs $t, t) \in$ trm－ord using 〈subterm－reduction－applicable $S$ t〉args－are－strictly－lower
subterm－reduction－applicable－def
by blast
have（rhs $t, t) \in$ trm－ord $\mathbf{u s i n g}$ 〈subterm－reduction－applicable $S t\rangle$ args－are－strictly－lower
subterm－reduction－applicable－def by blast
from $\operatorname{assms}(1)$ and $\langle(l h s t, t) \in t r m$－ord $\rangle$ have
$l:(($ lhs $t \neq($ trm－rep（lhs $t) S)) \longrightarrow(($ trm－rep （lhs $t) S),($ lhs $t)) \in$ trm－ord $)$
by metis
from $\operatorname{assms}(1)$ and $\langle(r h s t, t) \in$ trm－ord $\rangle$ have
$r:($ rhs $t \neq($ trm－rep（rhst）$S) \longrightarrow(($ trm－rep $($ rhs $t) S),($ rhs $t)) \in$ trm－ord $)$
by metis
from 〈subterm－reduction－applicable $S t\rangle$
have $(($ trm－rep $($ lhs $t) S) \neq($ lhs t）$\vee($ trm－rep $($ rhs $t) S) \neq($ rhs $t))$
unfolding subterm－reduction－applicable－def［of $S t]$ by blast
then show？thesis
proof
assume（trm－rep（lhs t）$S$ ）$\neq$（lhs t）
from this and $l$ have $(($ trm－rep（lhs $t) S),($ lhs $t)) \in$ trm－ord by metis
from this
have $i$ ：（（Comb（trm－rep（lhst）S）（trm－rep（rhst）S）），（Comb（lhst）（trm－rep
（rhs t）$S$ ）））
$\in$ trm－ord using trm－ord－reduction－left by blast
show（（Comb（trm－rep（lhst）S）（trm－rep（rhst）$S$ ）），$t) \in$ trm－ord
proof（cases）
assume（trm－rep（rhst）S）$=($ rhs $t)$
from this
and $\langle(($ Comb（trm－rep（lhs t）S）（trm－rep（rhst）S）），（Comb（lhs t）（trm－rep
（rhst）S）））
$\in$ trm－ord $\rangle$
show（（Comb（trm－rep（lhs t）S）（trm－rep（rhst）S）），t）$\in$ trm－ord by（metis assms（2）is－compound．elims（2）lhs．simps（1）
rhs．simps（1）subterm－reduction－applicable－def）

```
    next
        assume (trm-rep (rhs t)S)}\not=(\mathrm{ rhs t)
        from this and r have ((trm-rep (rhs t) S),(rhs t)) \in trm-ord by metis
    from this have ((Comb (lhst) (trm-rep (rhst)S)), ((Comb (lhs t) (rhs t))))
trm-ord
            using trm-ord-reduction-right by blast
    from this and i show ((Comb (trm-rep (lhs t)S) (trm-rep (rhst)S)),t)\in
trm-ord
            by (metis assms(2) is-compound.elims(2) lhs.simps(1) rhs.simps(1)
                subterm-reduction-applicable-def trm-ord-trans transE)
    qed
    next
        assume (trm-rep (rhs t)S)}\not=(\mathrm{ rhs t)
    from this and r have ((trm-rep (rhs t) S),(rhst))\in trm-ord by metis
    from this
        have i: ((Comb (trm-rep (lhs t)S) (trm-rep (rhs t) S)),(Comb (trm-rep (lhs
t) S)(rhs t)))
                trm-ord using trm-ord-reduction-right by blast
    show ((Comb (trm-rep (lhs t) S) (trm-rep (rhs t) S)),t)\in trm-ord
    proof (cases)
        assume (trm-rep (lhs t)S)=(lhs t)
        from this and
        «((Comb (trm-rep (lhs t) S) (trm-rep (rhs t) S)),(Comb (trm-rep (lhs t) S)
(rhs t))) \in trm-ord>
            show ((Comb (trm-rep (lhs t) S) (trm-rep (rhst)S)),t) \in trm-ord
                by (metis assms(2) basic-superposition.subterm-reduction-applicable-def
                basic-superposition-axioms is-compound.elims(2) lhs.simps(1) rhs.simps(1))
    next
            assume (trm-rep (lhs t) S)}\not=(\mathrm{ lhs t)
            from this and l have ((trm-rep (lhs t) S), (lhs t)) \in trm-ord by metis
            from this have ((Comb (trm-rep (lhst)S) (rhst)), ((Comb (lhs t) (rhst))))
                trm-ord
            using trm-ord-reduction-left by blast
            from this and i show ((Comb (trm-rep (lhs t)S) (trm-rep (rhs t)S)),t)\in
trm-ord
            by (metis assms(2) basic-superposition.subterm-reduction-applicable-def
                basic-superposition-axioms is-compound.elims(2) lhs.simps(1) rhs.simps(1)
                    trm-ord-trans transE)
        qed
    qed
qed
```

The following lemma corresponds to the initial definition of the function trm-rep.
lemma trm-rep-init-def:
shows (trm-rep t) $=(\lambda S$. (if (subterm-reduction-applicable-aux $S t)$
then (subterm-reduction-aux $S$ t)
else (get-min $t($ set-of-candidate-values $S t)))$ )
unfolding subterm－reduction－aux－def set－of－candidate－values－def candidate－values－def
subterm－reduction－applicable－aux－def maximal－literal－is－unique－def smaller－lits－are－false－def
trms－irreducible－def
using trm－rep．simps［of $t$ ］by force
lemma trm－rep－aux－def：
assumes $\forall y .(y, t) \in$ trm－ord $\longrightarrow$
$(y \neq($ trm－rep $y S) \longrightarrow(($ trm－rep y $S), y) \in$ trm－ord $)$
shows（trm－rep t $S$ ）$=($ if（subterm－reduction－applicable $S t)$ then（subterm－reduction $S$ t） else（get－min $t$（set－of－candidate－values $S t)$ ）
proof（cases）
assume subterm－reduction－applicable $S t$
then have subterm－reduction－applicable－aux $S t$
using args－are－strictly－lower
subterm－reduction－applicable－def subterm－reduction－applicable－aux－def by blast
from this have（trm－rep $t S)=($ subterm－reduction－aux $S t)$
using trm－rep－init－def［of $t$ ］by meson
then have（（Comb（trm－rep（lhs t）S）（trm－rep（rhs t）$S$ ）），t）$\in$ trm－ord
using «subterm－reduction－applicable $S$ t〉 assms trm－rep－is－lower－aux by blast
then show？thesis
by（metis $\langle$ trm－rep $t S=$ subterm－reduction－aux $S t\rangle$
〈subterm－reduction－applicable $S$ t $\rangle$
subterm－reduction－def subterm－reduction－aux－def）
next
assume $\neg$ subterm－reduction－applicable $S t$
then have $\neg$ subterm－reduction－applicable－aux $S t$
using subterm－reduction－applicable－def subterm－reduction－applicable－aux－def by blast
from this and $\neg \neg$ subterm－reduction－applicable $S$ t〉 show ？thesis
by（meson trm－rep－init－def）
qed
lemma trm－rep－is－lower：
shows $(t \neq($ trm－rep $t S)) \longrightarrow((($ trm－rep $t S), t) \in$ trm－ord $)($ is ？$P$ P $t)$
proof（（rule wf－induct［of trm－ord ？P t］$),($ simp add：trm－ord－wf）$)$
next
fix $x$ assume hyp－ind：$\forall y .(y, x) \in$ trm－ord $\longrightarrow(? P y)$
let ？$v=($ Comb（trm－rep（lhs x）S）（trm－rep（rhs x）S））
show（？P $x$ ）
proof（rule impI）
assume $x \neq($ trm－rep $x S)$
show $(($ trm－rep $x S), x) \in$ trm－ord
proof cases
assume c1：subterm－reduction－applicable $S$ x

```
            from this and hyp-ind
            have \((? v, x) \in\) trm-ord using trm-rep-is-lower-aux by metis
            from c1 and hyp-ind have (trm-rep \(x S)=(\) subterm-reduction \(S x)\)
            using trm-rep-aux-def [of \(x S\) ] by metis
            from this have (trm-rep x \(S\) ) \(=(\) trm-rep ?v \(S)\)
            unfolding subterm-reduction-def by metis
            from \(\langle(? v, x) \in\) trm-ord \(\rangle\) and hyp-ind have ?P ?v by metis
            from this and \(\langle(\) trm-rep \(x S)=(\) trm-rep ?v \(S)\rangle\) show ?thesis
            by (metis «(trm-rep (lhs x) \(S \cdot\) trm-rep (rhs x) \(S, x) \in\) trm-ord \(\rangle\) trm-ord-trans
transE)
                            next assume c2: ᄀsubterm-reduction-applicable \(S x\)
                            from \(c 2\) and hyp-ind have \((\) trm-rep \(x S)=(\) get-min \(x\) (set-of-candidate-values
\(S x)\) )
            using trm-rep-aux-def [of \(x S\) ] by metis
            from this and \(\langle x \neq(\) trm-rep \(x S)\rangle\)
                have (trm-rep \(x S\) ) \(\in\) (min-trms (set-of-candidate-values \(S\) x))
                            unfolding get-min-def by (metis (full-types) some-in-eq)
                            then obtain pair where pair \(\in\) (set-of-candidate-values \(S\) x) (trm-rep \(x\)
\(S)=\) fst pair
            unfolding min-trms-def by blast
                            from \(\langle\) pair \(\in\) (set-of-candidate-values \(S x)\rangle\)
            have
            \(\exists C C C^{\prime} C L L^{\prime} \sigma t^{\prime} s^{\prime}\). candidate-values (fst pair) CC \(C^{\prime} C\) (snd pair)
\(L L^{\prime} \sigma t^{\prime} s^{\prime} x S\)
            unfolding set-of-candidate-values-def by fastforce
                            from this have (snd pair, \(x\) ) \(\in\) trm-ord unfolding candidate-values-def by
blast
```


## from

```
                《 \(C C C^{\prime} C L L^{\prime} \sigma t^{\prime} s^{\prime}\). candidate-values (fst pair) CC \(C^{\prime} C\) (snd pair)
\(\left.L L^{\prime} \sigma t^{\prime} s^{\prime} x S\right\rangle\)
                have \(((\) snd pair,\(x) \in\) trm-ord \(\longrightarrow\) fst pair \(=\) trm-rep \((\) snd pair \() S)\)
                unfolding candidate-values-def by blast
            from \(\langle(\) snd pair,\(x) \in\) trm-ord \(\rangle\langle((\) snd pair,\(x) \in\) trm-ord \(\longrightarrow\) fst pair \(=\)
trm-rep (snd pair) S) >
            have fst pair \(=\) trm-rep (snd pair) \(S\) by blast
            from \(\langle(\) snd pair,\(x) \in\) trm-ord \(\rangle\) and hyp-ind have (?P (snd pair)) by blast
            from this and \(\langle\) fst pair \(=(\) trm-rep \((\) snd pair \() S)\rangle\)
                have fst pair \(=\) snd pair \(\vee(\) fst pair,snd pair \() \in\) trm-ord
                by metis
            from this and \(\langle(\) trm-rep \(x S)=\) fst pair \(\rangle\) and \(\langle(\) snd pair,\(x) \in\) trm-ord \(\rangle\langle x\)
\(\neq(\) trm-rep \(x S)\) >
                show ?thesis by (metis trm-ord-trans transD)
            qed
    qed
qed
lemma trm-rep-is-lower-subt-red:
    assumes (subterm-reduction-applicable \(S\) x)
    shows \(((\) trm-rep \(x S), x) \in\) trm-ord
```


## proof -

let $? v=($ Comb $($ trm-rep $($ lhs $x) S)($ trm-rep $($ rhs $x) S))$
from $\operatorname{assms}(1)$
have $(? v, x) \in$ trm-ord using trm-rep-is-lower-aux trm-rep-is-lower by metis
from $\operatorname{assms}(1)$ have (trm-rep $x S)=($ subterm-reduction $S x)$
using trm-rep-aux-def [of x S] trm-rep-is-lower by metis
from this have (trm-rep $x S)=($ trm-rep ?v $S)$
unfolding subterm-reduction-def by metis
have ?v $=$ trm-rep ?v $S \vee($ trm-rep ?v $S, ? v) \in$ trm-ord using trm-rep-is-lower by blast
from this and $\langle($ trm-rep $x S)=($ trm-rep ? $v S)\rangle$ show $((($ trm-rep $x S), x) \in$ trm-ord)
by (metis $\langle($ trm-rep (lhs $x$ ) $S \cdot$ trm-rep (rhs $x) S, x) \in$ trm-ord $\downarrow$ trm-ord-trans transE)
qed
lemma trm-rep-is-lower-root-red:
assumes $\neg$ (subterm-reduction-applicable $S x$ )
assumes min-trms (set-of-candidate-values $S x) \neq\{ \}$
shows $((($ trm-rep $x S), x) \in$ trm-ord $)$
proof -
from $\operatorname{assms}(1)$ have $($ trm-rep $x S)=($ get-min $x($ set-of-candidate-values $S x))$ using trm-rep-aux-def [of x S ] trm-rep-is-lower by metis
from this and assms(2) have (trm-rep x $S$ ) $\in$ (min-trms (set-of-candidate-values $S x)$ )
unfolding get-min-def by (metis (full-types) some-in-eq)
then obtain pair where pair $\in$ (set-of-candidate-values $S x$ ) and (trm-rep $x$ $S)=$ fst pair
unfolding min-trms-def by blast
from $\langle$ pair $\in$ (set-of-candidate-values $S x)\rangle$
have $\exists C C C^{\prime} C L L^{\prime} \sigma t^{\prime} s^{\prime}$. candidate-values (fst pair) CC $C^{\prime} C$ (snd pair)
$L L^{\prime} \sigma t^{\prime} s^{\prime} x S$
unfolding set-of-candidate-values-def by fastforce
from this have (snd pair,x) trm-ord unfolding candidate-values-def by blast
from $« \exists C C C^{\prime} C L L^{\prime} \sigma t^{\prime} s^{\prime}$. candidate-values (fst pair) CC $C^{\prime} C$ (snd pair)
$\left.L L^{\prime} \sigma t^{\prime} s^{\prime} x S\right\rangle$
have $(($ snd pair,$x) \in$ trm-ord $\longrightarrow$ fst pair $=$ trm-rep $($ snd pair $) S)$
unfolding candidate-values-def by blast
from $\langle($ snd pair,$x) \in$ trm-ord $\rangle$ and $\langle(($ snd pair, $x) \in$ trm-ord $\longrightarrow$ fst pair $=$ trm-rep (snd pair) $S$ ) >
have fst pair $=$ trm-rep (snd pair) $S$ by blast
have snd pair $=$ trm-rep (snd pair) $S \vee$ (trm-rep (snd pair) $S$,snd pair $) \in$ trm-ord using trm-rep-is-lower by blast
from this and $\langle($ snd pair,$x) \in$ trm-ord $\rangle$ have (trm-rep (snd pair) $S, x$ ) $\in$ trm-ord using trm-ord-trans trans-def by metis
from this and $\langle($ trm-rep $x S)=$ fst pair $\rangle$ and $\langle f s t$ pair $=$ trm-rep (snd pair) $S\rangle$ show ?thesis by metis

## qed

Finally, the next lemma gives a simpler and more convenient definition of the function trm-rep.

```
lemma trm-rep-simp-def:
    shows (trm-rep t S)=(if (subterm-reduction-applicable S t)
                                    then (subterm-reduction S t)
                                    else (get-min t (set-of-candidate-values S t)))
using trm-rep-is-lower trm-rep-aux-def by blast
```

We now establish some useful properties of the normalization function.

```
lemma trm-rep-involutive:
    shows (trm-rep (trm-rep tS)S) \(\begin{aligned} & \text { trm-rep } t S)(\text { is ?P } t) ~\end{aligned}\)
proof ((rule wf-induct [of trm-ord ?P \(t]),(\) simp add: trm-ord-wf \())\)
next
    fix \(x\) assume hyp-ind: \(\forall y .(y, x) \in\) trm-ord \(\longrightarrow(? P y)\)
    let \(? v=(\) Comb (trm-rep (lhs x) S) (trm-rep (rhs x) S) )
    show (?P \(x\) )
        proof cases
            assume c1: subterm-reduction-applicable \(S x\)
            from this and hyp-ind
                have \((? v, x) \in\) trm-ord \(\mathbf{u s i n g}\) trm-rep-is-lower-aux trm-rep-is-lower by
metis
            from this hyp-ind have (trm-rep (trm-rep ?v S) S) \(=(\) trm-rep ?v \(S)\)
            using trm-rep-aux-def [of \(x S\) ] by metis
            from \(c 1\) have trm-rep \(x S=\) trm-rep ?v \(S\)
            using trm-rep-simp-def [of \(x S\) ] unfolding subterm-reduction-def by metis
            from this and 〈(trm-rep (trm-rep ?v \(S\) ) \(S\) ) \(=(\) trm-rep ?v \(S)\rangle\langle\) trm-rep x \(S\)
\(=\) trm-rep ? \(v S\) >
                show ?thesis by metis
            next assume c2: \(\neg\) subterm-reduction-applicable \(S x\)
            from c2 have (trm-rep x \(S\) ) \(=(\) get-min \(x\) (set-of-candidate-values \(S x)\) )
                using trm-rep-simp-def [of \(x S\) ] by metis
            show ?thesis
            proof (rule ccontr)
                    assume \((\) trm-rep \((\) trm-rep \(x S) S) \neq(\) trm-rep \(x S)\)
                    from this have \(x \neq(\) trm-rep \(x S)\) by metis
                    from \(c 2\) and \(\langle x \neq(\) trm-rep \(x S)\rangle\)
                    have (trm-rep \(x S\) ) \(\in\) (min-trms (set-of-candidate-values \(S\) x) )
                    using trm-rep-simp-def [of \(x S\) ]
                    unfolding get-min-def by (metis (full-types) some-in-eq)
                    then obtain pair where
                    pair \(\in(\) set-of-candidate-values \(S x)\) and (trm-rep \(x S)=\) fst pair
                    unfolding min-trms-def by blast
                    from \(\langle\) pair \(\in\) (set-of-candidate-values \(S x)\rangle\)
                have \(i\) : \(\exists C C C^{\prime} C L L^{\prime} \sigma t^{\prime} s^{\prime}\).
                    candidate-values (fst pair) CC \(C^{\prime} C\) (snd pair) L \(L^{\prime} \sigma t^{\prime} s^{\prime} x S\)
                    unfolding set-of-candidate-values-def
                    by fastforce
```

from this have（snd pair，x）$\in$ trm－ord unfolding candidate－values－def

## by blast

from $i$ have $(($ snd pair,$x) \in$ trm－ord $\longrightarrow$ fst pair $=$ trm－rep $($ snd pair $)$ S）
unfolding candidate－values－def by blast
from 〈（snd pair,$x) \in$ trm－ord〉
and $\langle(($ snd pair,$x) \in$ trm－ord $\longrightarrow$ fst pair $=$ trm－rep $($ snd pair $) S)\rangle$
have fst pair $=$ trm－rep（snd pair）$S$ by blast
from $\langle(s n d$ pair,$x) \in$ trm－ord $\rangle$ and hyp－ind have（？P（snd pair））by blast from this and $\langle$ fst pair $=($ trm－rep $($ snd pair $) S)\rangle$ and $\langle($ trm－rep $x S)=$ fst pair＞
and $\langle($ trm－rep $($ trm－rep $x S) S) \neq($ trm－rep $x S)\rangle$
show False by metis
qed
qed
qed
The following predicate is true if all proper subterms are in normal form．

```
definition root-term \(::\) 'a eclause set \(\Rightarrow\) ' \(a\) trm \(\Rightarrow\) bool
    where
    \((\) root-term \(S t)=\)
        \(((\) trm-rep \(t S)=(\) get-min \(t(\) set-of-candidate-values \(S t)))\)
```

The following function checks that the considered term contains a subterm that can be reduced．

```
definition st-red :: 'a eclause set \(\Rightarrow{ }^{\prime}\) ' trm \(\Rightarrow\) bool
    where
    (st-red \(S t\) )
        \(=\left(\exists t^{\prime} p .\left(\left(\right.\right.\right.\) subterm t p \(\left.t^{\prime}\right) \wedge\left(\right.\) root-term \(\left.S t^{\prime}\right) \wedge\left(\right.\) trm-rep \(\left.\left.\left.t^{\prime} S \neq t^{\prime}\right)\right)\right)\)
```

lemma red-arg-implies-red-trm :
assumes st-red $S$ t1
assumes $t=($ Comb t1 t2 $) \vee t=($ Comb t2 t1 $)$
shows st-red $S t$
proof -
from assms(1) obtain $t^{\prime} p$ where subterm t1 $p t^{\prime}$ and root-term $S t^{\prime}$ and
trm-rep $t^{\prime} S \neq t^{\prime}$
unfolding st-red-def by blast
from 〈subterm t1 $p t^{\prime} 〉$ and $\operatorname{assms}(2)$ obtain $q$ where subterm $t q t^{\prime}$
by (metis subterm.simps(4) subterm.simps(5))
from this and $\left\langle\right.$ root-term $\left.S t^{\prime}\right\rangle$ and $\left\langle\right.$ trm-rep $\left.t^{\prime} S \neq t^{\prime}\right\rangle$
show ?thesis unfolding st-red-def by blast
qed
lemma subterms-of-irred-trms-are-irred:
(trm-rep $t S) \neq t \longrightarrow$ st-red $S t($ is $(? P t))$
proof ((rule wf-induct [of trm-ord ?P $t]),($ simp add: trm-ord-wf $))$
next
fix $x$ assume hyp-ind: $\forall y .(y, x) \in \operatorname{trm}$-ord $\longrightarrow(? P$ $y)$

```
    show (?P x)
    proof (rule impI)
    assume (trm-rep x S)}\not=
    show st-red S x
    proof (rule ccontr)
    assume neg-h: \negst-red S x
    have i: \negsubterm-reduction-applicable S x
    proof
            assume decomp-case: subterm-reduction-applicable S x
            then obtain x1 x2 where x = (Comb x1 x2) using is-compound.elims(2)
                unfolding subterm-reduction-applicable-def by blast
            from this and decomp-case have ((trm-rep x1 S) f x1 \vee (trm-rep x2 S)
# x2)
            using lhs.simps(1) rhs.simps(1)
            unfolding subterm-reduction-applicable-def by metis
            then show False
            proof
                assume (trm-rep x1 S) \not= x1
            from <x = (Comb x1 x2)> and trm-ord-subterm have (x1,x) \in trm-ord
by auto
            from this and hyp-ind and «(trm-rep x1 S) = x1>
                have st-red S x1 by blast
            from this and neg-h and \langlex = (Comb x1 x2)\rangle show False
                using red-arg-implies-red-trm [of S x1 x x2] by blast
            next
                assume (trm-rep x2 S)}\not=\mp@subsup{x}{2}{2
                from }\langlex=(\mathrm{ Comb x1 x2)> and trm-ord-subterm have (x2,x) }\in\mathrm{ trm-ord
by auto
            from this and hyp-ind and 〈(trm-rep x2 S) \not= x2`
                        have st-red S x2 by metis
                from this and neg-h and <x = (Comb x1 x2)> show False
                using red-arg-implies-red-trm [of S x2 x x1] by blast
            qed
            qed
            then have (trm-rep x S)=(get-min x (set-of-candidate-values S x)}
            using trm-rep-simp-def by metis
            then have root-term Sx}\mathrm{ unfolding root-term-def by blast
            have subterm x [] x by auto
            from this and <root-term S x\rangle and «(trm-rep x S) \not= x\ranglehave
                st-red S x unfolding st-red-def by blast
            from this and neg-h show False by auto
            qed
    qed
qed
lemma trm-rep-compatible-with-structure:
    shows value-is-compatible-with-structure ( }\lambdax.\mathrm{ trm-rep x S)
proof (rule ccontr)
    assume \negvalue-is-compatible-with-structure ( }\lambdax.\mathrm{ trm-rep x S)
```

from this obtain $t s$
where neg-h:trm-rep (Comb t s) $S \neq($ trm-rep (Comb (trm-rep t $S$ ) (trm-rep s S)) $S$ )
unfolding value-is-compatible-with-structure-def by blast
from this have $($ trm-rep $t S) \neq t \vee($ trm-rep $s S) \neq s$ by metis
from this have subterm-reduction-applicable $S$ (Comb ts) unfolding subterm-reduction-applicable-def
by (metis is-compound.simps(3) lhs.simps(1) rhs.simps(1))
from this have (trm-rep (Comb ts) $S$ ) $=($ subterm-reduction $S($ Comb $t s))$ using trm-rep-simp-def by metis
from this and neg-h show False unfolding subterm-reduction-def by (metis lhs.simps(1) rhs.simps(1))
qed
The following function checks that a position can be reduced, taking into account the order on positions associated with the considered clause and term. A term is reducible when all terms occurring at smaller positions are irreducible.

```
definition minimal-redex
    where minimal-redex p t C S t'
    =(\forallqs. ((q,p) ( (pos-ord C t')\longrightarrow(subterm t q s)\longrightarrow(trm-rep s S=s)))
```

The next function checks that a given clause contains two equations with the same left-hand side and whose right-hand sides are equivalent in a given interpretation. If no such equations exist then it is clear that the maximal literal is necessarily unique.

```
definition equivalent-eq-exists
    where equivalent-eq-exists t s C I \sigma L1 = (\existsL\inC-{L1 }. \existsuv.
    (orient-lit-inst L u v pos \sigma) ^((subst t \sigma) =(subst u \sigma))
    \wedge(I (subst s \sigma)(subst v \sigma)))
lemma maximal-literal-is-unique-lemma:
    assumes \negequivalent-eq-exists t s C (int-clset S) \sigma L1
    shows maximal-literal-is-unique (subst t \sigma) (subst s \sigma) C L1 S \sigma
proof (rule ccontr)
    let ?t = (subst t \sigma)
    let ?s=(subst s \sigma)
    let ?L}=(\mathrm{ subst-lit L }\sigma
    let ?C = (subst-cl C \sigma)
    assume }\neg(\mathrm{ maximal-literal-is-unique ?t ?s C L1 S %)
    from this obtain s"' where (eq-occurs-in-cl ?t s"\prime}(C-{L1 }) \sigma)
        and (trm-rep ?s S)}=(\mathrm{ trm-rep s" S) unfolding maximal-literal-is-unique-def
by blast
    from <(eq-occurs-in-cl ?t s" (C-{L1 }) \sigma)\rangle
        obtain }\mp@subsup{L}{}{\prime}\mp@subsup{t}{}{\prime}\mp@subsup{s}{}{\prime}\mathrm{ where }\mp@subsup{L}{}{\prime}\in(C-{L1}
            and orient-lit-inst L' L' t' sos \sigma and (subst t' \sigma) = ?t
            and }\mp@subsup{s}{}{\prime\prime}=\mathrm{ subst }\mp@subsup{s}{}{\prime}
```

unfolding eq－occurs－in－cl－def by auto
from $\left\langle s^{\prime \prime}=\right.$ subst $\left.s^{\prime} \sigma\right\rangle$ and $\left\langle(\right.$ trm-rep ?s $S)=\left(\right.$ trm-rep $\left.\left.s^{\prime \prime} S\right)\right\rangle$
have (trm-rep ?s $S)=\left(\right.$ trm-rep $\left(\right.$ subst $\left.\left.s^{\prime} \sigma\right) S\right)$ by blast
from $\left\langle L^{\prime} \in(C-\{L 1\})\right\rangle\left\langle o r i e n t-l i t-i n s t L^{\prime} t^{\prime} s^{\prime}\right.$ pos $\left.\sigma\right\rangle\left\langle\left(\right.\right.$ subst $\left.t^{\prime} \sigma\right)=$ ? $\left.t\right\rangle$
$\left\langle(\right.$ trm-rep ?s $S)=\left(\right.$ trm-rep $\left(\right.$ subst $\left.\left.\left.s^{\prime} \sigma\right) S\right)\right\rangle$
have equivalent-eq-exists $t s C$ (int-clset $S$ ) $\sigma$ L1
unfolding equivalent-eq-exists-def same-values-def int-clset-def
by metis
from this and assms(1) show False by blast
qed
lemma max-pos-lit-dominates-cl:
assumes maximal-literal (subst-lit L $\sigma$ ) (subst-cl C $\sigma$ )
assumes orient-lit-inst $L t s$ pos $\sigma$
assumes $L^{\prime} \in C-\{L\}$
assumes $\neg$ equivalent-eq-exists $t s C I \sigma L$
assumes vars-of-lit (subst-lit L $\sigma$ ) $=\{ \}$
assumes vars-of-lit (subst-lit $\left.L^{\prime} \sigma\right)=\{ \}$
assumes fo-interpretation I
shows $\left(\left(\right.\right.$ subst-lit $\left.L^{\prime} \sigma\right),($ subst-lit $\left.L \sigma)\right) \in$ lit-ord
proof -
let $? L^{\prime}=\left(\right.$ subst-lit $\left.L^{\prime} \sigma\right)$
let $? L=($ subst-lit $L \sigma)$
let $? t=($ subst $t \sigma)$
let $? s=($ subst $s \sigma)$
from $\operatorname{assms}(2)$ have $(? t, ? s) \notin$ trm-ord unfolding orient-lit-inst-def by auto
obtain $u^{\prime} v^{\prime}$ where $L^{\prime}=\left(\operatorname{Pos}\left(E q u^{\prime} v^{\prime}\right)\right) \vee L^{\prime}=\left(\operatorname{Neg}\left(E q u^{\prime} v^{\prime}\right)\right)$
using literal.exhaust equation.exhaust by metis
from this obtain polarity $u v$ where orient-lit-inst $L^{\prime} u v$ polarity $\sigma$
and $(($ subst $u \sigma),($ subst $v \sigma)) \notin$ trm-ord using
trm-ord-trans trm-ord-irrefl unfolding trans-def irrefl-def orient-lit-inst-def by
metis
let $? u=($ subst $u \sigma)$
let $? v=($ subst $v \sigma)$
from <orient-lit-inst $L^{\prime} u$ v polarity $\left.\sigma\right\rangle$ have orient-lit ? $L^{\prime}$ ?u ?v polarity
using lift-orient-lit by auto
from <orient-lit-inst $L t$ s pos $\sigma$ 〉 have orient-lit ?L ?t ?s pos
using lift-orient-lit by auto
from $\operatorname{assms}(6)$ and <orient-lit ? $L^{\prime}$ ?u ?v polarity>
have vars-of ? $u \subseteq\}$ using orient-lit-vars by metis
from $\operatorname{assms}(6)$ and oorient-lit ? $L^{\prime}$ ?u ?v polarity>
have vars-of ?v $\subseteq\}$ using orient-lit-vars by metis
from $\operatorname{assms}(5)$ and 〈orient-lit ?L ?t ?s pos〉
have vars-of ? $\subseteq \subseteq\}$ using orient-lit-vars by metis
from $\operatorname{assms}(5)$ and «orient-lit ?L ?t ?s pos〉
have vars-of ?s $\subseteq\}$ using orient-lit-vars by metis
from $\operatorname{assms}(1)$ and $\left\langle L^{\prime} \in C-\{L\}\right\rangle$ have $\left(? L, ? L^{\prime}\right) \notin$ lit－ord
unfolding maximal－literal－def by auto
from this and 〈orient－lit ？L ？t ？s pos〉＜orient－lit ？$L^{\prime}$ ？u ？v polarity〉 and $\operatorname{assms}(5) \operatorname{assms}(6)$
have（？t，？u）$\notin$ trm－ord using lit－ord－dominating－term by metis
from this and 〈vars－of ？t $\subseteq\}\rangle\langle v a r s-o f ? u \subseteq\{ \}\rangle$ have ？$u=? t \vee(? u, ? t) \in$ trm－ord
using trm－ord－ground－total unfolding ground－term－def by blast
from 〈（？u，？v）$\notin$ trm－ord〉 and «vars－of ？$u \subseteq\}\rangle\langle$ vars－of ？$v \subseteq\}\rangle$ have $? u=? v \vee(? v, ? u) \in$ trm－ord
using trm－ord－ground－total unfolding ground－term－def by blast
from 〈（？t，？s）$\notin$ trm－ord $\rangle$ and $\langle$ vars－of ？$\subseteq \subseteq\}\rangle\langle v a r s-o f ? s \subseteq\{ \}\rangle$
have $? t=$ ？$s \vee(? s, ? t) \in$ trm－ord
using trm－ord－ground－total unfolding ground－term－def by blast
from 〈vars－of ？v $\subseteq\}\rangle\langle$ vars－of ？s $\subseteq\}\rangle$ have ？v $=$ ？s $\vee(? v, ? s) \in$ trm－ord $\vee$ $(? s, ? v) \in$ trm－ord
using trm－ord－ground－total unfolding ground－term－def by blast
show ？thesis
proof（cases）
assume $(? u, ? t) \in$ trm－ord
from this and $\langle ? u=? v \vee(? v, ? u) \in$ trm－ord $\rangle$ have $(? v, ? t) \in$ trm－ord using trm－ord－trans unfolding trans－def by auto
from this and $\left\langle(? u, ? t) \in\right.$ trm－ord〉 and 〈orient－lit ？L ？t ？s pos〉〈orient－lit ？$L^{\prime}$ ？u ？v polarity＞
assms（5）assms（6）show ？thesis using lit－ord－dominating－term by metis
next
assume $(? u, ? t) \notin$ trm－ord
from this and $\langle ? u=? t \vee(? u, ? t) \in$ trm－ord $\rangle$ have ？$u=? t$ by auto
have polarity $=$ pos
proof（rule ccontr）
assume polarity $\neq$ pos
then have polarity $=$ neg using sign．exhaust by auto
from this and 〈？u $=$ ？t〉 and 〈orient－lit ？L ？t ？s pos〉
〈orient－lit ？L＇？u ？v polarity〉 assms（5）assms（6）
have $\left(? L, ? L^{\prime}\right) \in$ lit－ord using lit－ord－neg－lit－lhs by auto
from this and $\left\langle\left(? L, ? L^{\prime}\right) \notin\right.$ lit－ord $\rangle$ show False by auto
qed
have ？$v \neq$ ？s
proof
assume ？v $=$ ？s
from this assms（7）have $I$ ？s ？v unfolding fo－interpretation－def congru－ ence－def
equivalence－relation－def reflexive－def by auto
from this and $\left\langle\right.$ orient－lit－inst $L^{\prime} u$ v polarity $\left.\sigma\right\rangle\langle$ polarity $=$ pos $\langle\langle ? u=?\rangle$,
and $\operatorname{assms}(3)$ have equivalent－eq－exists $t s C I \sigma L$
unfolding equivalent－eq－exists－def by metis
from this and assms（4）show False by auto

## qed

have $(? s, ? v) \notin$ trm－ord
proof
assume（？s，？v）$\in$ trm－ord
from this and 〈？$u=$ ？t〉 and 〈orient－lit ？L ？t ？s pos〉〈orient－lit ？L＇？u ？v polarity＞
and $\langle$ polarity $=$ pos〉 $\operatorname{assms}(5) \operatorname{assms}(6)$
have $\left(? L, ? L^{\prime}\right) \in$ lit-ord using lit-ord-rhs by auto
from this and $«\left(? L, ? L^{\prime}\right) \notin$ lit-ord $\rangle$ show False by auto
qed
from this and $\langle ? v \neq ? s\rangle$ and $\langle ? v=? s \vee(? v, ? s) \in$ trm-ord $\vee(? s, ? v) \in$
trm-ord>
have $(? v, ? s) \in$ trm-ord by metis
from this and $\langle ? u=? t\rangle$ and 〈orient-lit ? L ?t ?s pos〉〈orient-lit ?L' ?u ?v
polarity>
and $\langle$ polarity $=$ pos〉 $\operatorname{assms}(5) \operatorname{assms}(6)$
show $\left(? L^{\prime}, ? L\right) \in$ lit-ord using lit-ord-rhs by auto
qed
qed
lemma if－all－smaller－are－false－then－cl－not－valid：
assumes（smaller－lits－are－false（subst t $\sigma$ ）（subst－cl C $\sigma$ ）S）
assumes（fo－interpretation（same－values（ $\lambda t$ ．（trm－rep $t S)$ ）））
assumes orient－lit－inst L1 t s pos $\sigma$
assumes maximal－literal（subst－lit L1 $\sigma$ ）（subst－cl C $\sigma$ ）
assumes ground－clause（subst－cl C $\sigma$ ）
assumes（subst－lit L1 $\sigma$ ）$\in($ subst－cl C $\sigma$ ）
assumes $\neg$ equivalent－eq－exists $t$ s $C$（same－values $(\lambda t$ ．$($ trm－rep $t S))) \sigma L 1$
assumes trm－rep（subst $t \sigma$ ）$S=$ trm－rep（subst s $\sigma$ ）$S$
shows（ $\neg$ validate－ground－clause（same－values $(\lambda t$ ．（trm－rep $t S))$ ）（subst－cl（ $C$
－\｛L1 \} ) $\sigma$ ））
proof
let $? I=($ same－values $(\lambda t .($ trm－rep $t S)))$
assume validate－ground－clause ？I（subst－cl $(C-\{L 1\}) \sigma$ ）
then obtain $L$ where $L \in($ subst－cl $(C-\{L 1\}) \sigma)$ and validate－ground－lit ？I $L$
using validate－ground－clause．simps $[$ of ？I（subst－cl $(C-\{L 1\}) \sigma)]$ by blast
from $\langle L \in($ subst－cl $(C-\{L 1\}) \sigma)\rangle$ obtain $L^{\prime}$ where $L^{\prime} \in C-\{L 1\}$ and
$L=\left(\right.$ subst－lit $\left.L^{\prime} \sigma\right)$ by auto
from $\left\langle L^{\prime} \in C-\{L 1\}\right\rangle$ and $\left\langle L=\left(\right.\right.$ subst－lit $\left.\left.L^{\prime} \sigma\right)\right\rangle$
have $L \in($ subst－cl $C \sigma)$ by auto
from $\langle L \in($ subst－cl $C \sigma)\rangle$ and assms（5）have vars－of－lit $L=\{ \}$ by auto
from $\operatorname{assms}(6)$ and $\operatorname{assms}(5)$ have vars－of－lit（subst－lit L1 $\sigma$ ）$=\{ \}$ by auto
obtain $u v$ polarity where $o$ ：orient－lit－inst $L^{\prime} u v$ polarity $\sigma$
and $(($ subst $u \sigma),($ subst $v \sigma)) \notin$ trm－ord
unfolding orient－lit－inst－def using literal．exhaust equation．exhaust
trm－ord－trans trm－ord－irrefl unfolding trans－def irrefl－def by metis from $o$ and $\left\langle L=\left(\right.\right.$ subst－lit $\left.\left.L^{\prime} \sigma\right)\right\rangle$
have $o^{\prime}$ ：orient－lit $L$（subst $u \sigma$ ）（subst $v \sigma$ ）polarity
using lift－orient－lit by auto
from $o^{\prime}$ and $\langle$ vars－of－lit $L=\{ \}\rangle$ have vars－of（subst $u \sigma$ ）$=\{ \}$ using orient－lit－vars by blast
from $o^{\prime}$ and 〈vars－of－lit $\left.L=\{ \}\right\rangle$ have vars－of（subst $v \sigma$ ）$=\{ \}$ using orient－lit－vars by blast
from $\operatorname{assms}(3)$
have o1：orient－lit（subst－lit L1 $\sigma$ ）（subst $t \sigma$ ）（subst s $\sigma$ ）pos
using lift－orient－lit［of L1 t s pos $\sigma$ ］by auto
from o1 and 〈vars－of－lit（subst－lit L1 $\sigma$ ）$=\{ \}$ 〉have vars－of（subst $t \sigma)=\{ \}$ using orient－lit－vars by blast
have polarity $=$ pos $\vee$ polarity $=$ neg using sign．exhaust by auto
then show False

## proof

assume polarity $=$ pos
from this and $o$ and assms（2）and $\langle$ validate－ground－lit ？$I L$ and $\langle L=$ （subst－lit $L^{\prime} \sigma$ ）＞
have（trm－rep（subst $u \sigma) S)=($ trm－rep $($ subst $v \sigma) S)$
using orient－lit－semantics－pos［of ？I ］unfolding same－values－def by metis
from $\operatorname{assms}(4)$ and $\langle L \in($ subst－cl $C \sigma)\rangle$
have $(($ subst－lit L1 $\sigma), L) \notin$ lit－ord unfolding maximal－literal－def
by blast
from this and $o^{\prime}$ and $o 1$ and $\langle$ polarity $=p o s\rangle$ and $\langle v a r s-o f-l i t ~ L=\{ \}\rangle$ and $\langle L$ $=\left(\right.$ subst－lit $\left.L^{\prime} \sigma\right)$＞
and 〈vars－of－lit（subst－lit L1 $\sigma$ ）$=\{ \}$ 〉
have（subst $t \sigma$ ，subst $u \sigma) \notin$ trm－ord
and（subst $t \sigma$ ，subst $v \sigma$ ）$\notin$ trm－ord
using lit－ord－dominating－term［of subst $t \sigma$ subst $u \sigma$
subst $v \sigma$ subst－lit $L 1 \quad \sigma$ subst $s \sigma$ pos $]$ by auto
show ？thesis
proof（cases）
assume（subst $t \sigma)=($ subst $u \sigma)$
from this and $\operatorname{assms}(8)$ and $<($ trm－rep $($ subst $u \sigma) S)=($ trm－rep（subst $v$
$\sigma) S$ ）
have（trm－rep（subst s $\sigma$ ）$S$ ）$=($ trm－rep $($ subst $v \sigma) S)$ by metis
from this $o$ and $\left\langle L^{\prime} \in C-\{L 1\}\right\rangle\langle$ polarity $=$ pos $\langle\langle$（subst $t \sigma)=($ subst $u \sigma)\rangle \operatorname{assms}(7)$
show False unfolding equivalent－eq－exists－def same－values－def by blast
next
assume（subst $t \sigma) \neq($ subst $u \sigma)$
from this and $\langle($ subst $t \sigma$ ，subst $u \sigma) \notin$ trm－ord $\rangle$
and 〈vars－of（subst $t \sigma)=\{ \}$ 〉 and «vars－of（subst $u \sigma)=\{ \}$ 〉
have（subst $u \sigma$ ，subst $t \sigma$ ）$\in$ trm－ord
using trm－ord－ground－total unfolding ground－term－def by auto

```
            from this and «(subst u \sigma, subst v \sigma) #trm-ord }
            and <vars-of (subst v \sigma)={}> and <vars-of (subst t \sigma)={}>
            have (subst v \sigma, subst t \sigma)\in trm-ord
            using trm-ord-ground-total trm-ord-trans
            unfolding ground-term-def trans-def by metis
            from <polarity = pos\rangle and o' and assms(1) and <L \in(subst-cl C \sigma)\rangle and
<L=(subst-lit L' \sigma)\rangle
            and «((subst u \sigma), subst t \sigma)\in trm-ord>
            and «((subst v \sigma), subst t \sigma) \in trm-ord\rangle
            have trm-rep (subst u \sigma)S\not= trm-rep (subst v \sigma)S
            unfolding smaller-lits-are-false-def by metis
            from this and <trm-rep (subst u\sigma)S= trm-rep (subst v \sigma)S〉
            show False by blast
        qed
    next assume polarity = neg
        from this and o and assms(2) and <validate-ground-lit ?I L\rangle and <L =
(subst-lit L' \sigma)>
            have (trm-rep (subst u \sigma)S)}=(\mathrm{ trm-rep (subst v }\sigma)S
            using orient-lit-semantics-neg [of ?I ] unfolding same-values-def by metis
    from assms(4) and <L (subst-cl C \sigma)\rangle
            have ((subst-lit L1 \sigma),L) # lit-ord unfolding maximal-literal-def
            by blast
                            from this and o' and o1 and <vars-of-lit L = {}> and <L = (subst-lit L' \sigma)\rangle
            and <vars-of-lit (subst-lit L1 \sigma) = {}>
            have (subst t \sigma, subst u\sigma)\not\in trm-ord
            and (subst t \sigma, subst v \sigma) & trm-ord
            using lit-ord-dominating-term [of subst t \sigma subst u \sigma
                    subst v \sigma subst-lit L1 \sigma subst s \sigma pos]
            by auto
```

from 〈（（subst－lit L1 $\sigma), L) \notin$ lit－ord〉 and $o^{\prime}$ and $o 1$ and $\langle p o l a r i t y=n e g\rangle$ and〈vars－of－lit $L=\{ \}\rangle$
and $\left\langle L=\left(\right.\right.$ subst-lit $\left.\left.L^{\prime} \sigma\right)\right\rangle$ and $\langle$ vars-of-lit (subst-lit L1 $\left.\sigma)=\{ \}\right\rangle$
have subst $t \sigma \neq$ subst $u \sigma$
using lit-ord-neg-lit-lhs by auto
from this and $\langle($ subst $t \sigma$, subst $u \sigma) \notin$ trm-ord $\rangle$ and $\langle$ vars-of (subst $t \sigma)=$
\{\}>
and «vars-of (subst $u \sigma$ ) $=\{ \}$ 〉
have (subst $u \sigma$, subst $t \sigma$ ) $\in$ trm-ord
using trm-ord-ground-total unfolding ground-term-def by auto
from this and $\langle($ subst $u \sigma$, subst $v \sigma) \notin$ trm-ord $\rangle$ and $\langle$ vars-of (subst $v \sigma)=$ \｛\}>
and «vars－of（subst $t \sigma)=\{ \}$ have（subst $v \sigma$ ，subst $t \sigma$ ）$\in$ trm－ord
using trm－ord－ground－total trm－ord－trans unfolding ground－term－def trans－def by metis
from $\langle$ polarity $=n e g\rangle$ and $o^{\prime}$ and $\operatorname{assms}(1)$ and $\langle L \in($ subst-cl $C \sigma)\rangle$ and $\langle L$ $=\left(\right.$ subst-lit $\left.\left.L^{\prime} \sigma\right)\right\rangle$
and $\langle(($ subst $u \sigma)$, subst $t \sigma) \in$ trm-ord $\rangle$ and $\langle(($ subst $v \sigma)$, subst $t \sigma) \in$ trm-ord>
have trm-rep (subst u $\sigma$ ) $S=$ trm-rep (subst $v \sigma$ ) $S$
unfolding smaller-lits-are-false-def by metis
from this and 〈trm-rep (subst $u \sigma$ ) $S \neq$ trm-rep (subst $v \sigma$ ) $S$ 〉 show False by blast
qed
qed
We introduce the notion of a reduction, which is a ground superposition inference satisfying some additional conditions:
(i) the "from" clause is smaller than the "into" clause;
(ii) its "body" (i.e., the part of the clause without the equation involved in the rule) is false in a given interpretation and strictly smaller than the involved equation.

```
definition reduction
where (reduction L1 C \(\sigma^{\prime} t\) s polarity L2 u u'p vDIS \(\sigma\) )=
    \(\left((D \in S) \wedge\left(\right.\right.\) eligible-literal L1 \(\left.C \sigma^{\prime}\right) \wedge\left(\right.\) eligible-literal L2 \(\left.D \sigma^{\prime}\right)\)
    \(\wedge\) ground-clause (subst-cl (cl-ecl D) \(\sigma^{\prime}\) )
    \(\wedge\) (minimal-redex \(p\) (subst \(t \sigma) C S t)\)
    \(\wedge\left(\right.\) coincide-on \(\sigma \sigma^{\prime}(\) vars-of-cl \((\) cl-ecl \(\left.C))\right)\)
    \(\wedge\) (allowed-redex \(u^{\prime} C \sigma\) )
    \(\wedge\left(\neg\right.\) is-a-variable \(\left.u^{\prime}\right)\)
    \(\wedge(L 1 \in(\) cl-ecl \(C)) \wedge(L 2 \in(\) cl-ecl \(D))\)
    \(\wedge\) (orient-lit-inst L1 ts polarity \(\left.\sigma^{\prime}\right)\)
    \(\wedge\) (orient-lit-inst L2 u v pos \(\sigma^{\prime}\) )
    \(\wedge\left(\right.\) subst \(\left.u \sigma^{\prime}\right) \neq\left(\right.\) subst \(\left.v \sigma^{\prime}\right)\)
    \(\wedge\left(\right.\) subst \(\left.u^{\prime} \sigma^{\prime}\right)=\left(\right.\) subst \(\left.u \sigma^{\prime}\right)\)
    \(\wedge\left(\neg\right.\) validate-ground-clause \(I\left(\right.\) subst-cl \(((\) cl-ecl \(\left.\left.D)-\{L 2\}) \sigma^{\prime}\right)\right)\)
    \(\wedge\left(\left(\right.\right.\) subst-lit L2 \(\left.\sigma^{\prime}\right),\left(\right.\) subst-lit L1 \(\left.\left.\sigma^{\prime}\right)\right) \in\) lit-ord
    \(\wedge\left(\forall x \in(\right.\) cl-ecl \(D)-\{\) L2 \(\} .\left(\left(\right.\right.\) subst-lit \(\left.x \sigma^{\prime}\right),\left(\right.\) subst-lit L2 \(\left.\left.\sigma^{\prime}\right)\right)\)
                        \(\in\) lit-ord)
    \(\wedge\left(\right.\) all-trms-irreducible (subst-set (trms-ecl D) \(\left.\sigma^{\prime}\right)\)
            ( \(\lambda t\). (trm-rep \(t S)\) )
    \(\wedge\left(I\left(\right.\right.\) subst \(\left.u \sigma^{\prime}\right)\left(\right.\) subst \(\left.\left.v \sigma^{\prime}\right)\right)\)
    \(\wedge\left(\right.\) subterm t \(\left.\left.p u^{\prime}\right)\right)\)
```

The next lemma states that the rules used to evaluate terms can be renamed so that they share no variable with the clause in which the term occurs.

```
lemma candidate-values-renaming:
    assumes (candidate-values z CC \(\left.C^{\prime} C s L L^{\prime} \sigma t^{\prime} s^{\prime} t S\right)\)
    assumes finite \(C^{\prime}\)
    assumes finite (cl-ecl ( \(D::\) 'a eclause))
    assumes closed-under-renaming \(S\)
    assumes Ball \(S\) well-constrained
    shows \(\exists C C\)-bis \(C^{\prime}\)-bis \(L^{\prime}\)-bis \(\sigma\)-bis \(t^{\prime}\)-bis \(s^{\prime}\)-bis \(t\)-bis.
```

（candidate－values z CC－bis $C^{\prime}$－bis C s L $L^{\prime}$－bis $\sigma$－bis $t^{\prime}$－bis $s^{\prime}$－bis $t S$ ）
$\wedge($ vars－of－cl $($ cl－ecl $D)) \cap$ vars－of－cl $($ cl－ecl CC－bis $)=\{ \}$
proof－
from $\operatorname{assms}(2)$ have finite（vars－of－cl $C^{\prime}$ ）using set－of－variables－is－finite－cl by auto
from $\operatorname{assms}(3)$ have finite（vars－of－cl（cl－ecl D））using set－of－variables－is－finite－cl
by auto
from infinite－vars have $\neg$（finite vars）by auto
from 〈finite（vars－of－cl $\left.\left.C^{\prime}\right)\right\rangle\langle$ finite（vars－of－cl（cl－ecl D））〉
and $\langle\neg$（finite vars）〉
obtain $\eta$ where renaming $\eta$（vars－of－cl $\left.C^{\prime}\right)$
and $\left(\left(\right.\right.$ subst－codomain $\eta\left(\right.$ vars－of－cl $\left.\left.C^{\prime}\right)\right) \cap($ vars－of－cl $($ cl－ecl $\left.D))\right)=\{ \}$
using renaming－exists［of vars ］by meson
from this 〈finite（vars－of－cl $C^{\prime}$ ）〉 obtain $\eta^{\prime}$
where $i:\left(\forall x \in\left(\right.\right.$ vars－of－cl $\left.C^{\prime}\right) .\left(\right.$ subst $($ subst $\left.(\operatorname{Var} x) \eta) \eta^{\prime}\right)$
$=(\operatorname{Var} x))$
using renamings－admit－inverse by blast
obtain CC－bis where $C C$－bis $=($ subst－ecl $C C \eta)$ by auto
obtain $C^{\prime}$－bis where $C^{\prime}$－bis $=\left(\right.$ subst－cl $\left.C^{\prime} \eta\right)$ by auto
from $\operatorname{assms}(1)$ have $C^{\prime}=($ cl－ecl $C C)$ unfolding candidate－values－def by metis
from this obtain $T$ where $C C=\left(E c l C^{\prime} T\right)$
by（metis cl－ecl．simps trms－ecl．elims）
from this and $\langle C C$－bis $=($ subst－ecl $C C \eta)\rangle$
and $\left\langle C^{\prime}\right.$－bis $=\left(\right.$ subst－cl $\left.\left.C^{\prime} \eta\right)\right\rangle$
have $C^{\prime}$－bis $=($ cl－ecl CC－bis）by auto
from assms（1）have $C C \in S$ unfolding candidate－values－def by metis
from $\operatorname{assms}(1)$ have $(s, t) \in$ trm－ord unfolding candidate－values－def by metis
from $\operatorname{assms}(1)$ have $((s, t) \in$ trm－ord $\longrightarrow(z=$ trm－rep $s S))$
unfolding candidate－values－def by metis
from assms（1）have（maximal－literal $L C$ ）unfolding candidate－values－def by metis
from $\operatorname{assms}(1)$ have（ground－clause $C$ ）unfolding candidate－values－def by metis
from $\operatorname{assms}(1)$ have $L^{\prime} \in C^{\prime}$ unfolding candidate－values－def by metis
from $\operatorname{assms}(1)$ have $L=\left(\right.$ subst－lit $\left.L^{\prime} \sigma\right)$
unfolding candidate－values－def by metis
from $\operatorname{assms}(1)$ have（smaller－lits－are－false t C S）
unfolding candidate－values－def by metis
from $\operatorname{assms}(1)$ have $C=\left(\right.$ subst－cl $\left.C^{\prime} \sigma\right)$
unfolding candidate－values－def by metis
from $\operatorname{assms}(1)$ have（orient－lit－inst $L^{\prime} t^{\prime} s^{\prime}$ pos $\sigma$ ） unfolding candidate－values－def by metis
from assms（1）have（trms－irreducible CC $\sigma S t$ ） unfolding candidate－values－def by metis
from assms（1）have $t=$ subst $t^{\prime} \sigma$ unfolding candidate－values－def by metis
from assms（1）have $s=$ subst $s^{\prime} \sigma$ unfolding candidate－values－def by metis
from $\langle C C \in S\rangle$ and $\operatorname{assms}(4)$ and $\left\langle r e n a m i n g ~ \eta\left(v a r s-o f-c l ~ C^{\prime}\right)\right\rangle$ and $\left\langle C^{\prime}=\right.$

```
(cl-ecl CC)>
    <CC-bis = (subst-ecl CC \eta)\rangle have CC-bis }\in
    unfolding closed-under-renaming-def renaming-cl-def by auto
    from assms(1) have (sel C'= {}) unfolding candidate-values-def by metis
    from this and <renaming \eta (vars-of-cl C')\rangle\langle\mp@subsup{C}{}{\prime}=(cl-ecl CC)\rangle
        <C'-bis = (subst-cl C' }\mp@subsup{C}{}{\prime})\rangle\mathrm{ have sel C'-bis = {}
    using sel-renaming by auto
obtain }\mp@subsup{L}{}{\prime}\mathrm{ -bis where }\mp@subsup{L}{}{\prime}\mathrm{ -bis = (subst-lit L' }\mp@subsup{L}{}{\prime}\eta)\mathrm{ by auto
from this and }\langle\mp@subsup{L}{}{\prime}\in\mp@subsup{C}{}{\prime}\rangle\langle\mp@subsup{C}{}{\prime}\mathrm{ -bis = (subst-cl C' }\eta\mathrm{ ) > have }\mp@subsup{L}{}{\prime}\mathrm{ -bis }\in\mp@subsup{C}{}{\prime}\mathrm{ -bis by auto
let ?\vartheta = (comp (comp \eta \mp@subsup{\eta}{}{\prime})\sigma)
let ? }\mp@subsup{\vartheta}{}{\prime}=(\operatorname{comp \eta}\mp@subsup{\eta}{}{\prime}\sigma
have coincide-on \sigma ?\vartheta (vars-of-cl C')
proof (rule ccontr)
    assume \negcoincide-on \sigma ?\vartheta (vars-of-cl C')
    from this obtain x where (subst (Var x) \sigma) \not= (subst (Var x) ?\vartheta)
        and x\invars-of-cl C' unfolding coincide-on-def by auto
    from <x \in vars-of-cl C'>}
        have (subst (subst (Var x) \eta) \eta')=(Var x )
        by blast
    from this and «(subst (Var x) \sigma) \not=( (subst (Var x) ?\vartheta)>
        show False by simp
qed
from }\langle\mp@subsup{L}{}{\prime}\in\mp@subsup{C}{}{\prime}\rangle\mathrm{ have vars-of-lit L' L' vars-of-cl C'' by auto
    from this and <coincide-on \sigma ?\vartheta (vars-of-cl C')\rangle
    have coincide-on \sigma ?\vartheta (vars-of-lit L') unfolding coincide-on-def by auto
from this and <L = (subst-lit L' \sigma)\rangle
    have L = (subst-lit L' ?\vartheta) using coincide-on-lit by auto
have subst-lit L' ?\vartheta
    = subst-lit (subst-lit L' }\mp@subsup{L}{}{\prime}\mathrm{ ) ?, '
    by (simp add: coincide-on-def coincide-on-lit composition-of-substs-lit)
from this and <L=(subst-lit L' ?\vartheta)> and
    <L'-bis = (subst-lit L' }\mp@subsup{L}{}{\prime})
    have L=(subst-lit L'-bis ?\vartheta')
    by auto
from <coincide-on \sigma ?\vartheta (vars-of-cl C')\rangle and <C=(subst-cl C' }\mp@subsup{C}{}{\prime}\sigma)
    have C= subst-cl C' ?\vartheta
    using coincide-on-cl by blast
have subst-cl C' ?\vartheta
    = subst-cl (subst-cl C' \eta) ?\vartheta'
    by (metis composition-of-substs-cl)
from this and «C = subst-cl C' '?\vartheta\rangle and \langleC'-bis = (subst-cl C' }\mp@subsup{C}{}{\prime})
    have C= subst-cl C'-bis ?\vartheta' by auto
```



```
have t #(subst-set (trms-ecl CC-bis) ?.\vartheta')
proof
    assume t\in(subst-set (trms-ecl CC-bis) ?\vartheta`)
    from this obtain u where }u\in\mathrm{ (trms-ecl CC-bis)
        and t=(subst u? `') by auto
```

from $\langle u \in($ trms－ecl $C C$－bis $)\rangle$ and $\langle C C$－bis $=($ subst－ecl $C C \eta)\rangle$ obtain $v$ where $v \in$ trms－ecl $C C$ and $u=($ subst $v \eta)$
using $\left\langle C C=E c l C^{\prime} T\right\rangle$ by auto
from $\langle u=($ subst $v \eta)\rangle\left\langle t=\left(\right.\right.$ subst $u$ ？$\left.\left.\vartheta^{\prime}\right)\right\rangle$ have subst $v ? \vartheta=t$ by simp
from $\langle v \in$ trms－ecl $C C\rangle\langle C C \in S\rangle \operatorname{assms}(5)$
have dom－trm $v$（cl－ecl CC）unfolding Ball－def well－constrained－def by metis
from this have vars－of $v \subseteq$ vars－of－cl（cl－ecl CC）using dom－trm－vars by auto
from this and $\left\langle C^{\prime}=(\right.$ cl－ecl $\left.C C)\right\rangle\left\langle c o i n c i d e-o n ~ \sigma ? \vartheta\left(v a r s-o f-c l C^{\prime}\right)\right\rangle$
have coincide－on $\sigma$ ？$\vartheta$（vars－of $v$ ）unfolding coincide－on－def by auto
from this and 〈subst $v ? \vartheta=t\rangle$ have（subst $v \sigma)=t$
using coincide－on－term by metis
from this and $\langle v \in$ trms－ecl $C C\rangle$ have
$(t \in($ subst－set $($ trms－ecl $C C) \sigma))$ by auto
from this and assms（1）show False unfolding candidate－values－def by metis
qed
have（trms－irreducible CC－bis ？$\vartheta^{\prime} S t$ ）
proof（rule ccontr）
assume $\neg$（trms－irreducible CC－bis ？$\vartheta^{\prime} S t$ ）
then obtain $x x^{\prime}$ where $x^{\prime} \in$ trms－ecl CC－bis occurs－in $x$（subst $\left.x^{\prime} ? \vartheta^{\prime}\right)$ $(x, t) \in$ trm－ord（trm－rep $x S) \neq x$ using trms－irreducible－def by blast
from $\left\langle x^{\prime} \in(\right.$ trms－ecl $C C$－bis $\left.)\right\rangle$ and $\langle C C$－bis $=($ subst－ecl $C C \eta)\rangle$ obtain $v$ where $v \in$ trms－ecl $C C$ and $x^{\prime}=($ subst $v \eta)$
using $\left\langle C C=E c l C^{\prime} T\right\rangle$ by auto
from $\left\langle\right.$ occurs－in $x\left(\right.$ subst $\left.\left.x^{\prime} ? \vartheta^{\prime}\right)\right\rangle\left\langle x^{\prime}=\right.$ subst $\left.v \eta\right\rangle$ have occurs－in $x$（subst $v$ ？$\vartheta$ ） by $\operatorname{simp}$
from $\langle v \in$ trms－ecl $C C\rangle\langle C C \in S\rangle$ assms（5）
have dom－trm $v$（cl－ecl CC）unfolding Ball－def well－constrained－def by auto
from this have vars－of $v \subseteq$ vars－of－cl（cl－ecl CC）using dom－trm－vars by auto
from this and $\left\langle C^{\prime}=(\right.$ cl－ecl $\left.C C)\right\rangle\left\langle c o i n c i d e-o n ~ \sigma ? \vartheta\left(v a r s-o f-c l C^{\prime}\right)\right\rangle$
have coincide－on $\sigma$ ？$\vartheta$（vars－of $v$ ）unfolding coincide－on－def by auto
from this have（subst $v \sigma)=($ subst $v$ ？$\vartheta)$
using coincide－on－term by metis
from this and＜occurs－in $x$（subst $v$ ？$\vartheta$ ）＞
have occurs－in $x$（subst $v \sigma$ ）by auto
from this and $\langle v \in$ trms－ecl $C C\rangle$ and $\langle(x, t) \in$ trm－ord $\rangle$ $\langle($ trm－rep $x S) \neq x\rangle$ and $\langle($ trms－irreducible $C C \sigma S t)\rangle$ show False unfolding trms－irreducible－def by metis
qed
obtain $t^{\prime}$－bis where $t^{\prime}$－bis $=\left(\right.$ subst $\left.t^{\prime} \eta\right)$ by auto
obtain $s^{\prime}$－bis where $s^{\prime}$－bis $=\left(\right.$ subst $s^{\prime} \eta$ ）by auto
from 〈（orient－lit－inst $L^{\prime} t^{\prime} s^{\prime}$ pos $\left.\left.\sigma\right)\right\rangle$ have vars－of $t^{\prime} \subseteq$ vars－of－lit $L^{\prime}$
using orient－lit－inst－vars by auto
from this 〈coincide－on $\sigma$ ？$\vartheta$（vars－of－lit $L^{\prime}$ ）〉
have coincide－on $\sigma$ ？$\vartheta$（vars－of $t^{\prime}$ ）unfolding coincide－on－def by blast
from this have subst $t^{\prime} ? \vartheta=$ subst $t^{\prime} \sigma$
using coincide－on－term by metis
from this $\left\langle t^{\prime}\right.$－bis $=\left(\right.$ subst $\left.\left.t^{\prime} \eta\right)\right\rangle$ have subst $t^{\prime}$－bis $? \vartheta^{\prime}=$ subst $t^{\prime} \sigma$ by simp
from 〈（orient－lit－inst $L^{\prime} t^{\prime} s^{\prime}$ pos $\left.\left.\sigma\right)\right\rangle$ have vars－of $s^{\prime} \subseteq$ vars－of－lit $L^{\prime}$
using orient－lit－inst－vars by auto
from this 〈coincide－on $\sigma$ ？$\vartheta$（vars－of－lit $L^{\prime}$ ）〉
have coincide－on $\sigma$ ？$\vartheta$（vars－of $s^{\prime}$ ）unfolding coincide－on－def by blast
from this have subst $s^{\prime} ? \vartheta=$ subst $s^{\prime} \sigma$
using coincide－on－term by metis
from this $\left\langle s^{\prime}\right.$－bis $=\left(\right.$ subst $\left.\left.s^{\prime} \eta\right)\right\rangle$ have subst $s^{\prime}$－bis $? \vartheta^{\prime}=$ subst $s^{\prime} \sigma$ by simp
have（orient－lit－inst $L^{\prime}$－bis $t^{\prime}$－bis $s^{\prime}$－bis pos ？$\vartheta^{\prime}$ ）
proof－
from $\left\langle\left(\right.\right.$ orient－lit－inst $L^{\prime} t^{\prime} s^{\prime}$ pos $\left.\left.\sigma\right)\right\rangle$
have $\left(\left(\right.\right.$ subst $\left.t^{\prime} \sigma\right),\left(\right.$ subst $\left.\left.s^{\prime} \sigma\right)\right) \notin$ trm－ord
using orient－lit－inst－def by simp
from 〈（orient－lit－inst $L^{\prime} t^{\prime} s^{\prime}$ pos $\left.\left.\sigma\right)\right\rangle$
have $L^{\prime}=\left(\operatorname{Pos}\left(E q t^{\prime} s^{\prime}\right)\right) \vee L^{\prime}=\left(\operatorname{Pos}\left(E q s^{\prime} t^{\prime}\right)\right)$
by（simp add：orient－lit－inst－def）
from this
$\left\langle L^{\prime}\right.$－bis $=\left(\right.$ subst－lit $\left.\left.L^{\prime} \eta\right)\right\rangle$
$\left\langle t^{\prime}\right.$－bis $=\left(\right.$ subst $\left.\left.t^{\prime} \eta\right)\right\rangle$
$\left\langle s^{\prime}\right.$－bis $=\left(\right.$ subst $\left.\left.s^{\prime} \eta\right)\right\rangle$
have $L^{\prime}$－bis $=\left(\operatorname{Pos}\left(E q t^{\prime}\right.\right.$－bis $s^{\prime}$－bis $\left.)\right) \vee L^{\prime}$－bis $=\left(\operatorname{Pos}\left(E q s^{\prime}\right.\right.$－bis $t^{\prime}$－bis $\left.)\right)$
by auto
from $\left\langle\right.$ subst $s^{\prime}$－bis ？$\vartheta^{\prime}=$ subst $\left.s^{\prime} \sigma\right\rangle$
and $\left\langle\right.$ subst $t^{\prime}$－bis ？$\vartheta^{\prime}=$ subst $\left.t^{\prime} \sigma\right\rangle$
and 〈（（subst $\left.t^{\prime} \sigma\right),\left(\right.$ subst $\left.\left.s^{\prime} \sigma\right)\right) \notin$ trm－ord〉
have $\left(\left(\right.\right.$ subst $t^{\prime}$－bis ？$\left.\vartheta^{\prime}\right),\left(\right.$ subst $s^{\prime}$－bis ？$\left.\left.\vartheta^{\prime}\right)\right) \notin$ trm－ord
by auto
from this and $\left\langle L^{\prime}\right.$－bis $=\left(\operatorname{Pos}\left(E q t^{\prime}\right.\right.$－bis $s^{\prime}$－bis $\left.)\right) \vee L^{\prime}$－bis $=\left(\operatorname{Pos}\left(E q s^{\prime}\right.\right.$－bis
$t^{\prime}$－bis））
show ？thesis unfolding orient－lit－inst－def by auto
qed
have（maximal－literal－is－unique ts $C^{\prime}$－bis $L^{\prime}$－bis $S$ ？$\vartheta^{\prime}$ ）
proof（rule ccontr）
assume $\neg$（maximal－literal－is－unique $t s C^{\prime}$－bis $L^{\prime}$－bis $S$ ？$\left.\vartheta^{\prime}\right)$
from this obtain $s^{\prime \prime}$ where（eq－occurs－in－cl t s＂$\left(C^{\prime}\right.$－bis－\｛ $L^{\prime}$－bis $\}$ ）？$\left.\vartheta^{\prime}\right)$
$\left(s^{\prime \prime}, t\right) \in$ trm－ord
$(s, t) \in t r m$－ord
$($ trm－rep $s S)=\left(\right.$ trm－rep $\left.s^{\prime \prime} S\right)$
unfolding maximal－literal－is－unique－def
by metis
from $\left\langle\left(e q-o c c u r s-i n-c l t s^{\prime \prime}\left(C^{\prime}\right.\right.\right.$－bis－$\left\{L^{\prime}\right.$－bis $\left.\left.\left.\}\right) ? \vartheta^{\prime}\right)\right\rangle$ obtain $M u v$ where
$M \in C^{\prime}$－bis－\｛ L＇－bis \} orient-lit-inst Muv pos ? $\vartheta^{\prime}$
$t=\left(\right.$ subst $u$ ？$\left.\vartheta^{\prime}\right) s^{\prime \prime}=\left(\right.$ subst $\left.v ? \vartheta^{\prime}\right)$
unfolding eq－occurs－in－cl－def by blast
from $\left\langle M \in C^{\prime}\right.$－bis $-\left\{L^{\prime}\right.$－bis $\left.\}\right\rangle$
and $\left\langle C^{\prime}\right.$－bis $=\left(\right.$ subst－cl $\left.\left.C^{\prime} \eta\right)\right\rangle$ and $\left\langle L^{\prime}\right.$－bis $=\left(\right.$ subst－lit $\left.\left.L^{\prime} \eta\right)\right\rangle$
obtain $M^{\prime}$ where $M^{\prime} \in C^{\prime}-\left\{L^{\prime}\right\}$ and subst－lit $M^{\prime} \eta=M$ by auto
from＜orient－lit－inst $M u v$ pos $\left.{ }^{2} \vartheta^{\prime}\right\rangle$ obtain $e$ where $M=($ Pos e）
unfolding orient－lit－inst－def by auto
from this and 〈orient－lit－inst Muvpos？$\vartheta^{\prime}$ 〉 have $e=(E q u v) \vee e=(E q v$
u）
unfolding orient－lit－inst－def by auto
from＜orient－lit－inst $M$ u v pos ？$\vartheta^{\prime}>$ have
$\left(\right.$（subst $u$ ？$\left.\vartheta^{\prime}\right),\left(\right.$ subst $\left.\left.v ? \vartheta^{\prime}\right)\right) \notin$ trm－ord
unfolding orient－lit－inst－def by auto
from $\left\langle M=(\right.$ Pos e）$\rangle$ and $\left\langle\right.$ subst－lit $\left.M^{\prime} \eta=M\right\rangle$
obtain $e^{\prime}$ where（subst－equation $\left.e^{\prime} \eta\right)=e$ and $M^{\prime}=\left(\right.$ Pos $\left.e^{\prime}\right)$
by（metis（no－types，opaque－lifting）subst－lit．simps（1）subst－lit．simps（2）atom．simps（1）
literal．distinct（1）positive－literal．elims（1））
from $\langle e=(E q u v) \vee e=(E q v u)\rangle$ and $\left\langle\left(\right.\right.$ subst－equation $\left.\left.e^{\prime} \eta\right)=e\right\rangle$
obtain $u^{\prime} v^{\prime}$ where $e^{\prime}=\left(E q u^{\prime} v^{\prime}\right) \vee\left(e^{\prime}=\left(E q v^{\prime} u^{\prime}\right)\right)$ and $\left(\right.$ subst $\left.u^{\prime} \eta\right)=u$
and
$\left(\right.$ subst $\left.v^{\prime} \eta\right)=v$
by（metis subst－equation．simps equation．inject subterms－of－eq．cases）
from 〈（（subst u ？$\left.\vartheta^{\prime}\right),\left(\right.$ subst $v$ ？$\left.\left.\vartheta^{\prime}\right)\right) \notin$ trm－ord〉
$\left\langle\left(\right.\right.$ subst $\left.u^{\prime} \eta\right)=u$ 〉
$\left\langle\left(\right.\right.$ subst $\left.\left.v^{\prime} \eta\right)=v\right\rangle$
have $\left(\left(\right.\right.$ subst $u^{\prime}$ ？$\left.\vartheta\right),\left(\right.$ subst $\left.\left.v^{\prime} ? \vartheta\right)\right) \notin$ trm－ord by simp
from this and $\left\langle M^{\prime}=\left(\right.\right.$ Pos $\left.\left.e^{\prime}\right)\right\rangle$ and $\left\langle e^{\prime}=\left(E q u^{\prime} v^{\prime}\right) \vee\left(e^{\prime}=\left(E q v^{\prime} u^{\prime}\right)\right)\right\rangle$
have orient－lit－inst $M^{\prime} u^{\prime} v^{\prime}$ pos ？$\vartheta$
unfolding orient－lit－inst－def by auto
from $\left\langle M^{\prime} \in C^{\prime}-\left\{L^{\prime}\right\}\right\rangle$ have vars－of－lit $M^{\prime} \subseteq$ vars－of－cl $C^{\prime}$ by auto
from this and 〈coincide－on $\sigma$ ？$\vartheta$（vars－of－cl $C^{\prime}$ ）〉 have coincide－on $\sigma$ ？$\vartheta$ （vars－of－lit $M^{\prime}$ ）
unfolding coincide－on－def by auto
from this have coincide－on ？V $\sigma$（vars－of－lit $M^{\prime}$ ）using coincide－sym by auto
from this and «orient－lit－inst $M^{\prime} u^{\prime} v^{\prime}$ pos ？$\vartheta>$ have orient－lit－inst $M^{\prime} u^{\prime} v^{\prime}$ pos
$\sigma$
using orient－lit－inst－coincide by auto
from 〈orient－lit－inst $M^{\prime} u^{\prime} v^{\prime}$ pos ？$\vartheta$ 〉 have vars－of $u^{\prime} \subseteq$ vars－of－lit $M^{\prime}$ and
vars－of $v^{\prime} \subseteq$ vars－of－lit $M^{\prime}$ using orient－lit－inst－vars by auto
from 〈vars－of $u^{\prime} \subseteq$ vars－of－lit $\left.M^{\prime}\right\rangle$ and＜coincide－on ？$v \sigma$（vars－of－lit $M^{\prime}$ ）〉
have coincide－on ？$V$（vars－of $u^{\prime}$ ）unfolding coincide－on－def by auto
from this have subst $u^{\prime} ? \vartheta=$ subst $u^{\prime} \sigma$ using coincide－on－term by metis
from this and $\left\langle\left(\right.\right.$ subst $\left.\left.u^{\prime} \eta\right)=u\right\rangle$ have subst $u$ ？$\vartheta^{\prime}=$ subst $u^{\prime} \sigma$ by simp
from 〈vars－of $v^{\prime} \subseteq$ vars－of－lit $\left.M^{\prime}\right\rangle$ and 〈coincide－on ？$\vartheta \sigma$（vars－of－lit $\left.M^{\prime}\right)$ 〉
have coincide－on ？$\vartheta \sigma$（vars－of $v^{\prime}$ ）unfolding coincide－on－def by auto
from this have subst $v^{\prime} ? \vartheta=$ subst $v^{\prime} \sigma$ using coincide－on－term by metis
from this and $\left\langle\left(\right.\right.$ subst $\left.\left.v^{\prime} \eta\right)=v\right\rangle$ have subst $v ? \vartheta^{\prime}=$ subst $v^{\prime} \sigma$ by simp
from $\left\langle\right.$ subst $v ? \vartheta^{\prime}=$ subst $\left.v^{\prime} \sigma\right\rangle\left\langle s^{\prime \prime}=\left(\right.\right.$ subst $\left.\left.v ? \vartheta^{\prime}\right)\right\rangle$
have $s^{\prime \prime}=\left(\right.$ subst $\left.v^{\prime} \sigma\right)$ by auto
from $\left\langle\right.$ subst $u ? \vartheta^{\prime}=$ subst $\left.u^{\prime} \sigma\right\rangle\left\langle t=\left(\right.\right.$ subst $\left.\left.u ? \vartheta^{\prime}\right)\right\rangle$
have $t=\left(\right.$ subst $\left.u^{\prime} \sigma\right)$ by auto
from $\left\langle s^{\prime \prime}=\left(\right.\right.$ subst $\left.\left.v^{\prime} \sigma\right)\right\rangle\left\langle t=\left(\right.\right.$ subst $\left.\left.u^{\prime} \sigma\right)\right\rangle$

```
        <orient-lit-inst M' u' v' pos \sigma\rangle\langleM'}\in\mp@subsup{M}{}{\prime}\in\mp@subsup{C}{}{\prime}-{\mp@subsup{L}{}{\prime}}
        have eq-occurs-in-cl t s''}(\mp@subsup{C}{}{\prime}-{\mp@subsup{L}{}{\prime}})
        unfolding eq-occurs-in-cl-def by auto
    from this and }\langle(\mp@subsup{s}{}{\prime\prime},t)\in\mathrm{ trm-ord }>\mathrm{ and }\langle(s,t)\in\mathrm{ trm-ord }>\mathrm{ and }«(\mathrm{ trm-rep s S)=
(trm-rep s" S)>
    have}\neg(\mathrm{ maximal-literal-is-unique t s C' L'S S) unfolding maximal-literal-is-unique-def
        by blast
    from this and assms(1) show False unfolding candidate-values-def by metis
qed
from <t'-bis = (subst t t }\eta\mathrm{ )>
    and <t = subst t' }\sigma
    have t= subst t'-bis (comp \eta}\mp@subsup{\eta}{}{\prime}\sigma
    using <subst t'-bis (comp \eta' \sigma) = subst t' \sigma> by auto
from 〈s'-bis = (subst s}\mp@subsup{s}{}{\prime}\eta)
    and }\langles=\mathrm{ subst s' }\sigma
    have}s=\mathrm{ subst }\mp@subsup{s}{}{\prime}\mathrm{ -bis (comp }\mp@subsup{\eta}{}{\prime}\sigma
    using <subst s'-bis (comp \eta' \sigma) = subst s' }\sigma\mathrm{ ` by auto
    from 〈CC-bis }\inS\rangle\langlet & subst-set (trms-ecl CC-bis) (comp \eta' \sigma)
    <trms-irreducible CC-bis (comp \eta' \sigma) S t\rangle
    <'-bis = cl-ecl CC-bis\rangle<(s,t)\in trm-ord }\langle<((s,t)\in\mathrm{ trm-ord }\longrightarrowz=\mathrm{ trm-rep
s S)>
    <orient-lit-inst L'-bis t'-bis s'-bis pos (comp \eta}\mp@subsup{\eta}{}{\prime}\sigma)
    <sel C'-bis = {}\rangle\langle\mp@subsup{L}{}{\prime}-bis\inC''-bis\rangle\langlemaximal-literal L C>
    <L= subst-lit L'-bis (comp \eta}\mp@subsup{\eta}{}{\prime}\sigma)
    <C= subst-cl C'-bis (comp \eta}\mp@subsup{\eta}{}{\prime}\sigma)
    \ground-clause C\rangle\langlet= subst t'-bis (comp \eta}\mp@subsup{\eta}{}{\prime}\sigma)
    <s= subst s'-bis (comp \eta}\mp@subsup{\eta}{}{\prime}\sigma)
    <inite C'-bis〉<smaller-lits-are-false t C S〉
    <maximal-literal-is-unique ts C'-bis L'-bis S (comp \eta}\mp@subsup{\eta}{}{\prime}\sigma)
    have (candidate-values z CC-bis C'-bis C s L L'-bis ? ?' t'-bis s'-bis t S)
    unfolding candidate-values-def by metis
    have vars-of-cl (cl-ecl D) \cap(vars-of-cl (cl-ecl CC-bis)) ={}
    proof (rule ccontr)
    assume vars-of-cl (cl-ecl D) \cap (vars-of-cl (cl-ecl CC-bis)) # {}
    from this and }\langle\mp@subsup{C}{}{\prime}\mathrm{ -bis = (cl-ecl CC-bis)>
        obtain x where x\invars-of-cl C'-bis and x\invars-of-cl (cl-ecl D) by auto
    from <x \in vars-of-cl C''bis\rangle
        obtain N where N\inC'-bis and x vars-of-lit N by auto
```



```
        N'}\in\mp@subsup{C}{}{\prime}\mathrm{ and }N=\mathrm{ subst-lit N' }\mp@subsup{N}{}{\prime}\eta\mathrm{ by auto
    from <x\in vars-of-lit N\rangle obtain e where N=(Pos e)\vee(N=(Nege))
        and }x\in\mathrm{ vars-of-eq e
    by (metis negative-literal.elims(2) negative-literal.elims(3) vars-of-lit.simps(1)
```

vars－of－lit．simps（2））
from $\langle N=($ Pos e $) \vee(N=(N e g e))\rangle$ and $\left\langle N=\right.$ subst－lit $\left.N^{\prime} \eta\right\rangle$ obtain $e^{\prime}$ where
$N^{\prime}=\left(\right.$ Pos $\left.e^{\prime}\right) \vee\left(N^{\prime}=\left(\right.\right.$ Neg $\left.\left.e^{\prime}\right)\right)$ and $e=$ subst－equation $e^{\prime} \eta$
by（metis subst－lit．elims atom．simps（1）atom．simps（2））
from $\langle x \in$ vars－of－eq $e\rangle$ obtain $u v$ where $e=(E q u v)$ and $x \in$ vars－of $u$ $\cup$ vars－of $v$
by（metis equation．exhaust vars－of－eq．simps）
from $\langle e=(E q u v)\rangle$ and $\left\langle e=\right.$ subst－equation $\left.e^{\prime} \eta\right\rangle$ obtain $u^{\prime} v^{\prime}$ where $e^{\prime}=$ （Equ＇$u^{\prime}$ ）
$u=\left(\right.$ subst $\left.u^{\prime} \eta\right)$ and $v=\left(\right.$ subst $\left.v^{\prime} \eta\right)$
by（metis subst－equation．simps equation．exhaust equation．inject）
from $\langle x \in$ vars－of $u \cup$ vars－of $v\rangle$ have $x \in$ vars－of $u \vee x \in$ vars－of $v$ by auto
then show False
proof
assume $x \in$ vars－of $u$
from this and $\left\langle u=\left(\right.\right.$ subst $\left.\left.u^{\prime} \eta\right)\right\rangle$
obtain $y$ where $y \in$ vars－of $u^{\prime}$ and $x \in \operatorname{vars-of}$（subst（Var $\left.y\right) \eta$ ）
using vars－of－instances［of $u^{\prime} \eta$ ］by auto
from $\left\langle y \in\right.$ vars－of $\left.u^{\prime}\right\rangle$ and $\left\langle e^{\prime}=\left(E q u^{\prime} v^{\prime}\right)\right\rangle$ have $y \in$ vars－of－eq $e^{\prime}$ by auto
from this and $\left\langle N^{\prime}=\left(\right.\right.$ Pos $\left.e^{\prime}\right) \vee\left(N^{\prime}=\left(\right.\right.$ Neg $\left.\left.\left.e^{\prime}\right)\right)\right\rangle$ have $y \in$ vars－of－lit $N^{\prime}$ by auto
from this and $\left\langle N^{\prime} \in C^{\prime}\right\rangle$ have $y \in$ vars－of－cl $C^{\prime}$ by auto
from this and 〈renaming $\eta$（vars－of－cl $C^{\prime}$ ）〉
have is－a－variable（subst（Var y）$\eta$ ）unfolding renaming－def by auto
from this and $\langle x \in$ vars－of（subst（Var $y$ ）$\eta$ ）〉 have
Var $x=($ subst $($ Var $y) \eta)$
by（metis is－a－variable．elims（2）singletonD vars－of．simps（1））
from this and $\left\langle y \in\right.$ vars－of－cl $\left.C^{\prime}\right\rangle$
have $x \in$（subst－codomain $\eta$（vars－of－cl $\left.C^{\prime}\right)$ ）unfolding subst－codomain－def by auto
from this and $\langle x \in$ vars－of－cl（cl－ecl D）＞
and «（（subst－codomain $\eta\left(\right.$ vars－of－cl $\left.\left.C^{\prime}\right)\right) \cap($ vars－of－cl $($ cl－ecl $\left.\left.D))\right)=\{ \}\right\rangle$
show False by auto
next
assume $x \in$ vars－of $v$
from this and $\left\langle v=\left(\right.\right.$ subst $\left.\left.v^{\prime} \eta\right)\right\rangle$
obtain $y$ where $y \in$ vars－of $v^{\prime}$ and $x \in \operatorname{vars-of~(subst~(Var~} y$ ）$\eta$ ）
using vars－of－instances $\left[\right.$ of $v^{\prime} \eta$ ］by auto
from $\left\langle y \in\right.$ vars－of $\left.v^{\prime}\right\rangle$ and $\left\langle e^{\prime}=\left(E q u^{\prime} v^{\prime}\right)\right\rangle$ have $y \in$ vars－of－eq $e^{\prime}$ by auto
from this and $\left\langle N^{\prime}=\left(\right.\right.$ Pos $\left.e^{\prime}\right) \vee\left(N^{\prime}=\left(\right.\right.$ Neg $\left.\left.\left.e^{\prime}\right)\right)\right\rangle$ have $y \in$ vars－of－lit $N^{\prime}$
by auto
from this and $\left\langle N^{\prime} \in C^{\prime}\right\rangle$ have $y \in$ vars－of－cl $C^{\prime}$ by auto
from this and 〈renaming $\eta$（vars－of－cl $C^{\prime}$ ）〉
have is－a－variable（subst（Var y）$\eta$ ）unfolding renaming－def by auto
from this and $\langle x \in$ vars－of（subst（Var $y$ ）$\eta$ ）〉 have
Var $x=($ subst $($ Var $y) \eta)$
by（metis is－a－variable．elims（2）singletonD vars－of．simps（1））

```
            from this and }\langley\invars-of-cl C'>
            have }x\in(\mathrm{ subst-codomain }\eta\mathrm{ (vars-of-cl C')) unfolding subst-codomain-def
by auto
            from this and <x vars-of-cl (cl-ecl D)\rangle
                and <((subst-codomain \eta (vars-of-cl C')) \cap (vars-of-cl (cl-ecl D))) = {}>
            show False by auto
        qed
    qed
    from this and «(candidate-values z CC-bis C'-bis C s L L'-bis ?\vartheta' t'-bis s'-bis
tS)>
            show ?thesis by blast
qed
lemma pos-ord-acyclic:
    shows acyclic (pos-ord C t)
by (simp add: acyclic-irrefl pos-ord-irrefl pos-ord-trans)
definition proper-subterm-red
    where proper-subterm-red tS \sigma=
        (\existsps. (subterm t p s ^ p\not= Nil ^(trm-rep (subst s \sigma)S\not=(subst s \sigma))))
```

The following lemma states that if an eligible term in a clause instance is not in normal form, then the clause instance must be reducible (according to the previous definition of reduction). This is the key lemma for proving completeness. Note that we assume that the considered substitution is in normal form, so that the reduction cannot occur inside a variable. We also rename the clause used for the reduction, to ensure that it shares no variable with the provided clause. The proof requires an additional hypothesis in the case where the reducible term occurs at the root position in an eligible term of a positive literal, see the first hypothesis below and function equivalent-eq-exists.

```
lemma reduction-exists:
    assumes polarity \(=\) neg \(\vee \neg\) equivalent-eq-exists \(t s(c l-e c l C)(\) int-clset \(S) \sigma L 1\)
        \(\checkmark\) proper-subterm-red t \(S \sigma\)
    assumes \(\forall x y .((x \in\) vars-of-cl \((\) cl-ecl \(C)) \longrightarrow(\) occurs-in \(y(\operatorname{subst}(\operatorname{Var} x) \sigma))\)
        \(\longrightarrow\) trm-rep y \(S=y\) )
    assumes eligible-literal L1 C \(\sigma\)
    assumes \((\) trm-rep \((\) subst \(t \sigma) S) \neq(\) subst \(t \sigma)\)
    assumes \(L 1 \in(\) cl-ecl \(C)\)
    assumes (orient-lit-inst L1 ts polarity \(\sigma\) )
    assumes \(\forall x \in S\). finite (cl-ecl x)
    assumes ground-clause (subst-cl (cl-ecl C) \(\sigma\) )
    assumes (fo-interpretation (same-values \((\lambda t\). \((\) trm-rep \(t S)))\) )
    assumes \(C \in S\)
    assumes Ball \(S\) well-constrained
    assumes all-trms-irreducible (subst-set (trms-ecl C) \(\sigma\) ) ( \(\lambda\) t. trm-rep \(t S\) )
    assumes \(\neg\) validate-ground-clause (int-clset \(S)(\) subst-cl (cl-ecl C) \(\sigma\) )
    assumes closed-under-renaming \(S\)
```

```
shows \exists G ' u u' p v D L2.
    ((reduction L1 C \sigma't s polarity L2 u u'p v D (same-values ( }\lambdat.(\mathrm{ trm-rep t S)))
S \sigma)
    \wedge(variable-disjoint C D))
proof -
```

The first step is to get the minimal reducible position in $t \triangleleft \sigma$ and the corresponding subterm $v$ ．

```
    let ?Redexes ={ p. \existsv. subterm(subst t \sigma) p v^ root-term Sv^ trm-rep v S
\not=v}
    have ?Redexes \subseteqpos-of (subst t \sigma)
    proof
        fix }x\mathrm{ assume }x\in\mathrm{ ?Redexes
        then have }\existsv.\mathrm{ subterm (subst t }\sigma\mathrm{ ) xv by blast
        then have position-in x (subst t \sigma) unfolding position-in-def by metis
        then show }x\in\mathrm{ pos-of (subst }t\sigma\mathrm{ ) by auto
    qed
    from this have finite ?Redexes using set-of-positions-is-finite [of (subst t \sigma)]
        using finite-subset by blast
    from assms(4) have st-red S (subst t \sigma) using subterms-of-irred-trms-are-irred
by blast
    from this obtain p' where }\mp@subsup{p}{}{\prime}\in\mathrm{ ?Redexes unfolding st-red-def by blast
    from〈finite ?Redexes〉 this obtain mp where mp\in?Redexes
        and }\\mp@subsup{p}{}{\prime}.(\mp@subsup{p}{}{\prime},mp)\in(\mathrm{ pos-ord C t) > p' # ?Redexes
        using pos-ord-acyclic [of C t] finite-proj-wf [of ?Redexes p' pos-ord C t] by blast
    have mr: minimal-redex mp (subst t \sigma) CSt
    proof (rule ccontr)
    assume \negminimal-redex mp (subst t \sigma) C S t
    from this obtain p'\prime v' where ( }\mp@subsup{p}{}{\prime\prime},mp)\in(\mathrm{ pos-ord C t) subterm (subst t }\sigma\mathrm{ ) p''
v
            and trm-rep v}\mp@subsup{v}{}{\prime}S\not=\mp@subsup{v}{}{\prime}\mathrm{ unfolding minimal-redex-def by blast
    show False
    proof (cases)
        assume (root-term S v')
        from this and <subterm (subst t \sigma) p'\prime v'〉\langletrm-rep v'S\not= v'>
            have p" }\in\mathrm{ ?Redexes by blast
```



```
E(pos-ord C t)>
            show False by blast
            next assume }\neg\mathrm{ (root-term S v')
            from <trm-rep v'}S\not=\mp@subsup{v}{}{\prime}\rangle have st-red S v
                    using subterms-of-irred-trms-are-irred by blast
            from this obtain p"\prime}\mp@subsup{v}{}{\prime\prime}\mathrm{ where subterm v}\mp@subsup{v}{}{\prime}\mp@subsup{p}{}{\prime\prime\prime}\mp@subsup{v}{}{\prime\prime}\mathrm{ root-term S v't trm-rep v"
S\not=\mp@subsup{v}{}{\prime\prime}
            unfolding st-red-def by blast
        from <subterm v' p'\prime\prime v'〉 and <subterm (subst t \sigma) p'\prime v'>
            have subterm (subst t \sigma) (append p}\mp@subsup{p}{}{\prime\prime}\mp@subsup{p}{}{\prime\prime\prime})\mp@subsup{v}{}{\prime\prime
```

using subterm－of－a－subterm－is－a－subterm by metis
from this and 〈trm－rep $\left.v^{\prime \prime} S \neq v^{\prime \prime}\right\rangle\left\langle r o o t-t e r m ~ S v^{\prime \prime}\right\rangle$
have（append $\left.p^{\prime \prime} p^{\prime \prime \prime}\right) \in$ ？Redexes by blast
from this and $\left\langle\backslash p^{\prime} .\left(p^{\prime}, m p\right) \in(\right.$ pos－ord $C t) \Longrightarrow p^{\prime} \notin$ ？Redexes $\rangle$
have（append $\left.p^{\prime \prime} p^{\prime \prime \prime}, m p\right) \notin($ pos－ord $C t)$ by blast
from this and $\left\langle\left(p^{\prime \prime}, m p\right) \in(\right.$ pos－ord $C$ t） show False using pos－ord－prefix by auto
qed
qed
 root－term $S v$
and trm－rep v $S \neq v$ unfolding st－red－def by blast
Second，we find the clause $C 2$ and substitution $\eta$ that are used to determine the value of $v$ according to the definition of trm－rep，and we prove that they satisfy all the desired properties．In particular，clause $C 2$ is renamed to ensure that it shares no variable with $C$ ．

```
    from <subterm (subst t \sigma) pv> have
    si:(\existsx q1 q2. (is-a-variable x) ^(subterm (subst x \sigma) q1 v) ^
            (subterm t q2 x) ^(p=(append q2 q1))) \vee
        ((\exists u. (\neg(is-a-variable u)^(subterm t p u)^(v=(subst u \sigma)))))
        using subterms-of-instances by metis
    let ?v = trm-rep vS
    from 〈trm-rep vS\not=v\rangle}\mathrm{ and «root-term S v> have ?v v min-trms (set-of-candidate-values
Sv)
            unfolding root-term-def get-min-def by (metis some-in-eq)
    from <?v | min-trms (set-of-candidate-values S v)\rangle obtain pair where ?v =fst
pair
    and pair \in (set-of-candidate-values S v) and
    min-pair: ( }\forall\mathrm{ pair'єset-of-candidate-values S v. (snd pair', snd pair) }\not\intrm-ord
    unfolding min-trms-def by blast
    from <pair }\in\mathrm{ (set-of-candidate-values S v)> have
    \existszCC C'CsLL'\sigma 知 s'. pair = (z,s)^(candidate-values z CC C'}C\mathrm{ s L L'
\sigmat's}\mp@subsup{s}{}{\prime}vS
            unfolding set-of-candidate-values-def [of S v] by blast
    from this obtain zz C2-init Cl-C2-init gr-Cl-C2 gr-rhs gr-L2 L2-init \eta-init
lhs-init rhs-init
    where pair = (zz, gr-rhs )
    and (candidate-values zz C2-init Cl-C2-init gr-Cl-C2 gr-rhs gr-L2 L2-init
        \eta-init lhs-init rhs-init v S)
    by blast
from \(\operatorname{assms}(7)\) and \(\langle C \in S\rangle\) have finite（cl－ecl \(C\) ）by auto
from 〈（candidate－values zz C2－init Cl－C2－init gr－Cl－C2 gr－rhs gr－L2 L2－init \(\eta\)－init lhs－init rhs－init \(v S\) ）＞
have finite Cl －C2－init unfolding candidate－values－def by metis
```

from $\operatorname{assms}(11)\langle$ closed－under－renaming $S\rangle\langle$ finite $C l-C 2$－init〉〈finite（cl－ecl C）〉
〈（candidate－values zz C2－init Cl－C2－init gr－Cl－C2 gr－rhs gr－L2 L2－init $\eta$－init lhs－init rhs－init $v S$ ）＞
obtain C2 Cl－C2 $\eta$ L2 lhs rhs where
（candidate－values zz C2 Cl－C2 gr－Cl－C2 gr－rhs gr－L2 L2 $\eta$ lhs rhs v S）
and $($ vars－of－cl $($ cl－ecl $C) \cap$ vars－of－cl $(c l-e c l ~ C 2)) ~=\{ \}$
using candidate－values－renaming［of zz C2－init Cl－C2－init gr－Cl－C2 gr－rhs gr－L2 L2－init $\eta$－init lhs－init rhs－init v $S C$ ］by auto
from 〈（candidate－values zz C2 Cl－C2 gr－Cl－C2 gr－rhs gr－L2 L2 $\eta$ lhs rhs v S）〉
and $\langle$ pair $=(z z, \quad g r-r h s)\rangle$ and $\langle ? v=$ fst pair $\rangle$
have cv：（candidate－values ？v C2 Cl－C2 gr－Cl－C2 gr－rhs gr－L2 L2 $\eta$ lhs rhs $v$
S）
by（metis fst－conv）
from $c v$ have $C 2 \in S$
unfolding candidate－values－def by metis
from $c v$ have ground－clause gr－Cl－C2
unfolding candidate－values－def by metis
from $\operatorname{assms}(7)$ and $\operatorname{assms}(10)$ have finite（vars－of－cl（cl－ecl C））
using set－of－variables－is－finite－cl by blast
from $c v$ have smaller－lits－are－false $v \mathrm{gr}$－Cl－C2 $S$
unfolding candidate－values－def by metis
from $c v$ have $g r-C l-C 2=s u b s t-c l C l-C 2 \eta$ unfolding candidate－values－def by metis
from $c v$ have orient－lit－inst L2 lhs rhs pos $\eta$ unfolding candidate－values－def by metis
from $c v$ have maximal－literal gr－L2 gr－Cl－C2 unfolding candidate－values－def by metis
from $c v$ have $g r-L 2=$ subst－lit L2 $\eta$ unfolding candidate－values－def by metis
from $c v$ have ground－clause $\mathrm{gr}-\mathrm{Cl}$－ C 2
unfolding candidate－values－def by metis
from $c v$ have $L 2 \in C l-C 2$ unfolding candidate－values－def by metis
from this and $\langle g r-C l-C 2=s u b s t-c l C l-C 2 \quad \eta\rangle$ and $\langle g r-L 2=s u b s t-l i t L 2 ~ \eta\rangle$
have $g r-L 2 \in g r-C l-C 2$ by auto
from $c v$ have trm－rep $v S=$ trm－rep gr－rhs $S$ unfolding candidate－values－def by metis
from $c v$ have（ $g r$－rhs,$v$ ）$\in$ trm－ord unfolding candidate－values－def by metis
from $c v$ have $C l-C 2=c l-e c l C 2$ unfolding candidate－values－def by metis
from $c v$ have $v \notin$ subst－set（trms－ecl C2）$\eta$ unfolding candidate－values－def by metis

```
    from \(c v\) have sel (cl-ecl C2) \(=\{ \}\)
        unfolding candidate-values-def by metis
    from this and 〈maximal-literal gr-L2 gr-Cl-C2〉 and \(\langle g r-C l-C 2=s u b s t-c l ~ C l-C 2\)
\(\eta\) >
    and \(\langle C l-C 2=(\) cl-ecl C2) \(\rangle\) and \(\langle g r-L 2=\) subst-lit L2 \(\eta\rangle\) have eligible-literal
L2 C2 \(\eta\)
    unfolding eligible-literal-def by auto
    from \(c v\) have ( \(g r\)-rhs,\(v\) ) \(\in\) trm-ord
        unfolding candidate-values-def by metis
```

    from \(c v\) have norm: \(\left(\forall x .\left(\exists x^{\prime} \in\right.\right.\) trms-ecl C2. occurs-in \(x\left(\right.\) subst \(\left.\left.x^{\prime} \eta\right)\right) \longrightarrow\)
        \((x, v) \in\) trm-ord \(\longrightarrow\) trm-rep \(x S=x)\)
        unfolding candidate-values-def trms-irreducible-def by metis
    from 〈ground-clause gr-Cl-C2〉 and 〈gr-L2 \(\in g r-C l-C 2\rangle\) have vars-of-lit gr-L2
    $=\{ \}$
by auto
from $c v$ have $v=$ subst lhs $\eta$ unfolding candidate-values-def by metis
from $c v$ have $g r-r h s=s u b s t ~ r h s ~ \eta$ unfolding candidate-values-def by metis
let ? $I=($ same-values $(\lambda t .($ trm-rep $t S)))$
have no-fact: $\neg$ equivalent-eq-exists lhs rhs Cl-C2 (same-values ( $\lambda t$. trm-rep $t$
S)) $\eta L 2$
proof
assume equivalent-eq-exists lhs rhs Cl-C2 (same-values ( $\lambda$ t. trm-rep $t S)$ ) $\eta$ L2
from this have $\exists L \in C l-C 2-\{L 2\} . \exists u$ v. orient-lit-inst $L$ u v pos $\eta \wedge$
subst lhs $\eta=$ subst $u \eta \wedge$ same-values ( $\lambda$ t. trm-rep $t S$ ) (subst rhs $\eta$ ) (subst
$v \eta)$
unfolding equivalent-eq-exists-def
by blast
from this obtain $M$ where $M \in C l-C 2-\{L 2\}$ and $e: \exists u v$. orient-lit-inst $M$
$u$ v pos $\eta \wedge$
subst lhs $\eta=$ subst $u \eta \wedge$ same-values ( $\lambda$ t. trm-rep $t S$ ) (subst rhs $\eta$ ) (subst
$v \eta)$
by blast
from $e$ obtain $u^{\prime} v^{\prime}$ where orient-lit-inst $M u^{\prime} v^{\prime}$ pos $\eta$
and $i$ : subst lhs $\eta=$ subst $u^{\prime} \eta \wedge$ same-values ( $\lambda t$. trm-rep $t S$ ) (subst rhs
$\eta)\left(\right.$ subst $v^{\prime} \eta$ )
by blast
from $i$ have subst lhs $\eta=$ subst $u^{\prime} \eta$ by blast
from $i$ have trm－rep（subst rhs $\eta$ ）$S=$ trm－rep（subst $v^{\prime} \eta$ ）$S$ unfolding same－values－def by blast
let $? u^{\prime}=\left(\right.$ subst $\left.u^{\prime} \eta\right)$
let $? v^{\prime}=\left(\right.$ subst $\left.v^{\prime} \eta\right)$
from＜orient－lit－inst $M u^{\prime} v^{\prime}$ pos $\eta$ 〉 have orient－lit（subst－lit M $\eta$ ）？$u^{\prime}$ ？$v^{\prime}$ pos using lift－orient－lit by auto
from «orient－lit－inst L2 lhs rhs pos $\eta\rangle$ have orient－lit（subst－lit L2 $\eta$ ）（subst lhs
$\eta)$
（subst rhs $\eta$ ）pos
using lift－orient－lit by auto
from «orient－lit－inst $M u^{\prime} v^{\prime}$ pos $\left.\eta\right\rangle$ and $\langle M \in(C l-C 2-\{L 2\})\rangle$ and
$\langle g r-C l-C 2=$ subst－cl Cl－C2 $\eta\rangle$
have eq－occurs－in－cl ？$u^{\prime}$ ？$v^{\prime}(C l-C 2-\{L 2\}) ~ \eta$ unfolding eq－occurs－in－cl－def
by auto
from 〈MECl－C2－\｛L2\}〉 and 〈gr-Cl-C2 = subst-cl Cl-C2 $\eta$ 〉
have（subst－lit $M \eta) \in($ subst－cl $(C l-C 2-\{L 2\}) \eta)$ by auto
from $\langle M \in C l-C 2-\{L 2\}\rangle$ and $\langle g r-C l-C 2=$ subst－cl Cl－C2 $\eta\rangle$ have（subst－lit $M \eta) \in g r-C l-C 2$ by auto
from $\langle$ vars－of－lit gr－L2 $=\{ \}\rangle$ and $\langle g r-L 2=$ subst－lit L2 $\eta\rangle$ ＜orient－lit（subst－lit L2 $\eta$ ）（subst lhs $\eta$ ）（subst rhs $\eta$ ）pos〉 have vars－of（subst rhs $\eta$ ）$=\{ \}$ using orient－lit－vars by blast
from〈ground－clause gr－Cl－C2〉 and 〈（subst－lit M $\eta$ ）$\in g r-C l-C 2\rangle$ have vars－of－lit（subst－lit $M \eta)=\{ \}$ by auto
from this and＜orient－lit（subst－lit $M \eta$ ）？$u^{\prime}$ ？$v^{\prime}$ pos〉
have vars－of ？$v^{\prime}=\{ \}$ using orient－lit－vars by blast
from 〈maximal－literal gr－L2 gr－Cl－C2〉 and 〈（subst－lit M $\eta$ ）$\in$ gr－Cl－C2〉
have（gr－L2，（subst－lit M $\eta$ ））$\notin$ lit－ord
unfolding maximal－literal－def by auto
from this and «orient－lit（subst－lit $M \eta$ ）？$u^{\prime}$ ？$v^{\prime}$ pos〉
and «orient－lit（subst－lit L2 $\eta$ ）（subst lhs $\eta$ ）（subst rhs $\eta$ ）pos〉 and 〈subst lhs $\eta=$ subst $u^{\prime} \eta$ ）
and $\langle$ vars－of－lit gr－L2 $=\{ \}\rangle$ and $\langle v a r s-o f-l i t ~(s u b s t-l i t ~ M ~ \eta)=\{ \}\rangle$ and $\langle g r$－L2 $=$ subst－lit L2 $\eta\rangle$ have $\left((\right.$ subst rhs $\left.\eta), ? v^{\prime}\right) \notin$ trm－ord using lit－ord－rhs by auto
from this and «vars－of ？$\left.v^{\prime}=\{ \}\right\rangle$ and «vars－of（subst rhs $\eta$ ）$\left.=\{ \}\right\rangle$ have $? v^{\prime}=($ subst rhs $\eta) \vee\left(? v^{\prime},(\right.$ subst rhs $\left.\eta)\right) \in$ trm－ord using trm－ord－ground－total unfolding ground－term－def by auto
from this and $\langle(g r-r h s, v) \in t r m$－ord $\rangle$ and $\langle g r$－rhs $=$ subst rhs $\eta\rangle$ have $\left(? v^{\prime}, v\right) \in$ trm－ord using trm－ord－trans unfolding trans－def by auto
from $c v$ have maximal－literal－is－unique v gr－rhs Cl－C2 L2 S $\eta$ unfolding candidate－values－def by metis
from＜orient－lit－inst $M u^{\prime} v^{\prime}$ pos $\left.\eta\right\rangle$ have（ $\left(\right.$ subst $\left.u^{\prime} \eta\right),\left(\right.$ subst $\left.\left.v^{\prime} \eta\right)\right) \notin$ trm－ord unfolding orient－lit－inst－def by auto
have trm－rep gr－rhs $S \neq$ trm－rep（subst $v^{\prime} \eta$ ）$S$
by $\left(\right.$ metis $\left\langle\left(s u b s t v^{\prime} \eta, v\right) \in\right.$ trm－ord $\rangle\langle(g r-r h s, v) \in$ trm－ord $\rangle$〈subst lhs $\eta=$ subst $\left.u^{\prime} \eta\right\rangle$〈eq－occurs－in－cl（subst $u^{\prime} \eta$ ）（subst $v^{\prime} \eta$ ）（Cl－C2－\｛L2\}) $\left.\eta\right\rangle$〈maximal－literal－is－unique v gr－rhs Cl－C2 L2 $S \eta\rangle\langle v=$ subst lhs $\eta\rangle$ maximal－literal－is－unique－def）
from this and 〈trm－rep（subst rhs $\eta$ ）$S=$ trm－rep（subst $v^{\prime} \eta$ ）$\left.S\right\rangle$
and $\langle g r-r h s=($ subst rhs $\eta)\rangle$ show False by blast
qed
from this $\langle g r-C l-C 2=s u b s t-c l ~ C l-C 2 ~ \eta 〉$
and $\langle g r-L 2=$ subst－lit L2 $\eta\rangle$
and 〈smaller－lits－are－false $v \mathrm{gr}-\mathrm{Cl}-\mathrm{C} 2 \mathrm{~S}$ 〉 and $\operatorname{assms}(9)$ and＜orient－lit－inst L2 lhe rhs pos $\eta$＞
and 〈maximal－literal gr－L2 gr－Cl－C2〉
and〈ground－clause gr－Cl－C2〉
and $\langle g r-L 2 \in g r-C l-C 2\rangle$ and $\langle v=$ subst lhs $\eta\rangle\langle g r-r h s=s u b s t$ rhs $\eta\rangle$
and 〈trm－rep $v S=$ trm－rep gr－rhs $S\rangle$
have（ $\neg$ validate－ground－clause ？I（subst－cl（ Cl－C2－\｛L2 \}) $\eta)$ ）
using if－all－smaller－are－false－then－cl－not－valid［of lhs $\eta$ Cl－C2 S L2 rhs］by blast
We fuse the substitutions $\sigma$ and $\eta$ so that the superposition rule can be applied：
from 〈ground－clause（subst－cl（cl－ecl C）$\sigma$ ）〉
have ground－on（vars－of－cl（cl－ecl C））$\sigma$ using ground－clauses－and－ground－substs by auto
from〈finite（vars－of－cl（cl－ecl C））〉〈（vars－of－cl（cl－ecl C）$\cap$ vars－of－cl（cl－ecl C2）$)=\{ \}$＞

〈ground－on（vars－of－cl（cl－ecl C））$\sigma$ 〉 obtain $\sigma^{\prime}$ where
coincide－on $\sigma^{\prime} \sigma($ vars－of－cl（cl－ecl $\left.C)\right)$ and coincide－on $\sigma^{\prime} \eta$（vars－of－cl（cl－ecl C2））
using combine－substs［of（vars－of－cl（cl－ecl C））（vars－of－cl（cl－ecl C2））$\sigma \eta]$ by blast
from 〈coincide－on $\sigma^{\prime} \sigma($ vars－of－cl（cl－ecl $\left.\left.C)\right)\right\rangle$ have coincide－on $\sigma \sigma^{\prime}(v a r s-o f-c l$ （cl－ecl C））
using coincide－sym by auto
from 〈coincide－on $\sigma^{\prime} \eta($ vars－of－cl（cl－ecl C2）$\left.)\right\rangle$ have coincide－on $\eta \sigma^{\prime}$（vars－of－cl （cl－ecl C2））
using coincide－sym by auto
from＜eligible－literal L1 $C \sigma\rangle\langle L 1 \in($ cl－ecl $C)\rangle\left\langle\right.$ coincide－on $\sigma \sigma^{\prime}$（vars－of－cl $($ cl－ecl C））＞
have eligible－literal L1 C $\sigma^{\prime}$ using eligible－literal－coincide by auto
from 〈eligible－literal L2 C2 $\eta\rangle\langle L 2 \in C l-C 2\rangle\langle C l-C 2=(c l-e c l C 2)\rangle\langle c o i n c i d e-o n$ $\eta \sigma^{\prime}$
（vars－of－cl（cl－ecl C2））＞
have eligible－literal L2 C2 $\sigma^{\prime}$ using eligible－literal－coincide by auto
from 〈ground－clause $g r-C l-C 2\rangle$ and $\langle g r-C l-C 2=(s u b s t-c l C l-C 2 \eta)\rangle$
have ground－clause（subst－cl Cl－C2 $\sigma^{\prime}$ ）
by（metis $\left\langle C l-C 2=\right.$ cl－ecl C2〉〈coincide－on $\sigma^{\prime} \eta($ vars－of－cl（cl－ecl C2）$\left.)\right\rangle$ coin－ cide－on－cl）
from 〈coincide－on $\sigma \sigma^{\prime}($ vars－of－cl $($ cl－ecl $\left.C))\right\rangle\langle L 1 \in($ cl－ecl $C)\rangle$ have coincide－on $\sigma \sigma^{\prime}$（vars－of－lit L1）unfolding coincide－on－def by auto
from 〈coincide－on $\eta \sigma^{\prime}($ vars－of－cl（cl－ecl C2））$\rangle\langle L 2 \in C l-C 2\rangle$ and $\langle C l-C 2=$ （cl－ecl C2）＞
have coincide－on $\eta \sigma^{\prime}$（vars－of－lit L2）unfolding coincide－on－def by auto
from $\langle($ orient－lit－inst L1 $t$ s polarity $\sigma)\rangle$ and 〈coincide－on $\sigma \sigma^{\prime}($ vars－of－lit L1）〉
have（orient－lit－inst L1 ts polarity $\sigma^{\prime}$ ）
using orient－lit－inst－coincide［of L1 tspolarity $\sigma \sigma^{\dagger}$ ］by blast
from 〈（orient－lit－inst L2 lhs rhs pos $\eta$ ）〉 and 〈coincide－on $\eta \sigma^{\prime}($ vars－of－lit L2）〉
have（orient－lit－inst L2 lhs rhs pos $\sigma^{\prime}$ ）using orient－lit－inst－coincide by blast
To prove that the superposition rule is applicable，we need to show that $v$ does not occur inside a variable：
have $\neg(\exists x q 1 q 2 .($ is－a－variable $x) \wedge($ subterm $($ subst $x \sigma) q 1 v) \wedge$ $($ subterm $t q 2 x) \wedge(p=($ append $q 2 q 1)))$
proof
assume $(\exists x$ q1 q2．（is－a－variable $x) \wedge($ subterm $($ subst $x \sigma) q 1 v) \wedge$
（subterm $t$ q2 $x) \wedge(p=($ append $q 2$ q1）$))$
then obtain $x q 1 q 2$ where $i s$－a－variable $x$ subterm（subst $x \sigma$ ）q1 $v$
（subterm（subst $x \sigma$ ）q1 $v$ ）（subterm $t q 2 x$ ）by auto
from $\langle($ subterm（subst $x \sigma) q 1 v)\rangle$ have occurs－in $v($ subst $x \sigma)$
unfolding occurs－in－def by auto
from 〈is－a－variable $x\rangle$ obtain $x^{\prime}$ where $x=\operatorname{Var} x^{\prime}$ using is－a－variable．elims（2） by blast
from $\langle$ subterm $t$ q2 $x$ 〉 have $x \in$ subterms－of $t$
using subterms－of．simps unfolding occurs－in－def by blast
from this have $x \in \operatorname{subterms-of-lit~L1~using~} \operatorname{assms}(6)$ by（simp add：ori－ ent－lit－inst－subterms）
from this $\langle L 1 \in($ cl－ecl $C)\rangle$ have $x \in$ subterms－of－cl（cl－ecl $C$ ）by auto
from this have vars－of $x \subseteq$ vars－of－cl（cl－ecl C）using subterm－vars by blast
from this and $\left\langle x=\left(\operatorname{Var} x^{\prime}\right)\right\rangle$ have $x^{\prime} \in$ vars－of－cl（cl－ecl $C$ ）by auto
from $\left\langle x^{\prime} \in\right.$ vars－of－cl（cl－ecl $\left.\left.C\right)\right\rangle\langle o c c u r s-i n v($ subst $x \sigma)$ ）
$\left\langle x=\operatorname{Var} x^{\prime}\right\rangle \operatorname{assms}(2)$ have trm－rep $v S=v$ by blast
from this and 〈trm－rep $v S \neq v$ 〉 show False by blast
qed
from this and si obtain $u$ where $\neg($ is－a－variable $u)($ subterm t $p u)$ and $v=$ （subst u $\sigma$ ）
by auto
from＜orient－lit－inst L1 $t$ s polarity $\sigma\rangle$ have vars－of $t \subseteq$ vars－of－lit L1 using orient－lit－inst－vars by auto
from 〈subterm t $p u\rangle$ have vars－of $u \subseteq$ vars－of $t$ using vars－subterm by auto
from 〈vars－of $t \subseteq$ vars－of－lit L1〉〈vars－of $u \subseteq$ vars－of $t\rangle\left\langle\right.$ coincide－on $\sigma \sigma^{\prime}$ （vars－of－lit L1）＞
have coincide－on $\sigma \sigma^{\prime}$（vars－of $u$ ）unfolding coincide－on－def by blast
from this have subst $u \sigma=$ subst $u \sigma^{\prime}$ using coincide－on－term by auto
from 〈orient－lit－inst L2 lhs rhs pos $\eta$ 〉 have vars－of lhs $\subseteq$ vars－of－lit L2 and vars－of rhs $\subseteq$ vars－of－lit L2 using orient－lit－inst－vars by auto
from 〈vars－of lhs $\subseteq$ vars－of－lit L2〉〈coincide－on $\eta \sigma^{\prime}($ vars－of－lit L2）〉 have coincide－on $\eta \sigma^{\prime}$（vars－of lhs）unfolding coincide－on－def by blast
from this have subst lhs $\eta=$ subst lhs $\sigma^{\prime}$
using coincide－on－term by auto
from 〈vars－of rhs $\subseteq$ vars－of－lit L2〉〈coincide－on $\eta \sigma^{\prime}($ vars－of－lit L2）〉 have coincide－on $\eta \sigma^{\prime}$（vars－of rhs）unfolding coincide－on－def by blast
from this have subst rhs $\eta=$ subst rhs $\sigma^{\prime}$ using coincide－on－term by auto
from 〈trm－rep v $S=$ trm－rep gr－rhs $S\rangle$ and $\langle v=$ subst lhs $\eta\rangle$ and $\langle g r$－rhs $=$ （subst rhs $\eta$ ）＞
have trm－rep（subst rhs $\eta$ ）$S=$ trm－rep（subst lhs $\eta$ ）$S$ by metis
from this and «subst rhs $\eta=$ subst rhs $\left.\sigma^{\prime}\right\rangle\left\langle s u b s t ~ l h s ~ \eta=\right.$ subst lhs $\sigma^{\prime}$ 〉 have trm－rep（subst rhs $\sigma^{\prime}$ ）$S=$ trm－rep（subst lhs $\sigma^{\prime}$ ）$S$ by metis
from 〈subst lhs $\eta=$ subst lhs $\left.\sigma^{\prime}\right\rangle\left\langle\right.$ subst $u \sigma=$ subst $\left.u \sigma^{\prime}\right\rangle\langle v=$ subst $u \sigma\rangle$ and $\langle v=$ subst lhs $\eta\rangle$
have subst $u \sigma^{\prime}=$ subst lhs $\sigma^{\prime}$ by auto
from 〈coincide－on $\sigma^{\prime} \eta$（vars－of－cl（cl－ecl C2））〉 and 〈Cl－C2 $=($ cl－ecl C2）$)$ have coincide－on $\sigma^{\prime} \eta$（vars－of－cl（Cl－C2－\｛ L2 \})) unfolding coincide－on－def by auto
from this and «（ $\neg$ validate－ground－clause？ ？（subst－cl（Cl－C2－\｛ L2 \} ) $\eta$ ））〉 have（ $\neg$ validate－ground－clause ？I（subst－cl（ Cl－C2－\｛ L2 \} ) $\sigma^{\prime}$ ）） using coincide－on－cl by metis
have $\left(\forall x \in\right.$ cl－ecl C2 $-\{$ L2 $\} .\left(\right.$ subst－lit $x \sigma^{\prime}$ ，subst－lit L2 $\left.\sigma^{\prime}\right) \in$ lit－ord $)$
proof
fix $x$ assume $x \in c l-e c l C 2-\{L 2\}$
from $\langle L 2 \in C l-C 2\rangle$ and $\langle g r-L 2=($ subst－lit $L 2 \eta)\rangle$ $\langle g r-C l-C 2=(s u b s t-c l C l-C 2 \eta)\rangle$ have $g r-L 2 \in g r-C l-C 2$ by auto
from this and 〈ground－clause gr－Cl－C2〉 have vars－of－lit gr－L2 $=\{ \}$ by auto
from $\langle x \in c l-e c l C 2-\{L 2\}\rangle$ and $\langle C l-C 2=(c l-e c l C 2)\rangle\langle g r-C l-C 2=(s u b s t-c l$ Cl－C2 7 ）＞
have（subst－lit $x \eta$ ）$\in$ gr－Cl－C2 by auto
from this and 〈ground－clause gr－Cl－C2〉 have vars－of－lit（subst－lit $x \quad \eta)=\{ \}$ by auto
from this $\langle x \in$ cl－ecl C2－$\{$ L2 $\}\rangle\langle$ maximal－literal $g r-L 2$ gr－Cl－C2〉〈Cl－C2 $=$ cl－ecl C2）
$\langle g r-L 2=($ subst－lit L2 $\eta)\rangle$


```
assms(9)
    〈vars-of-lit gr-L2 \(=\{ \}\rangle\langle\) vars-of-lit (subst-lit \(x \eta)=\{ \}\rangle\)
    have (subst-lit \(x \quad \eta\), subst-lit L2 \(\eta\) ) \(\in\) lit-ord
    using max-pos-lit-dominates-cl [of L2 \(\eta\) Cl-C2 lhs rhs \(x\) ? I] by metis
    from \(\langle L 2 \in C l-C 2\rangle\) have vars-of-lit L2 \(\subseteq\) vars-of-cl Cl-C2 by auto
    from this and «coincide-on \(\sigma^{\prime} \eta\) (vars-of-cl (cl-ecl C2))〉 and \(\langle C l-C 2=c l-e c l\)
C2 \({ }^{\text {> }}\)
            have coincide-on \(\sigma^{\prime} \eta\) (vars-of-lit L2) unfolding coincide-on-def by auto
    from this have subst-lit L2 \(\sigma^{\prime}=\) subst-lit L2 \(\eta\) using coincide-on-lit by auto
    from \(\langle x \in(\) cl-ecl C2) \(-\{L 2\}\rangle\) have \(x \in\) cl-ecl C2 by auto
    from this have vars-of-lit \(x \subseteq\) vars-of-cl (cl-ecl C2) by auto
    from this and «coincide-on \(\sigma^{\prime} \eta\) (vars-of-cl (cl-ecl C2))〉
            have coincide-on \(\sigma^{\prime} \eta\) (vars-of-lit \(x\) ) unfolding coincide-on-def by auto
    from this have subst-lit \(x \sigma^{\prime}=\) subst-lit \(x \eta\) using coincide-on-lit by auto
    from 〈(subst-lit \(x ~ \eta\), subst-lit L2 \(\eta) \in\) lit-ord \(〉\)
        \(\left\langle\right.\) subst-lit L2 \(\sigma^{\prime}=\) subst-lit L2 \(\left.\eta\right\rangle\)
        «subst-lit \(x \sigma^{\prime}=\) subst-lit \(\left.x \eta\right\rangle\)
        show (subst-lit \(x \sigma^{\prime}\),subst-lit L2 \(\sigma^{\prime}\) ) \(\in\) lit-ord by metis
    qed
    have all-trms-irreducible (subst-set (trms-ecl C2) \(\left.\sigma^{\prime}\right)(\lambda t\). trm-rep \(t S)\)
    proof (rule ccontr)
    assume \(\neg\) all-trms-irreducible (subst-set (trms-ecl C2) \(\sigma^{\prime}\) ) ( \(\lambda\) t. trm-rep \(t S\) )
    from this obtain \(x y\) where \(x \in\) (subst-set (trms-ecl C2) \(\sigma^{\prime}\) ) and occurs-in \(y\)
\(x\)
        and trm-rep y \(S \neq y\) unfolding all-trms-irreducible-def by blast
    from \(\left\langle x \in\right.\) (subst-set (trms-ecl C2) \(\sigma^{\prime}\) ) \(\rangle\) obtain \(x^{\prime}\) where \(x^{\prime} \in\) trms-ecl C2
        and \(x=\left(\right.\) subst \(\left.x^{\prime} \sigma^{\prime}\right)\) by auto
    from \(\operatorname{assms}(11)\) and \(\left\langle x^{\prime} \in(\right.\) trms-ecl C2) \(\rangle\) and \(\langle C 2 \in S\rangle\)
        have dom-trm \(x^{\prime}\) (cl-ecl C2) unfolding Ball-def well-constrained-def by blast
    from this obtain \(x^{\prime \prime}\)
        where \(x^{\prime \prime} \in\) subterms-of-cl (cl-ecl C2) and \(x^{\prime \prime}=x^{\prime} \vee\left(x^{\prime}, x^{\prime \prime}\right) \in\) trm-ord
        using dom-trm-lemma by blast
    from 〈dom-trm \(x^{\prime}\left(\right.\) cl-ecl C2)〉 have vars-of \(x^{\prime} \subseteq\) vars-of-cl (cl-ecl C2)
        using dom-trm-vars by blast
    from this and «coincide-on \(\sigma^{\prime} \eta\) (vars-of-cl (cl-ecl C2))〉 have coincide-on \(\sigma^{\prime}\)
\(\eta\) (vars-of \(x^{\prime}\) )
            unfolding coincide-on-def by auto
    from this have (subst \(\left.x^{\prime} \eta\right)=\left(\right.\) subst \(\left.x^{\prime} \sigma^{\prime}\right)\) using coincide-on-term by metis
    from this and \(\left\langle x=\left(\right.\right.\) subst \(\left.\left.x^{\prime} \sigma^{\prime}\right)\right\rangle\) have \(x=\left(\right.\) subst \(\left.x^{\prime} \eta\right)\) by auto
    from this and \(\left\langle x^{\prime} \in\right.\) trms-ecl C2〉 have \(x \in(\) subst-set (trms-ecl C2) \(\eta\) )
        by auto
    from \(\left\langle x^{\prime \prime} \in\right.\) (subterms-of-cl (cl-ecl C2)) \(\rangle\) have
            (subst \(\left.x^{\prime \prime} \eta\right) \in(\) subterms-of-cl (subst-cl (cl-ecl C2) \(\eta)\) )
            using subterm-cl-subst [of \(x^{\prime \prime}\) cl-ecl C2] by auto
```

from 〈orient－lit－inst L2 lhs rhs pos $\eta\rangle\langle g r-r h s=($ subst rhs $\eta)\rangle$
$\langle g r-L 2=($ subst－lit L2 $\eta)\rangle$
have orient－lit gr－L2（subst lhs $\eta$ ）gr－rhs pos
using lift－orient－lit
by auto
from 〈ground－clause gr－Cl－C2〉 have vars－of－cl gr－Cl－C2 $=\{ \}$ by auto
from 〈vars－of－lit gr－L2 $=\{ \}\rangle\langle v a r s-o f-c l ~ g r-C l-C 2=\{ \} 〉$
〈（subst $\left.x^{\prime \prime} \eta\right) \in($ subterms－of－cl（subst－cl（cl－ecl C2）$\eta)$ ）〉
〈orient－lit gr－L2（subst lhs $\eta$ ）gr－rhs pos〉〈maximal－literal gr－L2 gr－Cl－C2〉
$\langle C l-C 2=$ cl－ecl C2 $\rangle\langle g r-L 2=($ subst－lit L2 $\eta)\rangle$
$\langle g r-C l-C 2=($ subst－cl Cl－C2 $\eta)\rangle\langle v=($ subst lhs $\eta)\rangle\langle v=($ subst lhs $\eta)\rangle$
have $\left(\right.$ subst $\left.x^{\prime \prime} \eta\right)=v \vee\left(\left(\left(\right.\right.\right.$ subst $\left.\left.x^{\prime \prime} \eta\right), v\right) \in$ trm－ord $)$
using subterms－dominated［of gr－L2 gr－Cl－C2（subst lhs $\eta$ ）gr－rhs pos subst $x^{\prime \prime} \eta$ ］
by metis
from $\left\langle x^{\prime \prime}=x^{\prime} \vee\left(x^{\prime}, x^{\prime \prime}\right) \in\right.$ trm－ord $\rangle\left\langle x=\left(\right.\right.$ subst $\left.\left.x^{\prime} \eta\right)\right\rangle$ have
$\left(\right.$ subst $\left.x^{\prime \prime} \eta\right)=x \vee\left(x,\left(\right.\right.$ subst $\left.\left.x^{\prime \prime} \eta\right)\right) \in$ trm－ord
using trm－ord－subst by metis
from this and $\left\langle\left(\right.\right.$ subst $\left.x^{\prime \prime} \eta\right)=v \vee\left(\left(\left(\right.\right.\right.$ subst $\left.\left.x^{\prime \prime} \eta\right), v\right) \in$ trm－ord $\left.)\right\rangle$
have $x=v \vee((x, v) \in$ trm－ord $)$ using trm－ord－trans trans－def by metis
then show False
proof
assume $x=v$
from this and $\langle v \notin$ subst－set（trms－ecl C2）$\eta\rangle$
$\langle x \in($ subst－set（trms－ecl C2）$\eta$ ）$\rangle$ show False by auto
next
assume $(x, v) \in$ trm－ord
from 〈occurs－in $y$ x have $y=x \vee(y, x) \in$ trm－ord unfolding occurs－in－def using subterm－trm－ord－eq by auto
from this and $\langle(x, v) \in$ trm－ord $\rangle$ have $(y, v) \in$ trm－ord using trm－ord－trans unfolding trans－def by metis
from this and norm and $\langle$ trm－rep y $S \neq y\rangle$ and $\langle o c c u r s-i n$ y $x\rangle$ and $\left\langle x^{\prime} \in\right.$ trms－ecl C2＞
and $\left\langle x=\left(\right.\right.$ subst $\left.\left.x^{\prime} \eta\right)\right\rangle$ show False by metis
qed
qed
from $\langle$ subterm $t p u\rangle$ have $u=t \vee(u, t) \in$ trm－ord using subterm－trm－ord－eq by auto
from $\operatorname{assms}(8)$ and $\langle L 1 \in($ cl－ecl $C)\rangle$ have vars－of－lit（subst－lit L1 $\sigma)=\{ \}$
by auto
from 〈coincide－on $\sigma \sigma^{\prime}($ vars－of－lit L1）〉have（subst－lit L1 $\sigma$ ）$=($ subst－lit L1 $\sigma^{\prime}$ ）
using coincide－on－lit by metis
from this and 〈vars－of－lit（subst－lit L1 $\sigma$ ）$=\{ \}$ 〉
have vars－of－lit（subst－lit L1 $\sigma^{\prime}$ ）$=\{ \}$ by auto
from 〈coincide－on $\eta \sigma^{\prime}($ vars－of－lit L2）$\rangle$ have $($ subst－lit L2 $\eta)=($ subst－lit L2 $\sigma^{\prime}$ ）
using coincide－on－lit by metis
from 〈vars－of－lit gr－L2 $=\{ \}\rangle\langle g r o u n d-c l a u s e ~ g r-C l-C 2\rangle\langle g r-C l-C 2=($ subst－cl Cl－C2 ${ }^{2}$ ）＞
$\langle L 2 \in C l-C 2\rangle$ have vars－of－lit（subst－lit L2 $\eta$ ）$=\{ \}$ by auto
from this and 〈（subst－lit L2 $\eta$ ）$=\left(\right.$ subst－lit L2 $\left.\sigma^{\prime}\right)$ 〉
have vars－of－lit（subst－lit L2 $\sigma^{\prime}$ ）$=\{ \}$ by auto
We now prove that the＂into＂clause is strictly smaller than the＂from＂ clause．This is easy if the rewritten literal is negative or if the reduction does not occur at root level．Otherwise，we must use the fact that the function trm－rep selects the smallest right－hand side to compute the value of a term．

```
have (subst-lit L2 \(\sigma^{\prime}\), subst-lit L1 \(\sigma^{\prime}\) ) \(\in\) lit-ord
    proof (rule ccontr)
    assume \(\neg\left(\right.\) subst-lit L2 \(\sigma^{\prime}\), subst-lit L1 \(\left.\sigma^{\prime}\right) \in\) lit-ord
    from 〈orient-lit-inst L1 \(t\) s polarity \(\sigma^{\prime}\) 〉
        have orient-lit (subst-lit L1 \(\sigma^{\prime}\) )
            (subst \(t \sigma^{\prime}\) ) (subst \(s \sigma^{\prime}\) ) polarity
        using lift-orient-lit [of L1 ts polarity \(\sigma^{\prime}\) ] by auto
    from <orient-lit-inst L2 lhs rhs pos \(\sigma^{\prime}\) 〉
        have orient-lit (subst-lit L2 \(\sigma^{\prime}\) )
            (subst lhs \(\sigma^{\prime}\) ) (subst rhs \(\sigma^{\prime}\) ) pos
        using lift-orient-lit by auto
    have \((u, t) \notin\) trm-ord
    proof
    assume \((u, t) \in\) trm-ord
    from this have (subst \(u \sigma^{\prime}\), subst \(\left.t \sigma^{\prime}\right) \in\) trm-ord
        using trm-ord-subst by auto
    have False subst \(u \sigma^{\prime}=\) subst lhs \(\sigma^{\prime}\)
        using 〈(subst \(u \sigma^{\prime}\), subst \(\left.t \sigma^{\prime}\right) \in\) trm-ord〉
        \(\left\langle\left(\right.\right.\) subst-lit L2 \(\sigma^{\prime}\), subst-lit L1 \(\left.\sigma^{\prime}\right) \notin\) lit-ord〉〈subst lhs \(\eta=\) subst lhs \(\left.\sigma^{\prime}\right\rangle\)
        〈subst \(u \sigma=\) subst \(\left.u \sigma^{\prime}\right\rangle\left\langle\right.\) subst-lit L2 \(\eta=\) subst-lit L2 \(\left.\sigma^{\prime}\right\rangle\langle g r-L 2=\) subst-lit
\(L 2\) 7>
            〈orient-lit (subst-lit L1 \(\sigma^{\prime}\) ) (subst \(\left.t \sigma^{\prime}\right)\left(\right.\) subst \(\left.s \sigma^{\prime}\right)\) polarity>
            〈orient-lit (subst-lit L2 \(\sigma^{\prime}\) ) (subst lhs \(\sigma^{\prime}\) ) (subst rhs \(\left.\sigma^{\prime}\right)\) pos〉 \(\langle v=\) subst lhs
\(\eta>\)
        \(\langle v=\) subst \(u \sigma\rangle\left\langle\right.\) vars-of-lit (subst-lit L1 \(\sigma^{\prime}\) ) \(\left.=\{ \}\right\rangle\langle\) vars-of-lit gr-L2 \(=\{ \}\rangle\)
        lit-ord-dominating-term apply fastforce
        using «(subst \(u \sigma^{\prime}\), subst \(\left.t \sigma^{\prime}\right) \in\) trm-ord \(\rangle\)
        \(\left\langle\left(\right.\right.\) subst-lit L2 \(\sigma^{\prime}\), subst-lit L1 \(\left.\sigma^{\prime}\right) \notin\) lit-ord \(\rangle\)
        «subst \(u \sigma^{\prime}=\) subst lhs \(\sigma^{\prime}\) 〉
        〈orient-lit (subst-lit L1 \(\sigma^{\prime}\) ) (subst \(\left.t \sigma^{\prime}\right)\left(\right.\) subst \(\left.s \sigma^{\prime}\right)\) polarity〉
        <orient-lit (subst-lit L2 \(\sigma^{\prime}\) ) (subst lhs \(\sigma^{\prime}\) ) (subst rhs \(\sigma^{\prime}\) ) pos〉
        \(\left\langle\right.\) vars-of-lit (subst-lit L1 \(\sigma^{\prime}\) ) \(\left.=\{ \}\right\rangle\left\langle\right.\) vars-of-lit (subst-lit L2 \(\left.\left.\sigma^{\prime}\right)=\{ \}\right\rangle\)
        lit-ord-dominating-term by fastforce
    then show False by auto
    qed
    from this and «subterm \(t p u\) have \(p=\) Nil using subterm-trm-ord by auto
```

```
have }\neg\mathrm{ proper-subterm-red t S }
proof
    assume proper-subterm-red t S \sigma
    from this obtain p's}\mathrm{ where }\mp@subsup{p}{}{\prime}\not=Nil and subterm t p'
        trm-rep (subst s \sigma) S}\not=(\mathrm{ subst s }\sigma
        unfolding proper-subterm-red-def by blast
    from }\langlep=Nil\rangle and \langle\mp@subsup{p}{}{\prime}\not=Nil\rangle have ( p',p)\in(pos-ord Ct
        using pos-ord-nil by auto
    from <subterm t p' s` have subterm (subst t \sigma) p' (subst s \sigma)
        by (simp add: substs-preserve-subterms)
    from this and }\langle(\mp@subsup{p}{}{\prime},p)\in(\mathrm{ pos-ord C t)> mr and <trm-rep (subst s }\sigma)S\not
(subst s \sigma)\rangle\langlemp=p\rangle
    show False using minimal-redex-def by blast
    qed
```

    from \(\langle(u, t) \notin\) trm-ord \(\rangle\) and \(\langle u=t \vee(u, t) \in\) trm-ord \(\rangle\) have \(u=t\) by auto
    have polarity \(=\) pos
    proof (rule ccontr)
    assume polarity \(\neq\) pos
    then have polarity \(=\) neg using sign.exhaust by auto
    from \(\langle u=t\rangle\) have subst \(t \sigma^{\prime}=\) subst \(u \sigma^{\prime}\) by auto
    from this and \(\langle v=(\) subst \(u \sigma)\rangle\) and \(\langle v=(\) subst lhs \(\eta)\rangle\)
            and 〈subst lhs \(\eta=\) subst lhs \(\left.\sigma^{\prime}\right\rangle\)
            and \(\left\langle(\right.\) subst \(u \sigma)=\left(\right.\) subst \(\left.\left.u \sigma^{\prime}\right)\right\rangle\)
            have (subst \(\left.t \sigma^{\prime}\right)=\left(\right.\) subst lhs \(\left.\sigma^{\prime}\right)\) by auto
    from this and \(\left\langle\right.\) polarity \(=\) neg〉 〈orient-lit (subst-lit L1 \(\sigma^{\prime}\) )
            (subst \(\left.t \sigma^{\prime}\right)\left(\right.\) subst \(\left.s \sigma^{\prime}\right)\) polarity>
            and <orient-lit (subst-lit L2 \(\sigma^{\prime}\) )
            (subst lhs \(\sigma^{\prime}\) ) (subst rhs \(\sigma^{\prime}\) ) pos>
            〈(subst-lit L2 \(\sigma^{\prime}\), subst-lit L1 \(\left.\sigma^{\prime}\right) \notin\) lit-ord \(\rangle\)
            \(\left\langle\right.\) vars-of-lit (subst-lit L1 \(\sigma^{\prime}\) ) \(\left.=\{ \}\right\rangle\)
            \(\left\langle\right.\) vars-of-lit (subst-lit L2 \(\sigma^{\prime}\) ) \(\left.=\{ \}\right\rangle\)
        show False using lit-ord-neg-lit-lhs by auto
    qed
    from 〈vars-of-lit (subst-lit L1 \(\sigma\) ) \(=\{ \}\rangle \operatorname{assms}(6)\)
        have vars-of \((\) subst \(t \sigma)=\{ \}\) using lift-orient-lit orient-lit-vars
        by blast
    from 〈vars-of-lit (subst-lit L1 \(\sigma\) ) \(=\{ \}\) 〉assms( 6 )
        have vars-of (subst s \(\sigma\) ) \(=\{ \}\) using lift-orient-lit orient-lit-vars
        by blast
    have trm-rep (subst t \(\sigma\) ) \(S \neq\) trm-rep (subst s \(\sigma\) ) \(S\)
    proof
    assume trm-rep (subst \(t \sigma\) ) \(S=\) trm-rep (subst s \(\sigma\) ) \(S\)
    from this have validate-ground-eq ?I ( \(E q\) (subst \(t \sigma\) ) (subst s \(\sigma\) ))
            unfolding same-values-def using validate-ground-eq.simps by (metis
    (mono-tags, lifting))
from 〈trm-rep (subst t $\sigma$ ) $S=$ trm-rep (subst s $\sigma$ ) $S\rangle$
have validate－ground－eq ？I（Eq（subst s $\sigma$ ）（subst $t \sigma)$ ）
unfolding same－values－def using validate－ground－eq．simps by（metis （mono－tags，lifting））
from 〈orient－lit－inst L1 t s polarity $\sigma\rangle$ and $\langle p o l a r i t y=p o s 〉$
have $L 1=(\operatorname{Pos}(E q t s)) \vee L 1=(\operatorname{Pos}(E q s t))$
unfolding orient－lit－inst－def by auto
from this have subst－lit L1 $\sigma=(\operatorname{Pos}(E q($ subst $t \sigma)($ subst $s \sigma))) \vee$ subst－lit L1 $\sigma=($ Pos $(E q($ subst $s \sigma)($ subst $t \sigma)))$ by auto
from this and «validate－ground－eq ？I（Eq（subst s $\sigma$ ）（subst $t \sigma$ ））＞ and «validate－ground－eq？？$(E q$（subst $t \sigma)($ subst $s \sigma))$ 〉
have validate－ground－lit ？I（subst－lit L1 $\sigma$ ）using validate－ground－lit．simps（1） by metis
from $\langle L 1 \in($ cl－ecl $C)\rangle$ have $($ subst－lit L1 $\sigma) \in($ subst－cl $($ cl－ecl $C) \sigma)$ by auto
from this and «validate－ground－lit ？I（subst－lit L1 $\sigma$ ）＞
have validate－ground－clause ？I（subst－cl（cl－ecl C）$\sigma$ ）
using validate－ground－clause．simps by metis
from this and $\langle\neg$ validate－ground－clause（int－clset $S$ ）（subst－cl（cl－ecl C）$\sigma$ ）〉 show False unfolding int－clset－def by blast
qed
have $c v^{\prime}$ ：（candidate－values（trm－rep（subst s $\sigma$ ）$S$ ）$C$（cl－ecl C）（subst－cl（cl－ecl C）$\sigma$ ）
$($ subst s $\sigma)($ subst－lit L1 $\sigma) L 1 \sigma t s v S)$
proof－
from 〈polarity＝pos〉 and 〈orient－lit－inst L1 tspolarity $\sigma\rangle$ have $\neg$ nega－ tive－literal L1
unfolding orient－lit－inst－def by auto
from this and «eligible－literal L1 $C \sigma$ 〉
have sel $($ cl－ecl $C)=\{ \}$ and maximal－literal（subst－lit L1 $\sigma$ ）（subst－cl （cl－ecl C）$\sigma$ ）
using sel－neg unfolding eligible－literal－def by auto
from $\langle v=$ subst $u \sigma\rangle$ and $\langle u=t\rangle$ have $v=$ subst $t \sigma$ by auto
from $\operatorname{assms}(7)\langle C \in S\rangle$ have finite（ cl－ecl $C$ ）by auto
have $v \notin$ subst－set（trms－ecl C）$\sigma$
proof
assume $v \in$ subst－set（trms－ecl C）$\sigma$
from this and assms（12） （subterm（subst $t \sigma$ ）p v〉〈v＝subst $t \sigma\rangle$
have trm－rep vS $=v$ unfolding all－trms－irreducible－def occurs－in－def
by blast
from this $\langle v=$ subst $t \sigma\rangle\langle$ trm－rep（subst $t \sigma) S \neq($ subst $t \sigma)\rangle$ show False by blast
qed
from assms（13）have smaller－lits－are－false $v$（subst－cl（cl－ecl C）$\sigma$ ）$S$
using smaller－lits－are－false－if－cl－not－valid $[$ of $S($ subst－cl $(c l-e c l C) \sigma)]$ by blast
from $\operatorname{assms}(1)\langle\neg$ proper－subterm－red $t S \sigma\rangle\langle$ polarity $=$ pos $\langle\langle v=$ subst $t \sigma\rangle$ have
maximal－literal－is－unique $v($ subst $s \sigma)($ cl－ecl C）L1 $S \sigma$
using maximal－literal－is－unique－lemma［of $t$ s（cl－ecl C）$S \sigma$ L1］by blast from 〈all－trms－irreducible（subst－set（trms－ecl C）$\sigma$ ）（ $\lambda$ t．trm－rep $t S$ ）〉
have trms－irreducible $C \sigma S v$ using trms－irreducible－lemma［of $C \sigma S v$ ］ by blast
have（subst s $\sigma$ ，subst $t \sigma$ ）$\in$ trm－ord
proof－
from 〈orient－lit－inst L1 ts polarity $\sigma\rangle$ have（subst t $\sigma$ ，subst s $\sigma$ ）$\notin$ trm－ord unfolding orient－lit－inst－def by auto
from 〈trm－rep（subst $t \sigma$ ）$S \neq$ trm－rep（subst s $\sigma$ ）$S\rangle$
have（subst $t \sigma) \neq($ subst $s \sigma)$ by metis
from this and 〈（subst $t \sigma$ ，subst $s \sigma) \notin$ trm－ord $\rangle$
$\langle$ vars－of（subst $t \sigma)=\{ \}\rangle$
$\langle$ vars－of（subst s $\sigma$ ）$=\{ \}\rangle$
show（subst $s \sigma$ ，subst $t \sigma$ ）$\in$ trm－ord
using trm－ord－ground－total unfolding ground－term－def by metis
qed
from $\langle C \in S\rangle\langle($ subst $s \sigma$ ，subst $t \sigma) \in$ trm－ord $\rangle$
and $\langle$ polarity $=$ pos〉〈orient－lit－inst L1 tspolarity $\sigma\rangle$ and $\langle s e l(c l-e c l ~ C)=$
and $\langle L 1 \in$ cl－ecl $C\rangle$
and «maximal－literal（subst－lit L1 $\sigma$ ）（subst－cl（cl－ecl C）$\sigma$ ）〉
and $\langle$ ground－clause（subst－cl（cl－ecl $C$ ）$\sigma$ ）〉 and $\langle v=$ subst $t \sigma\rangle$
and 〈finite（cl－ecl C）〉
and $\langle v \notin$ subst－set（trms－ecl $C$ ）$\sigma\rangle$
and 〈smaller－lits－are－false v（subst－cl（cl－ecl C）$\sigma$ ）S〉
and 〈maximal－literal－is－unique $v($ subst $s \sigma)($ cl－ecl $C) L 1 S \sigma\rangle$
and 〈trms－irreducible $C \sigma S$ v〉
show $c v^{\prime}$ ：（candidate－values（trm－rep（subst s $\sigma$ ）$S$ ）$C$（cl－ecl C）（subst－cl （cl－ecl C）$\sigma$ ）
（subst s $\sigma$ ）（subst－lit L1 $\sigma$ ）L1 $\sigma t \operatorname{s} v$ S）
unfolding candidate－values－def by blast
qed
from $c v^{\prime}$ have $($ trm－rep $($ subst $s \sigma) S,($ subst $s \sigma)) \in$ set－of－candidate－values $S$
unfolding set－of－candidate－values－def by blast
from this and min－pair and $\langle$ pair $=(z z, \quad g r$－rhs $)\rangle$
have $(($ subst $s \sigma), g r$－rhs $) \notin$ trm－ord
by（metis snd－conv）
have（subst s $\sigma$ ）$\neq$ gr－rhs
using $\langle$ trm－rep $v S=$ trm－rep gr－rhs $S\rangle\langle u=t\rangle\langle v=$ subst $u \sigma\rangle$ $\langle$ trm－rep（subst $t \sigma$ ）$S \neq$ trm－rep（subst $s \sigma$ ）$S\rangle$ by blast
have vars－of gr－rhs $=\{ \}$
using 〈subst rhs $\eta=$ subst rhs $\sigma^{\prime}$ 〉
〈subst－lit L2 $\eta=$ subst－lit L2 $\sigma^{\prime}$ 〉
$\langle g r-L 2=$ subst－lit L2 $\eta\rangle\langle g r-r h s=$ subst rhs $\eta\rangle$


```
    <vars-of-lit gr-L2 = {}> orient-lit-vars by fastforce
    from〈(subst s \sigma)}\not=gr\mathrm{ -rhs〉 and <vars-of (subst s }\sigma)={}\rangle\langlevars-of gr-rhs
    〈((subst s \sigma),gr-rhs) & trm-ord`
    have (gr-rhs,(subst s \sigma)) \in trm-ord
    using trm-ord-ground-total unfolding ground-term-def by blast
    have (subst-lit L2 }\mp@subsup{\sigma}{}{\prime}\mathrm{ , subst-lit L1 }\mp@subsup{\sigma}{}{\prime}\mathrm{ ) }\in\mathrm{ lit-ord
    using <(gr-rhs, subst s \sigma) \in trm-ord`
        <subst lhs }\eta=\mathrm{ subst lhs }\mp@subsup{\sigma}{}{\prime}〉\langlesubst rhs \eta= subst rhs \sigma'〉
        <subst-lit L1 \sigma = subst-lit L1 \sigma}\mp@subsup{\sigma}{}{\prime
        <gr-rhs = subst rhs \eta
```



```
        <polarity = pos\rangle\langleu=t\rangle\langlev= subst lhs \eta\rangle\langlev=subst u \sigma\rangle
        <vars-of-lit (subst-lit L1 尔)={}>〈vars-of-lit (subst-lit L2 的)}={}
        assms(6) lit-ord-rhs lift-orient-lit by fastforce
    from this and «(subst-lit L2 }\mp@subsup{\sigma}{}{\prime}\mathrm{ , subst-lit L1 沙) #lit-ord〉 show False by auto
qed
have trm-rep (subst u \sigma) S\not=(subst u \sigma)
    using <trm-rep vS\not=v\rangle\langlev= subst u \sigma\rangle by blast
have allowed-redex u C \sigma
proof (rule ccontr)
    assume \negallowed-redex и C \sigma
    from this obtain ss where ss \in trms-ecl C
    and occurs-in (subst u\sigma) (subst ss \sigma) unfolding allowed-redex-def by auto
from <ss \in trms-ecl C` have (subst ss \sigma) \in(subst-set (trms-ecl C) \sigma) by auto
from this and assms(12) and {occurs-in (subst u \sigma) (subst ss \sigma)>
    <rm-rep (subst u \sigma)S\not=(subst u \sigma)〉
    show False
    unfolding all-trms-irreducible-def by blast
qed
have subst lhs 的}\not=\mathrm{ subst rhs }\mp@subsup{\sigma}{}{\prime
    using «(gr-rhs,v) \in trm-ord`
    <subst lhs \eta = subst lhs 生`
    <subst rhs }\eta=\mathrm{ subst rhs }\mp@subsup{\sigma}{}{\prime}\mathrm{ 〉
    <gr-rhs = subst rhs \eta\rangle\langlev = subst lhs \eta\rangle trm-ord-wf by auto
from this <mp=p\rangle\langle\neg(is-a-variable u)\rangle
<all-trms-irreducible (subst-set (trms-ecl C2) }\mp@subsup{\sigma}{}{\prime})(\lambdat.trm-rep t S)>
<(subst-lit L2 \sigma', subst-lit L1 }\mp@subsup{\sigma}{}{\prime})\inlit-ord>
<all-trms-irreducible (subst-set (trms-ecl C2) \sigma') (\lambdat. trm-rep t S)>
<(\forallx\incl-ecl C2 - {L2}.(subst-lit x 尔, subst-lit L2 \sigma') \in lit-ord)>
\langleC2 G S\rangle\langleeligible-literal L1 C \sigma'> <eligible-literal L2 C2 \sigma
<ground-clause (subst-cl Cl-C2 \sigma')\rangle\langleCl-C2 = cl-ecl C2 >
    mr <coincide-on \sigma \sigma'(vars-of-cl (cl-ecl C))\rangle\langleL1 \incl-ecl C\rangle\langleL2 \inCl-C2\rangle
    <orient-lit-inst L1 t s polarity }\mp@subsup{\sigma}{}{\prime}\rangle\langle(orient-lit-inst L2 lhs rhs pos \mp@subsup{\sigma}{}{\prime})
```



```
    <trm-rep (subst rhs 的)S= trm-rep (subst lhs 海)S〉
```

\{\}>
$\left\langle\left(\neg\right.\right.$ validate－ground－clause ？I（subst－cl（Cl－C2－\｛L2 \} ) $\left.\left.\left.\sigma^{\prime}\right)\right)\right\rangle$
〈allowed－redex u $C \sigma$ 〉
have（reduction L1 $C \sigma^{\prime} t$ s polarity L2 lhs u mp rhs C2（same－values（ $\lambda t$ ． （trm－rep $t S)$ ））$S \sigma$ ）
unfolding reduction－def same－values－def
by metis
from $\langle$ vars－of－cl（cl－ecl $C) \cap$ vars－of－cl（cl－ecl C2）$=\{ \}\rangle$ have variable－disjoint C C2
unfolding variable－disjoint－def by auto
from this and
〈（reduction L1 C $\sigma^{\prime} t$ s polarity L2 lhs u mp rhs C2（same－values（ $\lambda t$ ．（trm－rep $t S)$ ））$S \sigma$ ）＞
show ？thesis by blast
qed
lemma subts－of－irred－trms－are－irred：
assumes trm－rep y $S \neq y$
shows $\wedge x$ ．subterm x p $y \longrightarrow$ trm－rep $x S \neq x$
proof（induction p）
case（Nil）
from $\operatorname{assms}(1)$ show ？case by（metis subterm．simps（1））
next case（Cons i $p$ ）
show $\wedge x$ ．subterm $x($ Cons $i p) y \longrightarrow \operatorname{trm}$－rep $x S \neq x$
proof
fix $x$ assume subterm $x$（Cons ip）$y$
from this obtain $x 1$ x2 where $x=$ Comb $x 1$ x2 using subterm．elims（2）by blast
have $i=$ Left $\mid i=$ Right using indices．exhaust by auto
then show trm－rep $x S \neq x$
proof
assume $i=$ Left
from this and $\langle$ subterm $x$（Cons ip）$y\rangle\langle x=$ Comb $x 1$ x2 $\rangle$ have subterm
$x 1 p y$ by auto
from this and Cons．IH have trm－rep $x 1 S \neq x 1$ by blast
from this and $\langle x=C o m b x 1$ x2〉 have subterm－reduction－applicable $S x$
unfolding subterm－reduction－applicable－def
by（metis is－compound．simps（3）lhs．simps（1））
from this have $($ trm－rep $x S, x) \in$ trm－ord using trm－rep－is－lower－subt－red by blast
from this show ？thesis using trm－ord－irrefl unfolding irrefl－def by metis next
assume $i=$ Right
from this and $\langle$ subterm $x$（Cons ip）$y\rangle\langle x=$ Comb $x 1$ x2 $\rangle$ have subterm
$x 2 p y$ by auto
from this and Cons．IH have trm－rep $x 2 S \neq x 2$ by blast
from this and $\langle x=C o m b x 1$ x2〉 have subterm－reduction－applicable $S x$ unfolding subterm－reduction－applicable－def
by（metis is－compound．simps（3）rhs．simps（1））
from this have（trm－rep $x S, x) \in$ trm－ord using trm－rep－is－lower－subt－red
by blast
from this show ?thesis using trm-ord-irrefl unfolding irrefl-def by metis qed
qed
qed
lemma allowed-redex-coincide:
assumes allowed-redex $t C \sigma$
assumes $t \in$ subterms-of-cl (cl-ecl $C$ )
assumes coincide-on $\sigma \sigma^{\prime}$ (vars-of-cl (cl-ecl C))
assumes well-constrained $C$
shows allowed-redex $t C \sigma^{\prime}$
proof (rule ccontr)
assume $\neg$ allowed-redex $t C \sigma^{\prime}$
from this obtain $s$
where $s \in$ trms-ecl $C$ and occurs-in (subst $t \sigma^{\prime}$ ) (subst s $\sigma^{\prime}$ )
unfolding allowed-redex-def by auto
from $\langle s \in$ trms-ecl $C\rangle$ and assms(4) have vars-of $s \subseteq$ vars-of-cl (cl-ecl $C$ )
using dom-trm-vars unfolding well-constrained-def by blast
from this have vars-of $s \subseteq$ vars-of-cl (cl-ecl C) using subterm-vars by blast
from this and assms(3) have coincide-on $\sigma \sigma^{\prime}$ (vars-of $s$ ) unfolding coin-cide-on-def by auto
from this have (subst $s \sigma)=\left(\right.$ subst $\left.s \sigma^{\prime}\right)$ using coincide-on-term by auto
from $\operatorname{assms}(2)$ have vars-of $t \subseteq$ vars-of-cl (cl-ecl $C$ ) using subterm-vars by blast
from this and assms(3) have coincide-on $\sigma \sigma^{\prime}$ (vars-of $t$ ) unfolding coin-cide-on-def by auto
from this have (subst $t \sigma)=\left(\right.$ subst $\left.t \sigma^{\prime}\right)$ using coincide-on-term by auto
from this and $\left\langle(\right.$ subst $s \sigma)=\left(\right.$ subst $\left.\left.s \sigma^{\prime}\right)\right\rangle$ and 〈occurs-in (subst $\left.t \sigma^{\prime}\right)($ subst $s$ $\sigma^{\prime}$ ) >
have occurs-in (subst $t \sigma$ ) (subst $s \sigma$ ) by auto
from this and $\langle s \in$ trms-ecl $C\rangle$ have $\neg$ allowed-redex $t C \sigma$ unfolding al-lowed-redex-def by auto
from this and assms(1) show False by auto
qed
The next lemma states that the irreducibility of an instance of a set of terms is preserved when the substitution is replaced by its equivalent normal form.

```
lemma irred-terms-and-reduced-subst:
    assumes \(f=(\lambda t\). \((\) trm-rep \(t S))\)
    assumes \(\eta=(\) map-subst \(f \sigma)\)
    assumes all-trms-irreducible (subst-set E \(\sigma\) ) f
    assumes \(I=\) int-clset \(S\)
    assumes equivalent-on \(\sigma \eta\) (vars-of-cl (cl-ecl C)) I
    assumes lower-on \(\eta \sigma\) (vars-of-cl (cl-ecl \(C)\) )
    assumes \(E=(\) trms-ecl \(C)\)
    assumes \(\forall x \in S . \forall y .(y \in\) trms-ecl \(x \longrightarrow\) dom-trm \(y(\) cl-ecl \(x))\)
    assumes \(C \in S\)
    assumes fo-interpretation I
```

```
    shows all-trms-irreducible (subst-set E \eta )f
proof (rule ccontr)
    assume \negall-trms-irreducible (subst-set E \eta) f
    from this obtain ty where y (subst-set E \eta) occurs-in t yft\not=t
        unfolding all-trms-irreducible-def by metis
    from <occurs-in t y` obtain p where subterm y pt unfolding occurs-in-def by
auto
    from this and }\langleft\not=t\rangle\mathrm{ and assms(1) have fy}=y\mathrm{ using subts-of-irred-trms-are-irred
by blast
    from }\langley\in(\mathrm{ subst-set E }\eta)\rangle\mathrm{ obtain z where z E E and y=(subst z 
        by auto
    from }\langlez\inE\rangle\mathrm{ have (subst z %) ( (subst-set E %) by auto
    have subterm (subst z \sigma) [] (subst z \sigma) by auto
    then have occurs-in (subst z\sigma) (subst z \sigma) unfolding occurs-in-def
        by blast
    from this and assms(3) and <(subst z \sigma) \in(subst-set E \sigma)\rangle
        have f (subst z \sigma) = (subst z \sigma)
        unfolding all-trms-irreducible-def by metis
    from this and \langlef y\not=y\rangle and \langley=( subst z \eta)\rangle
        have (subst z \sigma)}\not=(\mathrm{ subst z }\eta\mathrm{ ) by metis
    from }\langlez\inE\rangle\mathrm{ and assms(7) assms(8) assms(9) have dom-trmz (cl-ecl C) by
metis
    from this have vars-of z\subseteqvars-of-cl (cl-ecl C) using dom-trm-vars by auto
    from this assms(5) have equivalent-on \sigma \eta (vars-of z) I
        unfolding equivalent-on-def by auto
    from <vars-of z\subseteqvars-of-cl (cl-ecl C)> assms(6) have lower-on \eta \sigma (vars-of z)
        unfolding lower-on-def by auto
    from <(subst z \sigma) \not=(subst z \eta)>
                <lower-on \eta\sigma (vars-of z)>
            have ((subst z \eta),(subst z\sigma)) \in trm-ord
            using lower-on-term unfolding lower-or-eq-def by metis
    from this have ( (subst z \sigma),(subst z \eta)) )\not trm-ord
            using trm-ord-trans trm-ord-irrefl irrefl-def trans-def by metis
    from assms(10)<equivalent-on \sigma \eta (vars-of z) I`
            have (I (subst z \sigma) (subst z \eta)) using equivalent-on-term
                unfolding fo-interpretation-def by auto
    from this and assms(4) assms(1)<f (subst z \sigma)=(subst z \sigma)\rangle
        have (subst z \sigma) =f (subst z \eta) unfolding same-values-def int-clset-def
        by metis
    from this «( (subst z \sigma),(subst z \eta)) # trm-ord`
            <(subst z \sigma)\not=(subst z \eta)> assms(1)
            show False using trm-rep-is-lower by metis
qed
lemma no-valid-literal:
    assumes L\inC
    assumes orient-lit-inst Lt s pos \sigma
    assumes }\neg(validate-ground-clause (int-clset S) (subst-cl C \sigma))
```

```
    shows trm-rep (subst t \sigma) S\not= trm-rep (subst s \sigma) S
proof
    assume neg-hyp: trm-rep (subst t \sigma) S= trm-rep (subst s \sigma) S
    let ?I = int-clset S
    from neg-hyp have validate-ground-eq ?I (Eq (subst t \sigma) (subst s \sigma))
        unfolding same-values-def int-clset-def using validate-ground-eq.simps
        by (metis (mono-tags, lifting))
    from <trm-rep (subst t \sigma)S= trm-rep (subst s \sigma) S>
        have validate-ground-eq ?I (Eq (subst s \sigma) (subst t \sigma))
        unfolding same-values-def int-clset-def using validate-ground-eq.simps
        by (metis (mono-tags, lifting))
    from<orient-lit-inst L t s pos \sigma〉 have L=(Pos (Eq t s))}\vee\mp@code{L=(Pos (Eq s
t))
            unfolding orient-lit-inst-def by auto
    from this have subst-lit L \sigma = (Pos (Eq (subst t \sigma) (subst s \sigma))) \vee
                subst-lit L \sigma =(Pos (Eq(subst s \sigma) (subst t \sigma))) by auto
    from this and <validate-ground-eq ?I (Eq (subst s \sigma) (subst t \sigma))>
        and <validate-ground-eq ?I (Eq (subst t \sigma) (subst s \sigma))>
    have validate-ground-lit ?I (subst-lit L \sigma) using validate-ground-lit.simps(1)
by metis
    from assms(1) have (subst-lit L \sigma) \in(subst-cl C \sigma) by auto
    from <(subst-lit L \sigma) \in(subst-cl C \sigma)〉
        and <validate-ground-lit ?I (subst-lit L \sigma)>
        have validate-ground-clause ?I (subst-cl C \sigma)
        using validate-ground-clause.elims(3) by blast
    from this and «\neg validate-ground-clause?I (subst-cl C \sigma)` show False by
blast
qed
```


### 7.2 Lifting

This section contains all the necessary lemmata for transforming ground inferences into first-order inferences. We show that all the necessary properties can be lifted.

```
lemma lift-orient-lit-inst:
    assumes orient-lit-inst L t s polarity \vartheta
    assumes subst-eq \vartheta (comp \sigma \eta)
    shows orient-lit-inst L t s polarity }
proof -
    let ?\vartheta = (comp \sigma \eta)
    have polarity = pos \vee polarity = neg using sign.exhaust by auto
    then show ?thesis
    proof
        assume polarity = pos
        from this and assms(1) have L=Pos (Eqts)\veeL=Pos (Eqst)
            and ((subst t \vartheta), (subst s \vartheta)) & trm-ord
            unfolding orient-lit-inst-def by auto
    from assms(2) have (subst t \vartheta)=(subst (subst t \sigma) \eta)
                by auto
```

from assms(2) have (subst s $\vartheta)=($ subst $($ subst $s \sigma) \eta)$
by auto
from 〈(subst t $\vartheta)=($ subst $($ subst $t \sigma) \eta)\rangle$
$\langle($ subst s $\vartheta)=($ subst $($ subst $s \sigma) \eta)\rangle$
$\langle(($ subst $t \vartheta), \quad($ subst s $\vartheta)) \notin$ trm-ord $\rangle$
have $(($ subst $($ subst $t \sigma) \eta),($ subst $($ subst s $\sigma) \eta)) \notin$ trm-ord
by auto
from this have $(($ subst $t \sigma),($ subst $s \sigma)) \notin$ trm-ord
using trm-ord-subst by auto
from this and $\langle$ polarity $=$ pos $\langle L=\operatorname{Pos}(E q t s) \vee L=\operatorname{Pos}(E q s t)\rangle$ show ?thesis
unfolding orient-lit-inst-def by blast
next
assume polarity $=$ neg
from this and assms(1) have $L=\operatorname{Neg}(E q t s) \vee L=N e g(E q s t)$
and $(($ subst $t \vartheta),($ subst s $\vartheta)) \notin$ trm-ord
unfolding orient-lit-inst-def by auto
from $\operatorname{assms}(2)$ have $($ subst $t \vartheta)=($ subst (subst $t \sigma) \eta)$
by auto
from $\operatorname{assms}(2)$ have $($ subst s $\vartheta)=($ subst $($ subst s $\sigma) \eta)$
by auto
from 〈(subst $t \vartheta)=($ subst $($ subst $t \sigma) \eta)\rangle$
$\langle($ subst s $\vartheta)=($ subst $($ subst s $\sigma) \eta)\rangle$
$\langle($ (subst t $\vartheta), \quad($ subst s $\vartheta)) \notin$ trm-ord $\rangle$
have $(($ subst $($ subst $t \sigma) \eta),($ subst $($ subst $s \sigma) \eta)) \notin$ trm-ord
by auto
from this have ( (subst $t \sigma),($ subst $s \sigma)) \notin$ trm-ord
using trm-ord-subst by auto
from this and $\langle$ polarity $=n e g\rangle\langle L=N e g(E q t s) \vee L=N e g(E q s t)\rangle$ show
?thesis
unfolding orient-lit-inst-def by blast
qed
qed
lemma lift-maximal-literal:
assumes maximal-literal (subst-lit L $\sigma$ ) (subst-cl C $\sigma$ )
shows maximal-literal L C
proof (rule ccontr)
assume $\neg$ maximal-literal $L C$
then obtain $M$ where $M \in C$ and $(L, M) \in$ lit-ord unfolding maximal-literal-def by auto
from $\langle M \in C\rangle$ have (subst-lit $M \sigma) \in($ subst-cl $C \sigma)$ by auto
from $\langle(L, M) \in$ lit-ord $\rangle$ have $(($ subst-lit $L \sigma),($ subst-lit $M \sigma)) \in$ lit-ord
using lit-ord-subst by auto
from this and $\langle(s u b s t-l i t M \sigma) \in($ subst-cl $C \sigma)\rangle$ and $\operatorname{assms}(1)$
show False unfolding maximal-literal-def by auto
qed
lemma lift-eligible-literal:

```
    assumes eligible-literal L C \sigma
    assumes }\sigma\doteq\vartheta\diamond
    shows eligible-literal L C \vartheta
proof -
    from assms(1) have (L\in sel (cl-ecl C) \vee
        (sel(cl-ecl C) = {}
        \wedge(maximal-literal (subst-lit L \sigma) (subst-cl (cl-ecl C) \sigma))))
        unfolding eligible-literal-def by auto
    then show ?thesis
    proof
        assume L E sel (cl-ecl C)
        then show ?thesis unfolding eligible-literal-def by auto
    next
        assume sel(cl-ecl C)={}
        \wedge(maximal-literal (subst-lit L \sigma) (subst-cl (cl-ecl C) \sigma))
        then have sel (cl-ecl C) ={} and maximal-literal (subst-lit L \sigma) (subst-cl
(cl-ecl C) \sigma)
            by auto
    let ? }\sigma=\vartheta\diamond
    from assms(2) have (subst-lit L \sigma)=(subst-lit L ? \sigma)
        using subst-eq-lit by auto
    then have (subst-lit L \sigma) =(subst-lit (subst-lit L \vartheta) \eta)
        using composition-of-substs-lit [of L \vartheta \eta] by auto
    from assms(2) have (subst-cl (cl-ecl C) \sigma)=(subst-cl (cl-ecl C) ?\sigma)
        using subst-eq-cl [of \sigma ?\sigma (cl-ecl C)] by auto
    then have (subst-cl (cl-ecl C) \sigma) =(subst-cl (subst-cl (cl-ecl C) \vartheta) \eta)
            using composition-of-substs-cl [of cl-ecl C \vartheta \eta ] by auto
    from <maximal-literal (subst-lit L \sigma) (subst-cl (cl-ecl C) \sigma)>
            〈(subst-lit L \sigma) = (subst-lit (subst-lit L \vartheta) \eta)>
            <(subst-cl (cl-ecl C) \sigma) = (subst-cl (subst-cl (cl-ecl C) \vartheta) \eta)>
            have maximal-literal (subst-lit (subst-lit L \vartheta) \eta)
                (subst-cl (subst-cl (cl-ecl C) \vartheta) \eta) by auto
    from this have maximal-literal (subst-lit L \vartheta) (subst-cl (cl-ecl C) \vartheta)
            using lift-maximal-literal by metis
    from this and «sel (cl-ecl C) ={}`show ?thesis unfolding eligible-literal-def
by auto
    qed
qed
lemma lift-allowed-redex:
    assumes }\sigma\doteq\vartheta\diamond
    assumes (allowed-redex u C \sigma)
    shows (allowed-redex u C \vartheta)
proof (rule ccontr)
    assume }\neg(\mathrm{ allowed-redex и C ७)
    from this obtain s}\mathrm{ where }s\in(\mathrm{ trms-ecl C) and (occurs-in (subst u Э) (subst s
\vartheta))
```

unfolding allowed－redex－def by metis
from 〈（occurs－in（subst u $\vartheta)($ subst s $\vartheta)$ ）〉
have（occurs－in（subst（subst u ७）$\eta$ ）（subst（subst s $)$ ）$\eta$ ））
using substs－preserve－occurs－in by auto
from $\langle\sigma \doteq \vartheta \diamond \eta\rangle$ have（subst $u \sigma)=($ subst（subst $u \vartheta) \eta$ ）by auto
from $\langle\sigma \doteq \vartheta \diamond \eta\rangle$ have（subst $s \sigma)=($ subst（subst s $\vartheta) ~ \eta)$ by auto
from 〈（occurs－in（subst（subst u $\vartheta) \eta$ ）（subst（subst s $\vartheta) \eta$ ））〉
$\langle($ subst $u \sigma)=($ subst $($ subst $u \vartheta) \eta)\rangle$
$\langle($ subst $s \sigma)=($ subst $($ subst s $\vartheta) \eta)\rangle$
have（occurs－in（subst $u \sigma$ ）（subst $s \sigma$ ））by auto
from this and $\langle s \in($ trms－ecl $C)\rangle$ assms（2）show False unfolding allowed－redex－def
by auto
qed
lemma lift－decompose－literal：
assumes decompose－literal（subst－lit $L \sigma$ ）ts polarity
assumes subst－eq $\vartheta(c o m p \sigma \eta)$
shows decompose－literal（subst－lit L $\vartheta$ ）（subst t $\eta$ ）（subst s $\eta$ ）polarity proof－
let $? L=($ subst－lit $L \sigma)$
let $? t^{\prime}=($ subst $t \eta)$
let $? s^{\prime}=($ subst $s \eta)$
let $? \vartheta=(\operatorname{comp} \sigma \eta)$
let $? L^{\prime}=($ subst－lit $? L \eta)$
from $\operatorname{assms}(2)$ have（subst－lit $L \vartheta)=($ subst－lit $L$ ？$\vartheta)$ using subst－eq－lit by auto
from this have（subst－lit $L \vartheta$ ）$=$ ？$L^{\prime}$ using composition－of－substs－lit by metis
have polarity $=$ pos $\vee$ polarity $=$ neg using sign．exhaust by auto
then show ？thesis
proof
assume polarity $=$ pos
from this and $\operatorname{assms}(1)$ have $? L=\operatorname{Pos}(E q t s) \vee ? L=\operatorname{Pos}(E q s t)$
unfolding decompose－literal－def decompose－equation－def by auto
from 〈？$L=\operatorname{Pos}(E q t s) \vee ? L=\operatorname{Pos}(E q s t)\rangle$ have ？$L^{\prime}=\operatorname{Pos}\left(E q ? t^{\prime} ? s^{\prime}\right) \vee ? L^{\prime}=\operatorname{Pos}\left(E q ? s^{\prime} ? t^{\prime}\right)$ by auto
from this $\left\langle(\right.$ subst－lit $L \vartheta)=$ ？$\left.L^{\prime}\right\rangle$
have $($ subst－lit $L \vartheta)=\operatorname{Pos}\left(E q ? t^{\prime} ? s^{\prime}\right) \vee($ subst－lit L $\vartheta)=\operatorname{Pos}(E q$ ？s＇？t＇$)$
by auto
from this $\langle$ polarity $=$ pos〉 show ？thesis unfolding decompose－literal－def decompose－equation－def by auto
next
assume polarity $=$ neg
from this and $\operatorname{assms}(1)$ have $? L=\operatorname{Neg}(E q t s) \vee ? L=N e g(E q s t)$
unfolding decompose－literal－def decompose－equation－def by auto
from $\langle ? L=N e g(E q t s) \vee ? L=N e g(E q s t)\rangle$

```
            have ? L' }=Neg(Eq?\mp@subsup{t}{}{\prime}?\mp@subsup{s}{}{\prime})\vee?\mp@subsup{L}{}{\prime}=Neg(Eq?\mp@subsup{s}{}{\prime}?\mp@subsup{t}{}{\prime})\mathrm{ by auto
    from this and «(subst-lit L \vartheta) =? 'L`
            have (subst-lit L \vartheta) =Neg(Eq?.t' ?s')\vee(subst-lit L \vartheta) = Neg (Eq?s' ?t')
by auto
    from this <polarity = neg〉 show ?thesis unfolding decompose-literal-def
            decompose-equation-def by auto
    qed
qed
lemma lift-dom-trm:
    assumes dom-trm (subst t \vartheta) (subst-cl C \vartheta)
    assumes }\sigma\doteq\vartheta\diamond
    shows dom-trm (subst t \sigma) (subst-cl C \sigma)
proof -
    let ?t = (subst t \vartheta)
    let ?t' = (subst ?t \eta)
    let ?t'\prime}=(\mathrm{ subst t }\sigma
    have ? }\mp@subsup{t}{}{\prime}=(\mathrm{ subst }t(\vartheta\diamond\eta))\mathrm{ by auto
    from assms(2) have ? }\mp@subsup{t}{}{\prime\prime}=(\mathrm{ subst }t(\vartheta\diamond\eta))\mathrm{ by auto
    from this and «? t' = (subst t (\vartheta\diamond\eta))> have ? t' = ?t'" by metis
    from assms(1) have (\existsLuvp. (L\in(subst-cl C \vartheta) ^(decompose-literal L uv
p)
            \wedge(( (p=neg ^ ?t = u)\vee (?t,u) \in trm-ord)))) unfolding dom-trm-def by
auto
    from this obtain Luvp where L \in(subst-cl C \vartheta)
        decompose-literal L uvp((( }p=neg\wedge?t=u)\vee(?t,u)\intrm-ord)
        unfolding dom-trm-def by blast
    from }\langleL\in(\mathrm{ subst-cl C v)> obtain }\mp@subsup{L}{}{\prime}\mathrm{ where }\mp@subsup{L}{}{\prime}\in
        L=(subst-lit L'\vartheta) by auto
    from this and <decompose-literal L uv p〉 have decompose-literal (subst-lit L' \vartheta)
uvp by auto
    from this assms(2) 〈L=(subst-lit L' \vartheta)>
            have decompose-literal (subst-lit L' }\sigma\mathrm{ ) (subst u }\eta\mathrm{ ) (subst v }\eta\mathrm{ ) p
            using lift-decompose-literal [of L' }\mp@subsup{L}{}{\prime}\varthetauvpol\eta] by aut
    let ?u = (subst u \eta)
    from }\langle\mp@subsup{L}{}{\prime}\inC\rangle\mathrm{ have (subst-lit L' }\mp@subsup{L}{}{\prime}\sigma)\in(\mathrm{ subst-cl C }\sigma\mathrm{ ) by auto
    from <((( p= neg ^ ?t =u)\vee (?t,u) \in trm-ord ))>
            have }(((p=neg\wedge?\mp@subsup{t}{}{\prime}=?u)\vee(?\mp@subsup{t}{}{\prime},?u)\in\mathrm{ trm-ord })
                using trm-ord-subst by auto
    from this and <? t' =? 't'> have ((( }p=neg\wedge?\mp@subsup{t}{}{\prime\prime}=?u)\vee(?\mp@subsup{t}{}{\prime\prime},?u)\in\mathrm{ trm-ord )}
by auto
    from this 〈(subst-lit L' \sigma) \in(subst-cl C \sigma)\rangle
        〈decompose-literal (subst-lit L'\sigma) (subst u \eta) (subst v \eta) p〉
        show dom-trm (subst t \sigma) (subst-cl C \sigma)
        unfolding dom-trm-def by auto
qed
lemma lift－irreducible－terms：
```

```
    assumes \(T=\) get-trms \(C\) (dom-trms (subst-cl D \(\sigma\) ) (subst-set \(E \sigma\) ) Ground
    assumes \(\sigma \doteq \vartheta \diamond \eta\)
    shows \(\exists T^{\prime}\). ( (subst-set \(\left.T^{\prime} \eta\right) \subseteq T \wedge T^{\prime}=\) get-trms \(C^{\prime}\)
    (dom-trms (subst-cl D \(\vartheta)\) (subst-set E \(\vartheta)\) ) FirstOrder)
proof -
    let \({ }^{2} E=(\) dom-trms \((\) subst-cl D \(\vartheta)(\) subst-set \(E\) ७) \()\)
    let \(? E^{\prime}=(\) dom-trms \((\) subst-cl \(D \sigma)(\) subst-set \(E \sigma))\)
    let \(? T^{\prime}=\left(\right.\) filter-trms \(\left.C^{\prime} ? E\right)\)
    have ? \(T^{\prime}=\) get-trms \(C^{\prime} ? E\) FirstOrder
    unfolding get-trms-def by auto
    from \(\operatorname{assms}(1)\) have \(T=\) ? \(E^{\prime}\) unfolding get-trms-def by auto
    have (subst-set ? \(T^{\prime} \eta\) ) \(\subseteq\) ? \(E^{\prime}\)
    proof
    fix \(x\) assume \(x \in\left(\right.\) subst-set ? \(\left.T^{\prime} \eta\right)\)
    from this obtain \(x^{\prime}\) where \(x=\left(\right.\) subst \(\left.x^{\prime} \eta\right)\) and \(x^{\prime} \in ? T^{\prime}\) by auto
    from \(\left\langle x^{\prime} \in ? T^{\prime}\right\rangle\) have \(x^{\prime} \in ?\) e using filter-trms-inclusion by auto
    from \(\left\langle x^{\prime} \in\right.\) ? \(\left.E\right\rangle\) have \(x^{\prime} \in(\) subst-set \(E \vartheta)\)
        and dom-trm \(x^{\prime}(\) subst-cl \(D \vartheta)\) unfolding dom-trms-def by auto
    from \(\left\langle x^{\prime} \in(\right.\) subst-set \(\left.E \vartheta)\right\rangle\) obtain \(y\) where \(y \in E\)
        and \(x^{\prime}=(\) subst \(y \vartheta)\) by auto
    from \(\left\langle x^{\prime}=(\right.\) subst \(\left.y \vartheta)\right\rangle\) and 〈dom-trm \(x^{\prime}(\) subst-cl \(D \vartheta)\) 〉
        have dom-trm (subst y \(\vartheta)(\) subst-cl \(D \vartheta)\) by auto
    from this assms(2)
        have dom-trm (subst y \(\sigma\) ) (subst-cl D \(\sigma\) )
        using lift-dom-trm by auto
    from \(\langle y \in E\rangle\) have (subst \(y \sigma) \in(\) subst-set \(E \sigma)\) by auto
    from this and \(\langle\) dom-trm (subst y \(\sigma\) ) (subst-cl D \(\sigma\) ) 〉
        have (subst \(y \sigma\) ) \(\in\) ? \(E^{\prime}\) unfolding dom-trms-def by auto
    from \(\operatorname{assms}(2)\) have (subst \(y \sigma)=(\) subst \(y(\vartheta \diamond \eta))\) by auto
    from this \(\left\langle x=\left(\right.\right.\) subst \(\left.\left.x^{\prime} \eta\right)\right\rangle\) and \(\left\langle x^{\prime}=(\right.\) subst \(\left.y \vartheta)\right\rangle\)
        have \(x=(\) subst \(y \sigma)\) by auto
    from this and \(«(\) subst \(\left.y \sigma) \in ? E^{\prime}\right\rangle\) show \(x \in ? E^{\prime}\) by auto
    qed
    from this and \(\left\langle T=? E^{\prime}\right\rangle\left\langle ? T^{\prime}=\right.\) get-trms \(C^{\prime} ? E\) FirstOrder〉 show ?thesis by
auto
qed
```

We eventually deduce the following lemmas，which allows one to transform ground derivations into first－order derivations．
lemma lifting－lemma－superposition：
assumes superposition P1 P2 C $\sigma$ Ground $C^{\prime}$
shows $\exists D \vartheta$ ．superposition P1 P2 $D \vartheta$ FirstOrder $C^{\prime} \wedge \sigma \doteq \vartheta \diamond \sigma \wedge$ trms－subsumes D $C \sigma$
proof（rule ccontr）
assume hyp：$\ddagger D \vartheta$ ．superposition P1 P2 $D \vartheta$ FirstOrder $C^{\prime} \wedge \sigma \doteq \vartheta \diamond \sigma \wedge$
trms－subsumes $D C \sigma$
have not－sup：$\neg$ superposition P1 P2 $C \sigma$ Ground $C^{\prime}$
proof（rule notI）
assume superposition P1 P2 $C \sigma$ Ground $C^{\prime}$
from this obtain $L$ tsuv MpCl－P1 Cl－P2 Cl－C polarity $t^{\prime} u^{\prime} L^{\prime}$ trms－$C$
where $L \in C l-P 1(M \in C l-P 2)$（eligible－literal LP1 $\sigma$ ）（eligible－literal M P2
$\sigma)$
（variable－disjoint P1 P2）
$($ Cl－P1 $=($ cl－ecl P1 $))($ Cl－P2 $=($ cl－ecl P2 $))$
（ $\neg$ is－a－variable $u^{\prime}$ ）
（allowed－redex u＇P1 $\sigma$ ）
（ $C=($ Ecl Cl－C trms－C）$)$
（orient－lit－inst Muvpos $\sigma$ ）
（orient－lit－inst Ltspolarity $\sigma$ ）
$(($ subst $u \sigma) \neq($ subst $v \sigma))$
（subterm t p $u^{\prime}$ ）
（ck－unifier $u^{\prime} u \sigma$ Ground）
（replace－subterm t p v t＇）
（ $L^{\prime}=m k$－lit polarity $\left.\left(E q t^{\prime} s\right)\right)$
（trms－C $=$ get－trms Cl － C （dom－trms Cl － C （subst－set
$(($ trms－ecl P1 $) \cup($ trms－ecl P2 $) \cup$
$\{r . \exists q .(q, p) \in($ pos－ord P1 t）$\wedge($ subterm $t q r)\}) \sigma))$ Ground $)$
$\left(C l-C=\left(\right.\right.$ subst－cl $\left.\left.\left((C l-P 1-\{L\}) \cup\left((C l-P 2-\{M\}) \cup\left\{L^{\prime}\right\}\right)\right) \sigma\right)\right)$
$\left(C^{\prime}=(C l-P 1-\{L\}) \cup\left((C l-P 2-\{M\}) \cup\left\{L^{\prime}\right\}\right)\right)$
unfolding superposition－def get－trms－def by auto
from $\left\langle\left(c k\right.\right.$－unifier $u^{\prime} u \sigma$ Ground $\left.)\right\rangle$ have Unifier $\sigma u^{\prime} u$ unfolding ck－unifier－def by auto
from this have（subst $\left.u^{\prime} \sigma\right)=($ subst $u \sigma)$ unfolding Unifier－def by auto
from this have unify $u^{\prime} u \neq$ None using $M G U$－exists by auto
from this obtain $\vartheta$ where unify $u^{\prime} u=S o m e \vartheta$ by auto
hence min－IMGU $\vartheta u^{\prime} u$ by（rule unify－computes－min－IMGU）
with «Unifier $\sigma u^{\prime} u$ have $\sigma \doteq \vartheta \diamond \sigma$
unfolding min－IMGU－def IMGU－def by simp
with 〈（eligible－literal LP1 $\sigma$ ）〉 have eligible－literal LP1 $\begin{aligned} & \text { P }\end{aligned}$
using lift－eligible－literal by auto
from $\langle\sigma \doteq \vartheta \diamond \sigma\rangle$ and $\langle($ eligible－literal M P2 $\sigma)\rangle$ have eligible－literal M P2 $\vartheta$ using lift－eligible－literal by auto
from 〈min－IMGU $\left.\vartheta u^{\prime} u\right\rangle$ have ck－unifier $u^{\prime} u \vartheta$ FirstOrder unfolding
ck－unifier－def by auto
from $\langle\sigma \doteq \vartheta \diamond \sigma\rangle$ have（subst $u \sigma$ ）$=($ subst（subst $u \vartheta) \sigma$ ）by auto
from $\langle\sigma \doteq \vartheta \diamond \sigma\rangle$ have（subst $v \sigma)=($ subst $($ subst $v \vartheta) \sigma)$ by auto
from 〈（（subst $u \sigma) \neq($ subst $v \sigma))$ 〉
$\langle($ subst $u \sigma)=($ subst $($ subst $u \vartheta) \sigma)\rangle$
$\langle($ subst $v \sigma)=($ subst $($ subst $v \vartheta) \sigma)\rangle$
have（subst（subst u $\vartheta) \sigma) \neq($ subst $($ subst $v \vartheta) \sigma)$ by auto
from this have（subst $u \vartheta) \neq($ subst $v \vartheta)$ by auto
from $\langle\sigma \doteq \vartheta \diamond \sigma\rangle\left\langle\right.$ allowed－redex $\left.u^{\prime} P 1 \sigma\right\rangle$ have allowed－redex $u^{\prime} P 1 \vartheta$
using lift－allowed－redex $\left[\begin{array}{lll}\sigma & \vartheta & \sigma] \text { by auto }\end{array}\right.$
from $\langle\sigma \doteq \vartheta \diamond \sigma\rangle\langle o r i e n t-l i t-i n s t ~ M u v p o s ~ \sigma\rangle$ have orient－lit－inst Muvpos $\vartheta$ using lift－orient－lit－inst by auto
from $\langle\sigma \doteq \vartheta \diamond \sigma\rangle\langle o r i e n t-l i t-i n s t L t s$ polarity $\sigma\rangle$ have orient－lit－inst $L t s$
polarity $\vartheta$
using lift－orient－lit－inst by auto
from $\left\langle\left(C l-C=\left(\right.\right.\right.$ subst－cl $\left((C l-P 1-\{L\}) \cup\left((C l-P 2-\{M\}) \cup\left\{L^{\prime}\right\}\right)\right)$
$\sigma)$ ）＞
and $\left\langle C^{\prime}=(C l-P 1-\{L\}) \cup\left((C l-P 2-\{M\}) \cup\left\{L^{\prime}\right\}\right)\right\rangle$
have $\left(C l-C=\left(\right.\right.$ subst－cl $\left.\left.C^{\prime} \sigma\right)\right)$ by auto
obtain $E$ where $E=(($ trms－ecl P1 $) \cup($ trms－ecl P2 $) \cup$
$\{r . \exists q .(q, p) \in($ pos－ord P1 t $) \wedge($ subterm $t q r)\})$ by auto
from this and $\left\langle\left(C l-C=\left(\right.\right.\right.$ subst－cl $\left.\left.\left.C^{\prime} \sigma\right)\right)\right\rangle$
$\langle$ trms－C $=$（get－trms Cl－C（dom－trms Cl－C（subst－set $(($ trms－ecl P1）$\cup($ trms－ecl P2 $) \cup$
$\{r . \exists q .(q, p) \in($ pos－ord P1 t）$\wedge($ subterm $t q r)\}) \sigma))$ Ground $)\rangle$
have trms－$C=($ get－trms $C l-C$
（dom－trms（subst－cl $C^{\prime} \sigma$ ）（subst－set
E $\sigma$ ）Ground）
by auto
let ？ $\mathrm{Cl}^{\prime}-C^{\prime}=\left(\right.$ subst－cl $\left.C^{\prime} \vartheta\right)$
from $\langle\sigma \doteq \vartheta \diamond \sigma\rangle\langle t r m s-C=$（get－trms $C l-C$
（dom－trms（subst－cl $C^{\prime} \sigma$ ）（subst－set
E $\sigma$ ））Ground）＞
obtain $\exists T^{\prime}$ ．（（subst－set $\left.T^{\prime} \sigma\right) \subseteq$ trms－$C \wedge T^{\prime}=$ get－trms ？$C l-C^{\prime}$
（dom－trms（subst－cl $C^{\prime} \vartheta$ ）（subst－set E $\left.\left.\vartheta\right)\right)$ FirstOrder）
using lift－irreducible－terms by auto
from this obtain $T^{\prime}$ where（subst－set $T^{\prime} \sigma$ ）$\subseteq$ trms－$C$
and $T^{\prime}=$ get－trms ？ $\mathrm{Cl}-\mathrm{C}^{\prime}$
（dom－trms（subst－cl $\left.C^{\prime} \vartheta\right)$（subst－set E $\left.\vartheta\right)$ ）FirstOrder by auto
obtain $C$－fo where $C$－fo $=\left(E c l\right.$ ？$\left.C l-C^{\prime} T^{\prime}\right)$ by auto
from $\left\langle C^{\prime}=(C l-P 1-\{L\}) \cup\left((C l-P 2-\{M\}) \cup\left\{L^{\prime}\right\}\right)\right\rangle$
have $\left(? C l-C^{\prime}=\left(\right.\right.$ subst－cl $\left((C l-P 1-\{L\}) \cup\left((C l-P 2-\{M\}) \cup\left\{L^{\prime}\right\}\right)\right)$
७））
by auto
from $\langle L \in C l-P 1\rangle\langle(M \in C l-P 2)\rangle\langle($ eligible－literal LP1 $\operatorname{l})\rangle\langle($ eligible－literal $M$ P2 Ө）＞

〈（variable－disjoint P1 P2）〉
$\langle(C l-P 1=($ cl－ecl P1 $))\rangle\langle(C l-P 2=($ cl－ecl P2 $))\rangle$
〈（ $\neg$ is－a－variable $\left.u^{\prime}\right)$ 〉
〈（allowed－redex $u^{\prime}$ P1 $\left.\vartheta\right)$ 〉
$\left\langle\left(C-f o=\left(E c l ? C l-C^{\prime} T^{\prime}\right)\right)\right\rangle$
〈（orient－lit－inst Muvpos $\vartheta)$ 〉
〈（orient－lit－inst Lts polarity $\vartheta)\rangle$
$\langle(($ subst u $\vartheta) \neq($ subst $v \vartheta))\rangle$
〈（subterm t p $u^{\prime}$ ）〉
〈（ck－unifier $u^{\prime}$ u $\vartheta$ FirstOrder）〉
〈（replace－subterm t p v $\left.t^{\prime}\right)$ 〉
〈（ $L^{\prime}=m k$－lit polarity $\left.\left.\left(E q t^{\prime} s\right)\right)\right\rangle$

```
\(\left\langle T^{\prime}=\left(\right.\right.\) get-trms ? Cl-C' \({ }^{\prime}\left(\right.\) dom-trms \(\left(\right.\) subst-cl \(\left.C^{\prime} \vartheta\right)(\) subst-set E \(\left.\vartheta)\right)\) FirstOrder \(\left.)\right\rangle\)
    \(\left\langle\left(? C l-C^{\prime}=\left(\right.\right.\right.\) subst-cl \(\left.\left.\left.\left((C l-P 1-\{L\}) \cup\left((C l-P 2-\{M\}) \cup\left\{L^{\prime}\right\}\right)\right) \vartheta\right)\right)\right\rangle\)
    \(\left\langle\left(C^{\prime}=(C l-P 1-\{L\}) \cup\left((C l-P 2-\{M\}) \cup\left\{L^{\prime}\right\}\right)\right)\right\rangle\)
    \(\langle E=((\) trms-ecl P1 \() \cup(\) trms-ecl P2 \() \cup\)
    \(\{r . \exists q .(q, p) \in(\) pos-ord P1 t \() \wedge(\) subterm \(t q r)\})\) )
```

have superposition P1 P2 C-fo $\vartheta$ FirstOrder $C^{\prime}$ unfolding superposition-def
by blast
have subst-cl ?Cl-C' $\sigma=$ subst-cl (subst-cl $((C l-P 1-\{L\}) \cup(C l-P 2-\{M$ \}) $\left.\cup\left\{L^{\prime}\right\}\right)$ Ө) $\sigma$
using $<? C l-C^{\prime}=$ subst-cl $\left((C l-P 1-\{L\}) \cup\left((C l-P 2-\{M\}) \cup\left\{L^{\prime}\right\}\right)\right)$
$\vartheta$ 〉 by auto
also have $\ldots=\left(\right.$ subst-cl $\left((C l-P 1-\{L\}) \cup\left((C l-P 2-\{M\}) \cup\left\{L^{\prime}\right\}\right)\right)(\vartheta$ $\diamond \sigma)$ )
using composition-of-substs-cl $[$ of $((C l-P 1-\{L\}) \cup((C l-P 2-\{M\}) \cup$ $\left.\left.\left.\left\{L^{\prime}\right\}\right)\right)\right]$ by auto
also have $\ldots=\left(\right.$ subst-cl $\left.\left((C l-P 1-\{L\}) \cup\left((C l-P 2-\{M\}) \cup\left\{L^{\prime}\right\}\right)\right) \sigma\right)$
using subst-eq-cl[OF $\langle\sigma \doteq \vartheta \diamond \sigma\rangle]$ by blast
also have...$=C l-C$
using $\left\langle C l-C=\left(\right.\right.$ subst-cl $\left((C l-P 1-\{L\}) \cup\left((C l-P 2-\{M\}) \cup\left\{L^{\prime}\right\}\right)\right)$
$\sigma)>$ by argo
finally have subst-cl (cl-ecl C-fo) $\sigma=c l-e c l ~ C$
using $\langle C=E c l C l-C$ trms- $C\rangle\left\langle C\right.$-fo $=E c l$ ? $\left.C l-C^{\prime} T^{\prime}\right\rangle$ by simp
moreover have (subst-set (trms-ecl C-fo) $\sigma$ ) $\subseteq($ trms-ecl $C)$
using 〈subst-set $T^{\prime} \sigma \subseteq$ trms- $\left.C\right\rangle\langle C=E c l C l-C$ trms- $C\rangle\left\langle C\right.$-fo $=E c l$ ? $C l-C^{\prime}$
$T^{\prime}>$ by auto
ultimately have (trms-subsumes $C$-fo $C \sigma$ )
unfolding trms-subsumes-def by auto
with «superposition P1 P2 $C$-fo $\vartheta$ FirstOrder $\left.C^{\prime}\right\rangle\langle\sigma \doteq \vartheta \diamond \sigma\rangle$ hyp show False by auto
qed
from not-sup and assms(1) show False by blast qed
lemma lifting-lemma-factorization:
assumes factorization P1 $C \sigma$ Ground $C^{\prime}$
shows $\exists D \vartheta$. factorization P1 $D \vartheta$ FirstOrder $C^{\prime} \wedge \sigma \doteq \vartheta \diamond \sigma \wedge$ trms-subsumes DC $\sigma$
proof (rule ccontr)
assume hyp: $\ddagger D \vartheta$. factorization P1 D $\vartheta$ FirstOrder $C^{\prime} \wedge \sigma \doteq \vartheta \diamond \sigma \wedge$ trms-subsumes $D C \sigma$
have not-fact: $\neg$ factorization P1 $C \sigma$ Ground $C^{\prime}$
proof (rule notI)
assume factorization P1 $C \sigma$ Ground $C^{\prime}$
from this obtain L1 L2 L'ts uv Cl-P Cl-C trms- $C$ where
eligible-literal L1 P1 $\sigma$

```
    L1 \(\in(c l-e c l ~ P 1) L 2 \in(c l-e c l ~ P 1)-\{L 1\} C l-C=(c l-e c l C)(C l-P=(c l-e c l\)
P1))
    (orient-lit-inst L1 ts pos \(\sigma\) )
    (orient-lit-inst L2 u v pos \(\sigma\) )
    \(((\) subst \(t \sigma) \neq(\) subst \(s \sigma))\)
    (subst \(t \sigma) \neq(\) subst \(v \sigma)\)
    (ck-unifier t u \(\sigma\) Ground)
    \(\left(L^{\prime}=\operatorname{Neg}(E q s v)\right)\)
    \(C=(E c l C l-C\) trms-C)
    trms- \(C=\) (get-trms Cl-C
        \((\) dom-trms Cl-C \((\) subst-set \(((\) trms-ecl P1 \() \cup(\) proper-subterms-of \(t)) \sigma))\)
Ground
    \(\left(C l-C=\left(\right.\right.\) subst-cl \(\left.\left.\left((C l-P-\{L 2\}) \cup\left\{L^{\prime}\right\}\right)\right) \sigma\right)\)
    \(\left(C^{\prime}=\left((C l-P-\{L 2\}) \cup\left\{L^{\prime}\right\}\right)\right)\)
    unfolding factorization-def get-trms-def by auto
    from 〈(ck-unifier t u \(\sigma\) Ground) \(\rangle\) have Unifier \(\sigma t u\)
        unfolding ck-unifier-def Unifier-def by auto
    from this have (subst \(t \sigma)=(\) subst \(u \sigma\) ) unfolding Unifier-def by auto
    from this have unify \(t u \neq\) None using MGU-exists by auto
    from this obtain \(\vartheta\) where unify \(t u=S o m e \vartheta\) by auto
    hence min-IMGU \(\vartheta t u\) by (rule unify-computes-min-IMGU)
    with 〈Unifier \(\sigma t u\rangle\) have \(\sigma \doteq \vartheta \diamond \sigma\)
        unfolding min-IMGU-def IMGU-def by simp
    with \(\langle(\) eligible-literal L1 P1 \(\sigma\) ) 〉 have eligible-literal L1 P1 \(\vartheta\)
        using lift-eligible-literal by auto
    from \(\langle\) min-IMGU \(\vartheta t u\rangle\) have \(c k\)-unifier \(t u \vartheta\) FirstOrder unfolding \(c k\)-unifier-def
by auto
    from \(\langle\sigma \doteq \vartheta \diamond \sigma\rangle\) have (subst \(t \sigma)=(\) subst (subst \(t \vartheta) \sigma\) ) by auto
    from \(\langle\sigma \doteq \vartheta \diamond \sigma\rangle\) have (subst s \(\sigma\) ) \(=(\) subst (subst s \(\vartheta) \sigma\) ) by auto
    from \(\langle\sigma \doteq \vartheta \diamond \sigma\rangle\) have (subst \(v \sigma)=(\) subst (subst \(v \vartheta) \sigma\) ) by auto
    from 〈((subst \(t \sigma) \neq(\) subst \(s \sigma))\rangle\)
        \(\langle(\) subst \(t \sigma)=(\) subst \((\) subst \(t \vartheta) \sigma)\rangle\)
        \(\langle(\) subst \(s \sigma)=(\) subst \((\) subst \(s\) Э) \(\sigma)\rangle\)
        have (subst (subst t \(\vartheta) \sigma\) ) \(\neq(\) subst (subst \(s \vartheta) \sigma\) ) by auto
    from this have (subst \(t \vartheta) \neq(\) subst \(s \vartheta)\) by auto
    from 〈((subst \(t \sigma) \neq(\) subst \(v \sigma))\rangle\)
        \(\langle(\) subst \(t \sigma)=(\) subst \((\) subst \(t \vartheta) \sigma)\rangle\)
        \(\langle(\) subst \(v \sigma)=(\) subst \((\) subst \(v \vartheta) \sigma)\rangle\)
        have (subst (subst \(t \vartheta) \sigma) \neq(\) subst (subst \(v \vartheta) \sigma\) ) by auto
    from this have (subst \(t \vartheta) \neq(\) subst \(v \vartheta)\) by auto
    from \(\langle\sigma \doteq \vartheta \diamond \sigma\rangle\langle o r i e n t-l i t-i n s t L 1 t s p o s ~ \sigma\rangle\) have orient-lit-inst L1 tspos \(\vartheta\)
        using lift-orient-lit-inst by auto
    from \(\langle\sigma \doteq \vartheta \diamond \sigma\rangle\langle o r i e n t-l i t-i n s t L 2 u v\) pos \(\sigma\rangle\) have orient-lit-inst L2 uvpos
\(\vartheta\)
    using lift-orient-lit-inst by auto
```

from $\left\langle\left(C l-C=\left(\right.\right.\right.$ subst－cl $\left.\left.\left.\left((C l-P-\{L 2\}) \cup\left\{L^{\prime}\right\}\right)\right) \sigma\right)\right\rangle$
and $\left\langle C^{\prime}=\left((C l-P-\{L 2\}) \cup\left\{L^{\prime}\right\}\right)\right\rangle$
have $\left(C l-C=\left(\right.\right.$ subst－cl $\left.\left.C^{\prime} \sigma\right)\right)$ by auto
obtain $E$ where $E=($ trms－ecl P1）by auto
from this and $\left\langle\left(C l-C=\left(\right.\right.\right.$ subst－cl $\left.\left.\left.C^{\prime} \sigma\right)\right)\right\rangle$
$\langle t r m s-C=($ get－trms $\quad C l-C$
（dom－trms Cl－C（subst－set $(($ trms－ecl P1 $) \cup($ proper－subterms－of $t)) \sigma))$
Ground＞
have trms－$C=($ get－trms $\mathrm{Cl}-\mathrm{C}$
（dom－trms（subst－cl $C^{\prime} \sigma$ ）（subst－set
$(E \cup($ proper－subterms－of $t)) \sigma))$ Ground $)$
by auto
let ${ }^{\text {？}} \mathrm{Cl}-C^{\prime}=\left(\right.$ subst－cl $\left.C^{\prime} \vartheta\right)$
from $\langle\sigma \doteq \vartheta \diamond \sigma\rangle\langle t r m s-C=$（get－trms $C l-C$
（dom－trms（subst－cl $C^{\prime} \sigma$ ）（subst－set
$(E \cup($ proper－subterms－of $t)) \sigma))$ Ground $)$＞
obtain $T^{\prime}$ where（subst－set $\left.T^{\prime} \sigma\right) \subseteq$ trms－C
and $T^{\prime}=$ get－trms ？ $\mathrm{Cl}-\mathrm{C}^{\prime}$
（dom－trms（subst－cl $\left.C^{\prime} \vartheta\right)($ subst－set $(E \cup($ proper－subterms－of t））$\vartheta))$ FirstOrder using lift－irreducible－terms by blast
obtain $C$－fo where $C$－fo $=\left(E c l\right.$ ？$\left.C l-C^{\prime} T^{\prime}\right)$ by auto
from $\left\langle C^{\prime}=\left((C l-P-\{L 2\}) \cup\left\{L^{\prime}\right\}\right)\right\rangle$
have $\left(? C l-C^{\prime}=\left(\right.\right.$ subst－cl $\left.\left.\left((C l-P-\{L 2\}) \cup\left\{L^{\prime}\right\}\right) \vartheta\right)\right)$
by auto
from $\left\langle C\right.$－fo $\left.=\left(E c l ? C l-C^{\prime} T^{\prime}\right)\right\rangle$ have ？$C l-C^{\prime}=($ cl－ecl $C$－fo $)$ by auto
have ？ $\mathrm{Cl}^{\prime}-C^{\prime}=\left(\right.$ subst－cl $\left.C^{\prime} \vartheta\right)$ by auto
from
〈eligible－literal L1 P1 ${ }^{\text {〉 }}$ 〉
$\langle L 1 \in(c l-e c l P 1)\rangle\langle L 2 \in($ cl－ecl P1 $)-\{L 1\}\rangle\left\langle ? C l-C^{\prime}=(c l-e c l C-f o)\right\rangle\langle(C l-P$ $=($ cl－ecl P1 $)$ ）$\rangle$
$\langle($ orient－lit－inst L1 tspos $\vartheta)\rangle$
〈（orient－lit－inst L2 u v pos $\vartheta)\rangle$
$\langle(($ subst $t \vartheta) \neq($ subst s $\vartheta))\rangle$
$\langle($ subst $t \vartheta) \neq($ subst $v \vartheta)\rangle$
〈（ck－unifier t u $\vartheta$ FirstOrder）〉
$\left.\prec\left(L^{\prime}=\operatorname{Neg}(E q s v)\right)\right\rangle$
$\left\langle C-f o=\left(E c l ? C l-C^{\prime} T^{\prime}\right)\right\rangle$
$\left\langle T^{\prime}=\right.$ get－trms？${ }^{\text {Cl}} \mathrm{Cl}^{\prime}$
（dom－trms ？Cl－C＇（subst－set $(E \cup($ proper－subterms－of t））$\vartheta))$ FirstOrder＞ $\left\langle\left(? C l-C^{\prime}=\left(\right.\right.\right.$ subst－cl $\left.\left.\left.\left((C l-P-\{L 2\}) \cup\left\{L^{\prime}\right\}\right) \vartheta\right)\right)\right\rangle$
$\left\langle C^{\prime}=\left((C l-P-\{L 2\}) \cup\left\{L^{\prime}\right\}\right)\right\rangle$
$\langle E=($ trms－ecl P1）$\rangle$
have factorization P1 C－fo $\vartheta$ FirstOrder $C^{\prime}$ unfolding factorization－def by blast
have $i$ ：subst－cl ？Cl－$C^{\prime} \sigma=$ subst－cl（subst－cl $\left.\left(C l-P-\{L 2\} \cup\left\{L^{\prime}\right\}\right) \vartheta\right) \sigma$
using 〈?Cl-C' $=$ subst-cl $\left.\left((C l-P-\{L 2\}) \cup\left\{L^{\prime}\right\}\right) \vartheta\right\rangle$ by auto
have ii: subst-cl (subst-cl $\left.\left((C l-P-\{L 2\}) \cup\left\{L^{\prime}\right\}\right) \vartheta\right) \sigma$
$=$ subst-cl $\left((C l-P-\{L 2\}) \cup\left\{L^{\prime}\right\}\right)(\vartheta \diamond \sigma)$
using composition-of-substs-cl $\left[\right.$ of $\left.\left((C l-P-\{L 2\}) \cup\left\{L^{\prime}\right\}\right)\right]$ by auto
from $\langle\sigma \doteq \vartheta \diamond \sigma\rangle$ have (subst-cl $\left.\left((C l-P-\{L 2\}) \cup\left\{L^{\prime}\right\}\right) \sigma\right)$
$=\left(\right.$ subst-cl $\left.\left((C l-P-\{L 2\}) \cup\left\{L^{\prime}\right\}\right)(\vartheta \diamond \sigma)\right)$
using subst-eq-cl $\left[\right.$ of $\left.\sigma \vartheta \diamond \sigma\left((C l-P-\{L 2\}) \cup\left\{L^{\prime}\right\}\right)\right]$ by auto
with $i i i\left\langle C l-C=\left(\right.\right.$ subst-cl $\left.\left.\left((C l-P-\{L 2\}) \cup\left\{L^{\prime}\right\}\right) \sigma\right)\right\rangle$
have (subst-cl ? Cl- $\left.C^{\prime} \sigma\right)=C l-C$ by metis
with $\langle C=E c l C l-C$ trms- $C\rangle\left\langle C\right.$-fo $\left.=E c l ? C l-C^{\prime} T^{\prime}\right\rangle$ have subst-cl (cl-ecl C-fo)
$\sigma=$ cl-ecl $C$
by auto
from $\left\langle\left(\right.\right.$ subst-set $\left.T^{\prime} \sigma\right) \subseteq$ trms- $\left.C\right\rangle$
and $\langle(C=(E c l C l-C$ trms $-C))\rangle$ and $\left\langle\left(C-f o=\left(E c l\right.\right.\right.$ ? $\left.\left.\left.C l-C^{\prime} T^{\prime}\right)\right)\right\rangle$
have (subst-set (trms-ecl C-fo) $\sigma$ ) $\subseteq($ trms-ecl $C$ ) by auto
from 〈(subst-cl (cl-ecl C-fo) $\sigma)=($ cl-ecl $C)\rangle\langle($ subst-set $($ trms-ecl $C$-fo) $\sigma) \subseteq$ (trms-ecl C) >
have (trms-subsumes $C$-fo $C \sigma$ )
unfolding trms-subsumes-def by auto
with $\left\langle\right.$ factorization P1 C-fo $\vartheta$ FirstOrder $\left.C^{\prime}\right\rangle\langle\sigma \doteq \vartheta \diamond \sigma\rangle$ hyp show False by auto
qed
from not-fact and assms(1) show False by blast
qed
lemma lifting-lemma-reflexion:
assumes reflexion P1 $C \sigma$ Ground $C^{\prime}$
shows $\exists D \vartheta$. reflexion P1 $D \vartheta$ FirstOrder $C^{\prime} \wedge \sigma \doteq \vartheta \diamond \sigma \wedge$ trms-subsumes $D$
C $\sigma$
proof (rule ccontr)
assume hyp: $\ddagger D \vartheta$. reflexion P1 D $\vartheta$ FirstOrder $C^{\prime} \wedge \sigma \doteq \vartheta \diamond \sigma \wedge$ trms-subsumes D $C \sigma$

```
    have not-ref: \(\neg\) reflexion P1 \(C \sigma\) Ground \(C^{\prime}\)
    proof (rule notI)
    assume reflexion P1 \(C \sigma\) Ground \(C^{\prime}\)
    from this obtain \(L 1 t s C l-P C l-C\) trms- \(C\) where
        eligible-literal L1 P1 \(\sigma\)
        L1 \(\in\) cl-ecl P1 Cl-C \(=\) cl-ecl \(C C l-P=c l-e c l ~ P 1\)
        orient-lit-inst L1 t s neg \(\sigma\)
        ck-unifier \(t s \sigma\) Ground
        \(C=E c l C l-C\) trms-C
        trms- \(C=\) get-trms \(\mathrm{Cl}-\mathrm{C}\)
            (dom-trms Cl-C (subst-set (trms-ecl P1 \(\cup\{t\}) \sigma\) )) Ground
        \(\mathrm{Cl}-\mathrm{C}=\) subst-cl \((C l-P-\{L 1\}) \sigma\)
        \(C^{\prime}=C l-P-\{L 1\}\)
        unfolding reflexion-def get-trms-def by auto
```

from 〈ck－unifier ts Ground〉 have Unifier $\sigma t s$ unfolding ck－unifier－def Unifier－def by auto
hence subst $t \sigma=$ subst $s \sigma$ unfolding Unifier－def by auto
hence unify $t s \neq$ None using $M G U$－exists by auto
then obtain $\vartheta$ where unify $t s=S o m e \vartheta$ by auto
hence min－IMGU $\vartheta t s$ by（rule unify－computes－min－IMGU）
with 〈Unifier $\sigma t s\rangle$ have $\sigma \doteq \vartheta \diamond \sigma$ unfolding $I M G U$－def min－IMGU－def by simp
with «eligible－literal L1 P1 $\sigma$ 〉 have eligible－literal L1 P1 $\vartheta$
using lift－eligible－literal by auto
from $\langle$ min－IMGU $\vartheta t s\rangle$ have $c k$－unifier $t s \vartheta$ FirstOrder unfolding $c k$－unifier－def by auto
from $\langle\sigma \doteq \vartheta \diamond \sigma\rangle\langle o r i e n t-l i t-i n s t L 1 t s$ neg $\sigma\rangle$ have orient－lit－inst L1 ts neg $\vartheta$
using lift－orient－lit－inst by auto
from $\langle C l-C=$ subst－cl $(C l-P-\{L 1\}) \sigma\rangle$ and $\left\langle C^{\prime}=C l-P-\{L 1\}\right\rangle$
have $C l-C=$ subst－cl $C^{\prime} \sigma$ by auto
obtain $E$ where $E=($ trms－ecl P1）by auto
with $\left\langle C l-C=\right.$ subst－cl $\left.C^{\prime} \sigma\right\rangle$
$\langle$ trms－$C=$ get－trms Cl－C（dom－trms Cl－C（subst－set（trms－ecl P1 $\cup\{t\})$
$\sigma$ ））Ground $>$
have trms－$C=$ get－trms $C l-C$（dom－trms（subst－cl $\left.C^{\prime} \sigma\right)($ subst－set $(E \cup\{t\})$
$\sigma)$ ）Ground
by auto
let $? \mathrm{Cl}-\mathrm{C}^{\prime}=$ subst－cl $C^{\prime} \vartheta$
from $\langle\sigma \doteq \vartheta \diamond \sigma\rangle$
$\left\langle\right.$ trms－$C=$ get－trms $C l-C$（dom－trms（subst－cl $\left.C^{\prime} \sigma\right)($ subst－set $(E \cup\{t\})$
б））Ground＞
obtain $T^{\prime}$ where
subst－set $T^{\prime} \sigma \subseteq$ trms－$C$ and
$T^{\prime}=$ get－trms ？Cl－$C^{\prime}\left(\right.$ dom－trms $\left(\right.$ subst－cl $\left.C^{\prime} \vartheta\right)($ subst－set $\left.(E \cup\{t\}) \vartheta)\right)$
FirstOrder
using lift－irreducible－terms by blast
obtain $C$－fo where $C$－fo $=E c l ? C l-C^{\prime} T^{\prime}$ by auto
from $\left\langle C^{\prime}=C l-P-\{L 1\}\right\rangle$
have ？$C l-C^{\prime}=$ subst－cl $(C l-P-\{L 1\}) \vartheta$ by auto
from $\left\langle C\right.$－fo $=E c l$ ？$\left.C l-C^{\prime} T^{\prime}\right\rangle$ have ？$C l-C^{\prime}=c l-e c l C$－fo by auto
have ？ $\mathrm{Cl}-C^{\prime}=\left(\right.$ subst－cl $\left.C^{\prime} \vartheta\right)$ by auto

## from

〈eligible－literal L1 P1 ७〉
$\langle L 1 \in$ cl－ecl P1〉〈？Cl－C＇$=$ cl－ecl $C$－fo〉〈Cl－P $=$ cl－ecl P1〉
〈orient－lit－inst L1 ts neg $\vartheta^{\text {〉 }}$
$\langle c k$－unifier t s $\vartheta$ FirstOrder〉
$\left\langle C\right.$－fo $\left.=E c l ? ~ C l-C^{\prime} T^{\prime}\right\rangle$
$\left\langle T^{\prime}=\right.$ get－trms ？ $\mathrm{Cl}-\mathrm{C}^{\prime}$
（dom－trms（subst－cl $\left.C^{\prime} \vartheta\right)($ subst－set $(E \cup\{t\}) \vartheta)$ ）FirstOrder＞ $\left\langle ? C l-C^{\prime}=\right.$ subst－cl $\left.(C l-P-\{L 1\}) \vartheta\right\rangle$
$\left\langle C^{\prime}=C l-P-\{L 1\}\right\rangle$
$\langle E=$ trms－ecl P1〉
have reflexion P1 C－fo $\vartheta$ FirstOrder $C^{\prime}$
unfolding reflexion－def by metis
have $i$ ：subst－cl ？Cl－C＇$\sigma=$ subst－cl（subst－cl（Cl－P $-\{L 1\}) \vartheta) \sigma$
using $\left\langle ? C l-C^{\prime}=\right.$ subst－cl $\left.(C l-P-\{L 1\}) \vartheta\right\rangle$ by auto
have ii：subst－cl（subst－cl（Cl－P $-\{L 1\}) \vartheta) \sigma=\operatorname{subst-cl}(C l-P-\{L 1\})$ $(\vartheta \diamond \sigma)$
using composition－of－substs－cl［of Cl－P－\｛ L1 \}] by auto
from $\langle\sigma \doteq \vartheta \diamond \sigma\rangle$ have subst－cl $(C l-P-\{L 1\}) \sigma=$ subst－cl $(C l-P-\{L 1$ \}) $(\vartheta \diamond \sigma)$
using subst－eq－cl［of $\sigma \vartheta \diamond \sigma C l-P-\{L 1\}]$ by auto
with $i i i\langle C l-C=($ subst－cl $((C l-P-\{L 1\})) \sigma)\rangle$
have subst－cl ？ $\mathrm{Cl}-\mathrm{C}^{\prime} \sigma=\mathrm{Cl}-\mathrm{C}$ by metis
with $\langle C=E c l C l-C$ trms－$C\rangle\left\langle C\right.$－fo $\left.=E c l ? C l-C^{\prime} T^{\prime}\right\rangle$
have subst－cl（cl－ecl $C$－fo）$\sigma=$ cl－ecl $C$ by auto
from 〈subst－set $T^{\prime} \sigma \subseteq$ trms－$\left.C\right\rangle\langle C=E c l C l-C$ trms－$C\rangle\left\langle C\right.$－fo $=E c l ? C l-C^{\prime}$ $T^{\prime \prime}$
have subst－set（trms－ecl $C$－fo）$\sigma \subseteq$ trms－ecl $C$ by auto
from «subst－cl（cl－ecl C－fo）$\sigma=$ cl－ecl $C\rangle\langle s u b s t-s e t(t r m s$－ecl $C$－fo）$\sigma \subseteq$ trms－ecl C $>$
have trms－subsumes C－fo $C \sigma$ unfolding trms－subsumes－def by auto
with $\left\langle\right.$ reflexion P1 C－fo $\vartheta$ FirstOrder $\left.C^{\prime}\right\rangle\langle\sigma \doteq \vartheta \diamond \sigma\rangle$ hyp show False by auto qed
from not－ref and assms（1）show False by blast
qed
lemma lifting－lemma：
assumes deriv：derivable $C$ P $S \sigma$ Ground $C^{\prime}$
shows $\exists D \vartheta$ ．$\left(\left(\right.\right.$ derivable $D P S \vartheta$ FirstOrder $\left.C^{\prime}\right) \wedge(\sigma \doteq \vartheta \diamond \sigma) \wedge($ trms－subsumes $D(\sigma))$
proof（rule ccontr）
assume hyp：$\neg\left(\exists D \vartheta\right.$ ．derivable $D P S \vartheta$ FirstOrder $C^{\prime} \wedge \sigma \doteq \vartheta \diamond \sigma \wedge$ trms－subsumes $D C \sigma$ ）
from deriv have $P \subseteq S$ unfolding derivable－def by auto
have not－sup：$\neg(\exists$ P1 P2．$P=\{P 1, P 2\} \wedge$ superposition P1 P2 $C \sigma$ Ground $C^{\prime}$ ）
using lifting－lemma－superposition
by（metis $\langle P \subseteq S\rangle$ derivable－def hyp insert－subset）
have not－fact：$\neg\left(\exists P 1 .\{P 1\}=P \wedge\right.$ factorization P1 $C \sigma$ Ground $\left.C^{\prime}\right)$
using lifting－lemma－factorization
by（metis $\langle P \subseteq S\rangle$ derivable－def hyp insert－subset）
have not－ref：$\neg\left(\exists P 1 .\{P 1\}=P \wedge\right.$ reflexion P1 $C \sigma$ Ground $\left.C^{\prime}\right)$
using lifting－lemma－reflexion
by（metis $\langle P \subseteq S\rangle$ derivable－def hyp insert－subset）
from not－sup not－ref not－fact deriv show False unfolding derivable－def by blast qed
lemma trms－subsumes－and－red－inf：
assumes trms－subsumes $D C \eta$
assumes redundant－inference（subst－ecl D $\eta$ ）S P $\sigma$
assumes $\sigma \doteq \vartheta \diamond \eta$
shows redundant－inference $C S P \sigma$
proof－
from $\operatorname{assms}(2)$ obtain $S^{\prime}$ where $S^{\prime} \subseteq($ instances $S)$
（set－entails－clause（clset－instances $\left.S^{\prime}\right)($ cl－ecl（subst－ecl $\left.D \eta)\right)$ ）
（ $\forall x \in S^{\prime}$ ．（ subterms－inclusion（subst－set（trms－ecl（fst x））（snd $\left.x\right)$ ）
（trms－ecl（subst－ecl D $\eta$ ））））and
ball－S＇－le：$\forall x \in S^{\prime} . \exists D^{\prime} \in$ cl－ecl＇$P$ ．$\left((\right.$ cl－ecl $(f s t x)$, snd $\left.x),\left(D^{\prime}, \sigma\right)\right) \in$ cl－ord unfolding redundant－inference－def by auto
from $\operatorname{assms}(1)$ have $(s u b s t-c l(c l-e c l ~ D) ~ \eta)=($ cl－ecl $C)$
unfolding trms－subsumes－def by auto
obtain $C l-D T$ where $D=E c l C l-D T$ using eclause．exhaust by auto
from this have（cl－ecl $D)=C l-D$ and trms－ecl $D=T$ by auto
from $\langle D=E c l C l-D T\rangle$ have subst－ecl $D \eta=E c l$（subst－cl Cl－D $\eta$ ）（subst－set $T$
7）
by auto
from this have（cl－ecl（subst－ecl $D \eta)$ ）$=($ subst－cl Cl－D $\eta$ ）
and trms－ecl（subst－ecl $D \eta)=($ subst－set $T \eta)$ by auto
from 〈（cl－ecl $($ subst－ecl $D \eta))=($ subst－cl Cl－D $\eta)$ 〉
and $\langle(c l-e c l ~ D)=C l-D\rangle\langle(s u b s t-c l(c l-e c l D) \eta)=(c l-e c l C)\rangle$
have $($ cl－ecl $($ subst－ecl $D \eta))=($ cl－ecl C）by auto
from this and 〈（set－entails－clause（clset－instances $\left.S^{\prime}\right)($ cl－ecl（subst－ecl D $\eta$ ）））〉
have（set－entails－clause（clset－instances $\left.S^{\prime}\right)($ cl－ecl $C)$ ）by auto
from 〈trms－ecl $D=T\rangle$ and $\langle$ trms－ecl $($ subst－ecl $D \eta)=($ subst－set $T \eta)\rangle$

```
    have trms-ecl (subst-ecl D \eta)=(subst-set (trms-ecl D) \eta) by auto
    from assms(1) have (subst-set (trms-ecl D) \eta)\subseteq(trms-ecl C)
    unfolding trms-subsumes-def by auto
    have ii:(\forallx\in S'.( subterms-inclusion (subst-set (trms-ecl (fst x)) (snd x))
    (trms-ecl C)))
    proof (rule ccontr)
    assume }\neg(\forallx\in\mp@subsup{S}{}{\prime}.(\mathrm{ subterms-inclusion (subst-set (trms-ecl (fst x)) (snd x))
            (trms-ecl C)))
    from this obtain x where }x\in\mp@subsup{S}{}{\prime}\mathrm{ and
        \neg( subterms-inclusion (subst-set (trms-ecl (fst x)) (snd x)) (trms-ecl C))
        by auto
    from «x \in S'` and }<(\forallx\in\mp@subsup{S}{}{\prime}.(\mathrm{ subterms-inclusion (subst-set (trms-ecl (fst x))
(snd x))
            (trms-ecl (subst-ecl D \eta))))>
        have ( subterms-inclusion (subst-set (trms-ecl (fst x)) (snd x))
            (trms-ecl (subst-ecl D \eta))) by auto
    obtain E1 where E1 = (subst-set (trms-ecl (fst x)) (snd x)) by auto
    obtain E2 where E2 = (subst-set (trms-ecl D) \eta) by auto
    obtain E2' where E2' = (trms-ecl C) by auto
    from <E2 = (subst-set (trms-ecl D) \eta)\rangle\langleE2' = (trms-ecl C)\rangle
            <(subst-set (trms-ecl D) \eta)\subseteq(trms-ecl C)>
        have EQ \subseteqEQ' by auto
    from <E1 = (subst-set (trms-ecl (fst x)) (snd x))>
                <E2 = (subst-set (trms-ecl D) \eta)\rangle
                <trms-ecl (subst-ecl D \eta) = (subst-set (trms-ecl D) \eta)>
                <( subterms-inclusion (subst-set (trms-ecl (fst x)) (snd x))
                    (trms-ecl (subst-ecl D \eta)))> have subterms-inclusion E1 E2 by auto
    from this and <E2 \subseteqE2'> have subterms-inclusion E1 E2'
        using subterms-inclusion-subset by auto
    from this and <E1 = (subst-set (trms-ecl (fst x)) (snd x) )\rangle\langleE2' = (trms-ecl
C)>
        and}«\neg(\mathrm{ subterms-inclusion (subst-set (trms-ecl (fst x)) (snd x))
                        (trms-ecl C))> show False by auto
    qed
    from this and <(set-entails-clause (clset-instances S') (cl-ecl C))>
        and ball-S'-le
        and \langleS'\subseteq(instances S)\rangle
    show redundant-inference C S P\sigma unfolding redundant-inference-def by auto
qed
lemma lift-inference:
    assumes inference-saturated S
    shows ground-inference-saturated S
proof (rule ccontr)
    assume }\neg(\mathrm{ ground-inference-saturated S)
    from this obtain CP\sigma C' where derivable C PS \sigma Ground C' ground-clause
(cl-ecl C)
```

grounding-set $P \sigma \neg$ redundant-inference $C S P \sigma$ unfolding ground-inference-saturated-def
by blast
from «derivable $C$ P $S \sigma$ Ground $C^{\prime}$ 〉 obtain $D \vartheta \eta$ where derivable DPS $\vartheta$ FirstOrder $C^{\prime}$
$\sigma \doteq \vartheta \diamond \eta$ trms-subsumes $D C \eta$ using lifting-lemma by blast
from 〈trms-subsumes $D C \eta\rangle$ and $\langle\neg$ redundant-inference $C S P \sigma\rangle\langle\sigma \doteq \vartheta \diamond \eta\rangle$
have $\neg$ redundant-inference (subst-ecl $D \eta$ ) S P $\sigma$
using trms-subsumes-and-red-inf by auto
from this and «derivable CPS Ground $\left.C^{\prime}\right\rangle\langle d e r i v a b l e ~ D P S \vartheta$ FirstOrder $\left.C^{\prime}\right\rangle\langle t r m s$-subsumes $D C \eta\rangle$
$\langle\sigma \doteq \vartheta \diamond \eta\rangle\langle g r o u n d-c l a u s e($ cl-ecl $C)\rangle\langle g r o u n d i n g-s e t P \sigma\rangle$
$\langle$ redundant-inference (subst-ecl D $\eta$ ) $S P \sigma\rangle$
assms(1) show False unfolding inference-saturated-def by auto
qed
lemma lift-redundant-cl :
assumes $C^{\prime}=$ subst-cl $D \vartheta$
assumes redundant-clause $C S \eta C^{\prime}$
assumes $\sigma \doteq \vartheta \diamond \eta$
assumes finite $D$
shows redundant-clause $C S \sigma D$
proof -
from $\operatorname{assms}$ (2) have
$\left(\exists S^{\prime} .\left(S^{\prime} \subseteq(\right.\right.$ instances $S) \wedge\left(\right.$ set-entails-clause $\left(\right.$ clset-instances $\left.S^{\prime}\right)($ cl-ecl $\left.C)\right)$ $\wedge$
$\left(\forall x \in S^{\prime} .(\right.$ subterms-inclusion (subst-set (trms-ecl $(f s t x))($ snd $\left.x)\right)$
(trms-ecl C))) $\wedge$
$\left(\forall x \in S^{\prime} .\left(\left((\right.\right.\right.$ mset-ecl $(($ fst $x),($ snd $x))),\left(\right.$ mset-cl $\left.\left.\left(C^{\prime}, \eta\right)\right)\right) \in($ mult (mult trm-ord))
$\vee($ mset-ecl $\left.\left.\left.\left.((f s t x),(\operatorname{snd} x)))=\operatorname{mset}-c l\left(C^{\prime}, \eta\right)\right)\right)\right)\right)$
unfolding redundant-clause-def by auto
from this obtain $S^{\prime}$ where $i: S^{\prime} \subseteq($ instances $S)$
and ii: (set-entails-clause (clset-instances $\left.S^{\prime}\right)($ cl-ecl $\left.C)\right)$
and iii: $\left(\forall x \in S^{\prime}\right.$. ( subterms-inclusion (subst-set (trms-ecl $(f$ st $\left.x)\right)($ snd $\left.x)\right)$
(trms-ecl $C$ )))
and
iv: $\left(\forall x \in S^{\prime} .\left(\left((\right.\right.\right.$ mset-ecl $((f s t x),($ snd $x))),\left(\right.$ mset-cl $\left.\left.\left(C^{\prime}, \eta\right)\right)\right) \in($ mult (mult trm-ord))
$\vee($ mset-ecl $(($ fst $x),($ snd $x)))=$ mset-cl $\left.\left.\left(C^{\prime}, \eta\right)\right)\right)$
by auto
let ? $m 1=$ mset-cl $\left(C^{\prime}, \eta\right)$
let $? m 2=\operatorname{mset}-c l(D, \sigma)$
from $\operatorname{assms}(1) \operatorname{assms}(3) \operatorname{assms}(4)$
have mset-cl $\left(C^{\prime}, \eta\right)=\operatorname{mset}-c l(D, \sigma) \vee\left(\right.$ mset-cl $\left(C^{\prime}, \eta\right)$, mset-cl $\left.(D, \sigma)\right) \in($ mult (mult trm-ord))
using mset-subst by auto
from this iv
have $\left(\forall x \in S^{\prime} .(((\right.$ mset-ecl $((f s t x),($ snd $x))),($ mset-cl $(D, \sigma))) \in($ mult (mult

```
trm-ord))
                \(\vee(\) mset-ecl \(((f s t x),(\) snd \(x)))=\) mset-cl \((D, \sigma)))\)
```

    using mult-mult-trm-ord-trans unfolding trans-def by metis
    from this and \(i\) ii iii show ?thesis unfolding redundant-clause-def by meson
    qed

We deduce the following（trivial）lemma，stating that sets that are closed under all inferences are also saturated．

```
lemma inference-closed-sets-are-saturated:
    assumes inference-closed \(S\)
    assumes \(\forall x \in S\). (finite (cl-ecl \(x\) ))
    shows clause-saturated \(S\)
proof (rule ccontr)
    assume \(\neg\) ? thesis
    from this obtain \(C P \sigma C^{\prime} D \vartheta \eta\)
        where
            (derivable C P S G Ground \(C^{\prime}\) ) (ground-clause (cl-ecl C))
            (derivable \(D P S \vartheta\) FirstOrder \(\left.C^{\prime}\right)(\) trms-subsumes \(D C \eta)\)
            \((\sigma \doteq \vartheta \diamond \eta)\)
            \(\neg\left(\right.\) redundant-clause (subst-ecl D \(\eta\) ) \(S \sigma C^{\prime}\) )
            unfolding clause-saturated-def by blast
    from «(derivable D P \(S\) ७ FirstOrder \(\left.\left.C^{\prime}\right)\right\rangle\) assms(1) have \(D \in S\)
        unfolding inference-closed-def by auto
    from «derivable \(D P S \vartheta\) FirstOrder \(C^{\prime} 〉\) have \((\) cl-ecl \(D)=\left(\right.\) subst-cl \(\left.C^{\prime} \vartheta\right)\)
        using derivable-clauses-lemma by auto
    from 〈trms-subsumes \(D C \eta\rangle\) have \((\) cl-ecl \(C)=(\) subst-cl \((c l-e c l ~ D) ~ \eta)\)
        unfolding trms-subsumes-def by blast
    from \(\langle\sigma \doteq \vartheta \diamond \eta\rangle\) have subst-cl (cl-ecl \(D) \sigma=\operatorname{subst-cl}(c l-e c l D)(\vartheta \diamond \eta)\)
        using subst-eq-cl by blast
```



```
        using composition-of-substs-cl [of cl-ecl \(D \vartheta \eta\) ] by auto
    from this and \(\langle(c l-e c l C)=(s u b s t-c l(c l-e c l ~ D) \eta)\rangle\)
        〈(ground-clause (cl-ecl C))〉 have ground-clause (subst-cl (cl-ecl D) \(\eta\) )
        by auto
    from this \(\langle D \in S\rangle\left\langle(\right.\) cl-ecl \(D)=\left(\right.\) subst-cl \(\left.\left.C^{\prime} \vartheta\right)\right\rangle\)
        \(\left\langle(\right.\) cl-ecl \(D)=\left(\right.\) subst-cl \(\left.\left.C^{\prime} \vartheta\right)\right\rangle\)
        have redundant-clause (subst-ecl \(D \eta\) ) \(S \eta\) (subst-cl \(C^{\prime} \vartheta\) )
        using self-redundant-clause by metis
    from «derivable \(D P S \vartheta\) FirstOrder \(C^{\prime}\) 〉 have \(P \subseteq S\) unfolding derivable-def
by auto
    from this assms(2) have \(\forall x \in P\). (finite (cl-ecl \(x\) )) by auto
    from this «derivable DPS ७ FirstOrder \(C^{\prime}\) 〉 have finite \(C^{\prime}\)
        using derivable-clauses-are-finite by auto
    from this \(\left\langle(c l-e c l ~ D)=\right.\) subst-cl \(\left.C^{\prime} \vartheta\right\rangle\)
                \(\left\langle r e d u n d a n t-c l a u s e(\right.\) subst-ecl \(D \eta) S \eta\left(\right.\) subst-cl \(\left.\left.C^{\prime} \vartheta\right)\right\rangle\langle(\sigma \doteq \vartheta \diamond \eta)\rangle\)
        have redundant-clause (subst-ecl D \(\eta\) ) \(S \sigma C^{\prime}\)
        using lift-redundant-cl by metis
    from this and \(\left\langle\neg(\right.\) redundant-clause (subst-ecl \(D \eta) S \sigma C^{\prime}\) ) show False by auto
qed
```


### 7.3 Satisfiability of Saturated Sets with No Empty Clause

We are now in the position to prove that the previously constructed interpretation is indeed a model of the set of extended clauses, if the latter is saturated and does not contain an extended clause with empty clausal part. More precisely, the constructed interpretation satisfies the clausal part of every extended clause whose attached set of terms is in normal form. This is the case in particular if this set is empty, hence if the clause is an input clause.
Note that we do not provide any function for explicitly constructing such saturated sets, except by generating all derivable clauses (see below).

```
lemma int-clset-is-a-model:
    assumes ground-inference-saturated \(S\)
    assumes all-finite: \(\forall x \in S\). (finite (cl-ecl \(x)\) )
    assumes Ball \(S\) well-constrained
    assumes all-non-empty: \(\forall x \in S .(\operatorname{cl-ecl} x) \neq\{ \}\)
    assumes closed-under-renaming \(S\)
    shows \(\forall C \sigma\). (fst pair \(=C) \longrightarrow \sigma=(\) snd pair \() \longrightarrow C \in S \longrightarrow\)
        ground-clause (subst-cl (cl-ecl C) \(\sigma\) )
    \(\longrightarrow(\) all-trms-irreducible (subst-set (trms-ecl C) \(\sigma\) ) \((\lambda t\). (trm-rep \(t S))\) )
                            \(\longrightarrow\) validate-ground-clause (same-values \((\lambda t\). (trm-rep \(t S)\) ) (subst-cl (cl-ecl
C) \(\sigma\) )
                (is ?P pair)
proof ((rule wf-induct [of ecl-ord ?P pair]),(simp add: wf-ecl-ord \()\) )
next
```

The proof is by induction and contradiction. We consider a minimal instance that is not true in the interpretation and we derive a contradiction.

```
fix pair assume hyp-ind: \(\forall y .(y, p a i r) \in\) ecl-ord \(\longrightarrow(? P\) y \()\)
let ? \(I=(\) int-clset \(S)\)
have fo-interpretation ?I
unfolding int-clset-def using trm-rep-compatible-with-structure same-values-fo-int
```

    by metis
    show (?P pair)
proof (rule ccontr)
assume $\neg$ (?P pair)
then obtain $C \sigma$ where $C=($ fst pair $)$ and $\sigma=($ snd pair $)$ and $C \in S$
and ground-clause (subst-cl (cl-ecl C) $\sigma$ )
and (all-trms-irreducible (subst-set (trms-ecl C) $\sigma$ )
$(\lambda t .($ trm-rep $t S)))$
and cm: ᄀvalidate-ground-clause (int-clset $S$ ) (subst-cl (cl-ecl C) $\sigma$ )
unfolding int-clset-def by metis

First, we prove that no reduction is possible (otherwise the superposition rule applies).

```
let ?nored \(=\left(\forall L 1\right.\) L2 \(D t s u^{\prime} u v p\) polarity \(\sigma^{\prime}\).
```

$\neg\left(\left(\right.\right.$ reduction L1 $C \sigma^{\prime} t s$ polarity L2 $u u^{\prime} p v D$ ?I $\left.S \sigma\right) \wedge($ variable-disjoint $C D))$ )
have ?nored
proof (rule ccontr)
assume $\neg$ ?nored
then obtain L1 L2 $D t s u^{\prime} u v p$ polarity $\sigma^{\prime}$ where red: reduction L1 $C \sigma^{\prime} t$ s polarity L2 u $u^{\prime} p v D$ ?I $S \sigma$ and variable-disjoint $C D$
by blast
from red have (subst $\left.u \sigma^{\prime}\right) \neq\left(\right.$ subst $\left.v \sigma^{\prime}\right)$ unfolding reduction-def by blast
from red have ground-clause (subst-cl (cl-ecl D) $\sigma^{\prime}$ )
unfolding reduction-def by blast
from red have (coincide-on $\sigma \sigma^{\prime}($ vars-of-cl (cl-ecl C))) unfolding reduction-def by blast
from red have $\neg$ is-a-variable $u^{\prime}$ unfolding reduction-def by blast
from red have $D \in S$ unfolding reduction-def by blast
from red have el1: (eligible-literal L1 $C \sigma^{\prime}$ ) unfolding reduction-def by blast
from red have el2: (eligible-literal L2 $D \sigma^{\prime}$ ) unfolding reduction-def by blast
from red have $D \in S$ unfolding reduction-def by blast
from red have (minimal-redex $p$ (subst $t \sigma$ ) C $S t$ )
unfolding reduction-def by blast
from red have $l 1: L 1 \in($ cl-ecl $C)$ unfolding reduction-def by blast
from red have l2: L2 $\in($ cl-ecl $D)$ unfolding reduction-def by blast
from red have o1: (orient-lit-inst L1 t s polarity $\sigma^{\prime}$ ) unfolding reduction-def by blast
from red have o2: (orient-lit-inst L2 $u v$ pos $\sigma^{\prime}$ ) unfolding reduction-def by blast
from red have $e:\left(\right.$ subst $\left.u^{\prime} \sigma^{\prime}\right)=\left(\right.$ subst $\left.u \sigma^{\prime}\right)$
unfolding reduction-def by blast
from red have ( $\neg$ validate-ground-clause ?I (subst-cl ( (cl-ecl D) - \{ L2 \}
) $\left.\sigma^{\prime}\right)$ )
unfolding reduction-def by blast
from red have $\left(\forall x \in(\right.$ cl-ecl $D)-\{$ L2 $\}$. ( (subst-lit $\left.x \sigma^{\prime}\right)$, (subst-lit L2 $\left.\sigma^{\prime}\right)$ )

$$
\in \text { lit-ord })
$$

unfolding reduction-def by blast
from red have st: (subterm t $p u^{\prime}$ ) unfolding reduction-def by blast from red have (allowed-redex $u^{\prime} C \sigma$ ) unfolding reduction-def by blast
from st have $u^{\prime} \in($ subterms-of $t)$ using occurs-in-def by auto
from this and o1 have $u^{\prime} \in($ subterms-of-lit L1) using orient-lit-inst-subterms by auto
from this and $\langle L 1 \in($ cl-ecl $C)\rangle$ have $u^{\prime} \in($ subterms-of-cl $($ cl-ecl $C))$ by auto
from this and $\left\langle\left(\right.\right.$ allowed-redex $\left.\left.u^{\prime} C \sigma\right)\right\rangle$ and $\langle C \in S\rangle$ and 〈(coincide-on $\sigma \sigma^{\prime}($ vars-of-cl $($ cl-ecl $\left.\left.C))\right)\right\rangle$ assms(3)
have rte: (allowed-redex $u^{\prime} C \sigma^{\prime}$ ) using allowed-redex-coincide [of $u^{\prime} C \sigma$ $\sigma^{\prime}$
by metis
from red have ( (subst-lit L2 $\left.\sigma^{\prime}\right),\left(\right.$ subst-lit L1 $\left.\sigma^{\prime}\right)$ )
$\in$ lit-ord unfolding reduction-def by blast
from red have (all-trms-irreducible (subst-set (trms-ecl D) $\sigma^{\prime}$ )
$(\lambda t$. (trm-rep $t S))$ ) unfolding reduction-def by blast
from red have ?I (subst $u \sigma^{\prime}$ ) (subst $v \sigma^{\prime}$ )
unfolding reduction-def by blast
from $e$ have $t$ : ck-unifier $u^{\prime} u \sigma^{\prime}$ Ground unfolding $c k$-unifier-def Unifier-def by auto
have $\forall x \in($ cl-ecl $D) .\left(\left(\right.\right.$ mset-lit $\left(\right.$ subst-lit $\left.\left.x \sigma^{\prime}\right)\right),\left(\right.$ mset-lit $\left(\right.$ subst-lit L1 $\left.\left.\left.\sigma^{\prime}\right)\right)\right)$
$\in$ (mult trm-ord $)$
proof (rule ccontr)
assume $\neg\left(\forall x \in(\right.$ cl-ecl $D) .\left(\left(\right.\right.$ mset-lit (subst-lit $\left.\left.x \sigma^{\prime}\right)\right),($ mset-lit (subst-lit L1 $\left.\sigma^{\prime}\right)$ ))
$\in($ mult trm-ord $))$
from this obtain $x$ where $x \in($ cl-ecl D)
and $\left(\left(\right.\right.$ mset-lit $\left(\right.$ subst-lit $\left.\left.x \sigma^{\prime}\right)\right),\left(\right.$ mset-lit $\left(\right.$ subst-lit L1 $\left.\left.\left.\sigma^{\prime}\right)\right)\right)$
$\notin$ (mult trm-ord)
by auto
show False
proof (cases)
assume $x=L 2$
from this and $\left\langle\left(\left(\right.\right.\right.$ subst-lit L2 $\left.\sigma^{\prime}\right),\left(\right.$ subst-lit L1 $\left.\left.\sigma^{\prime}\right)\right)$
$\in$ lit-ord $\rangle$ and $\left\langle\left(\left(\right.\right.\right.$ mset-lit $\left(\right.$ subst-lit $\left.\left.x \sigma^{\prime}\right)\right),\left(\right.$ mset-lit $\left(\right.$ subst-lit L1 $\left.\left.\left.\sigma^{\prime}\right)\right)\right)$ $\notin($ mult trm-ord) $)$
show False unfolding lit-ord-def by auto
next
assume $x \neq L 2$
from this and $\langle x \in($ cl-ecl $D)\rangle$ have $x \in($ cl-ecl $D)-\{L 2\}$ by auto
from this and $\left\langle\left(\forall x \in(\right.\right.$ cl-ecl $D)-\{$ L2 $\}$. ( (subst-lit $\left.x \sigma^{\prime}\right),($ subst-lit L2 $\left.\sigma^{\prime}\right)$ )

$$
\begin{gathered}
\in \text { lit-ord })> \\
\text { have }\left(\binom{\text { subst-lit } \left.\left.x \sigma^{\prime}\right),\left(\text { subst-lit L2 } \sigma^{\prime}\right)\right)}{\in \text { lit-ord by auto }} .\right.
\end{gathered}
$$

from $\left\langle\left(\left(\right.\right.\right.$ mset-lit $\left(\right.$ subst-lit $\left.\left.x \sigma^{\prime}\right)\right),\left(\right.$ mset-lit $\left(\right.$ subst-lit L1 $\left.\left.\left.\sigma^{\prime}\right)\right)\right)$
$\notin($ mult trm-ord $)>$
have ((subst-lit $\left.x \sigma^{\prime}\right),\left(\right.$ subst-lit L1 $\left.\sigma^{\prime}\right)$ )
$\notin$ lit-ord
unfolding lit-ord-def by auto
from this and «( (subst-lit x $\left.\sigma^{\prime}\right),\left(\right.$ subst-lit L2 $\left.\left.\sigma^{\prime}\right)\right)$
$\in$ lit-ord>
and $\left\langle\left(\left(\right.\right.\right.$ subst-lit L2 $\left.\sigma^{\prime}\right),\left(\right.$ subst-lit L1 $\left.\left.\sigma^{\prime}\right)\right)$
$\in$ lit-ord $>$
show False using lit-ord-trans unfolding trans-def by blast
qed
qed
from all-finite and $\langle C \in S\rangle$ have finite $($ cl-ecl $C)$ by auto
from this and $\langle L 1 \in($ cl-ecl $C)\rangle$
have (mset-lit (subst-lit L1 $\sigma^{\prime}$ )) $\in \#$ mset-ecl $\left(C, \sigma^{\prime}\right)$
using mset-ecl-and-mset-lit by auto
from this have (mset-lit (subst-lit L1 $\left.\left.\sigma^{\prime}\right)\right) \in\left(\right.$ set-mset $\left(\right.$ mset-ecl $\left.\left.\left(C, \sigma^{\prime}\right)\right)\right)$ by $\operatorname{simp}$
have $\forall x .\left(x \in\left(\right.\right.$ set-mset (mset-ecl $\left.\left.\left(D, \sigma^{\prime}\right)\right)\right)$
$\longrightarrow\left(\exists y \in \operatorname{set}\right.$-mset $\left(\right.$ mset-ecl $\left.\left(C, \sigma^{\prime}\right)\right) .(x, y) \in($ mult trm-ord $\left.\left.)\right)\right)$
proof ((rule allI),(rule impI))
fix $x$ assume $x \in\left(\right.$ set-mset $\left(\right.$ mset-ecl $\left.\left.\left(D, \sigma^{\prime}\right)\right)\right)$
then have $x \in \#$ mset-ecl $\left(D, \sigma^{\prime}\right)$ by simp
from $\left\langle x \in \#\right.$ mset-ecl $\left.\left(D, \sigma^{\prime}\right)\right\rangle$ obtain $x^{\prime}$
where $x^{\prime} \in \#($ mset-set $($ cl-ecl $D))$
and $x=\left(\right.$ mset-lit $\left(\right.$ subst-lit $\left.\left.x^{\prime} \sigma^{\prime}\right)\right)$ by auto
from $\left\langle x^{\prime} \in \#(\right.$ mset-set $($ cl-ecl $\left.D))\right\rangle$ have $x^{\prime} \in(c l-e c l D)$
using count-mset-set(3) by (fastforce simp: count-eq-zero-iff)
from this
and $\langle\forall x \in($ cl-ecl $D)$.
( ( mset-lit (subst-lit $\left.\left.x \sigma^{\prime}\right)\right),\left(\right.$ mset-lit (subst-lit L1 $\left.\left.\sigma^{\prime}\right)\right)$ ) $\in($ mult trm-ord $)\rangle$
and $\left\langle x=\left(\right.\right.$ mset-lit $\left(\right.$ subst-lit $\left.\left.\left.x^{\prime} \sigma^{\prime}\right)\right)\right\rangle$
have $\left(x,\left(\right.\right.$ mset-lit $\left(\right.$ subst-lit L1 $\left.\left.\left.\sigma^{\prime}\right)\right)\right) \in($ mult trm-ord $)$
by auto
from $\left\langle\left(\right.\right.$ mset-lit $\left(\right.$ subst-lit L1 $\left.\left.\sigma^{\prime}\right)\right) \in\left(\right.$ set-mset $\left(\right.$ mset-ecl $\left.\left.\left.\left(C, \sigma^{\prime}\right)\right)\right)\right\rangle$
and $\left\langle\left(x,\left(\right.\right.\right.$ mset-lit $\left(\right.$ subst-lit L1 $\left.\left.\left.\sigma^{\prime}\right)\right)\right) \in($ mult trm-ord $\left.)\right\rangle$
show
$\left(\exists y \in\right.$ set-mset $\left(\right.$ mset-ecl $\left.\left(C, \sigma^{\prime}\right)\right) .(x, y) \in($ mult trm-ord $\left.)\right)$
by auto
qed
from this have
dom: $\wedge I J K . J \neq\{\#\} \wedge(\forall k \in$ set-mset $K . \exists j \in$ set-mset $J .(k, j) \in($ mult trm-ord)) $\longrightarrow$
$(I+K, I+J) \in$ mult (mult trm-ord)
by (blast intro: one-step-implies-mult)
from 〈(mset-lit (subst-lit L1 $\left.\left.\sigma^{\prime}\right)\right) \in \#$ mset-ecl $\left.\left(C, \sigma^{\prime}\right)\right\rangle$
have mset-ecl $\left(C, \sigma^{\prime}\right) \neq\{\#\}$ by auto
from $\left\langle\forall x .\left(x \in\left(\right.\right.\right.$ set-mset $\left(\right.$ mset-ecl $\left.\left.\left(D, \sigma^{\prime}\right)\right)\right)$
$\longrightarrow\left(\exists y \in\right.$ set-mset $\left(\right.$ mset-ecl $\left.\left(C, \sigma^{\prime}\right)\right) .(x, y) \in($ mult trm-ord $\left.\left.\left.)\right)\right)\right\rangle$ and 〈mset-ecl $\left.\left(C, \sigma^{\prime}\right) \neq\{\#\}\right\rangle$
have $\left(\{\#\}+\right.$ mset-ecl $\left(D, \sigma^{\prime}\right),\{\#\}+\operatorname{mset}$-ecl $\left.\left(C, \sigma^{\prime}\right)\right) \in$ mult (mult
trm-ord)
using dom［of（mset－ecl $\left.\left(C, \sigma^{\prime}\right)\right)$ mset－ecl $\left.\left(D, \sigma^{\prime}\right)\{\#\}\right]$ by auto
from this have（mset－ecl $\left(D, \sigma^{\prime}\right)$ ，mset－ecl $\left.\left(C, \sigma^{\prime}\right)\right) \in$ mult（mult trm－ord）
by auto
from this have $\left(\left(D, \sigma^{\prime}\right),\left(C, \sigma^{\prime}\right)\right) \in$ ecl－ord
unfolding ecl－ord－def by auto
from st obtain $t^{\prime}$ where $r t$ ：（replace－subterm $t p v t^{\prime}$ ）
using replace－subterm－is－a－function by blast
from st obtain $R C l-R$ nt－$R L^{\prime} C l-C C l-D$ where
ntr：nt－R $=$（dom－trms Cl－R（subst－set
$(($ trms－ecl $C) \cup($ trms－ecl $D) \cup$
$\{r . \exists q .(q, p) \in($ pos－ord $C t) \wedge($ subterm $\left.\left.t q r)\}) \sigma^{\prime}\right)\right)$
and $r: R=E c l C l-R n t-R$
and clc：$C l-C=(c l-e c l C)$
and cld：$C l-D=(c l-e c l D)$
and clr：$C l-R=\left(\right.$ subst－cl $\left((C l-C-\{L 1\}) \cup\left((C l-D-\{L 2\}) \cup\left\{L^{\prime}\right.\right.\right.$
\} )) $\sigma^{\prime}$ ）
and $l^{\prime}: L^{\prime}=m k$－lit polarity $\left(E q t^{\prime} s\right)$
by auto
from 〈orient－lit－inst L1 t s polarity $\left.\sigma^{\prime}\right\rangle$ have vars－of $t \subseteq$ vars－of－lit L1
using orient－lit－inst－vars by auto
from $\langle L 1 \in($ cl－ecl $C)\rangle$ have vars－of－lit L1 $\subseteq$ vars－of－cl（cl－ecl C）by auto
from this and «vars－of $t \subseteq$ vars－of－lit L1〉 have vars－of $t \subseteq$ vars－of－cl（cl－ecl
C）by auto
from this and 〈coincide－on $\sigma \sigma^{\prime}($ vars－of－cl（cl－ecl C））〉
have coincide－on $\sigma \sigma^{\prime}$（vars－of $t$ ）unfolding coincide－on－def by auto
from this have subst $t \sigma=$ subst $t \sigma^{\prime}$ using coincide－on－term by auto
from $\left\langle\left(\forall x \in(\right.\right.$ cl－ecl $D)-\underset{\in \operatorname{Lit} \text {－ord })\rangle}{\{\operatorname{La}\} .\left(\left(\text { subst－lit } x \sigma^{\prime}\right),\left(\text { subst－lit L2 } \sigma^{\prime}\right)\right)}$
have strictly－maximal－literal D L2 $\sigma^{\prime}$ unfolding strictly－maximal－literal－def by metis
from $n t r$ have $n t-R=$ get－trms $C l-R$（dom－trms $C l-R$（subst－set
$(($ trms－ecl $C) \cup($ trms－ecl $D) \cup$
$\{r . \exists q .(q, p) \in($ pos－ord $C t) \wedge($ subterm $\left.\left.t q r)\}) \sigma^{\prime}\right)\right)$ Ground unfolding get－trms－def by auto
from this $\left\langle\left(\right.\right.$ subst $\left.u \sigma^{\prime}\right) \neq\left(\right.$ subst $\left.\left.v \sigma^{\prime}\right)\right\rangle\left\langle\neg\right.$ is－a－variable $\left.u^{\prime}\right\rangle l 1$ l2 el1 el2
〈variable－disjoint $C$ D rte r o1 o2 t st rt l＇clr ntr clr clc cld $\langle R=E c l \mathrm{Cl}$－$R$ $n t-R>$
$\left\langle\left(\left(\right.\right.\right.$ subst－lit L2 $\left.\sigma^{\prime}\right),\left(\right.$ subst－lit L1 $\left.\left.\sigma^{\prime}\right)\right)$
$\in$ lit－ord＞
〈strictly－maximal－literal DL2 $\sigma^{\prime}$ 〉
have superposition $C D R \sigma^{\prime}$ Ground $((C l-C-\{L 1\}) \cup((C l-D-\{L 2$ \}) $\left.\cup\left\{L^{\prime}\right\}\right)$ ）
unfolding superposition－def by blast
from l2 have（subst－lit L2 $\left.\sigma^{\prime}\right) \in\left(\right.$ subst－cl（cl－ecl D）$\left.\sigma^{\prime}\right)$ by auto
from this and 〈ground－clause（subst－cl（cl－ecl D）$\sigma^{\prime}$ ）〉
have vars－of－lit（subst－lit L2 $\sigma^{\prime}$ ）$=\{ \}$
by auto
from this and o2 have vars－of（subst $v \sigma^{\prime}$ ）$=\{ \}$
unfolding orient－lit－inst－def using vars－of－lit．simps vars－of－eq．simps by
from 〈（coincide－on $\sigma \sigma^{\prime}($ vars－of－cl $($ cl－ecl $\left.C))\right)$ 〉
have $\left(\right.$ subst－cl $($ cl－ecl $\left.C) \sigma^{\prime}\right)=($ subst－cl $(c l-e c l ~ C) ~ \sigma)$
using coincide－on－cl［of $\sigma \sigma^{\prime}($ cl－ecl $\left.C)\right]$ by auto
from this and 〈ground－clause（subst－cl（cl－ecl C）$\sigma$ ）〉
have ground－clause（subst－cl（cl－ecl C）$\sigma^{\prime}$ ）using coincide－on－cl by auto
from $l 1$ have（subst－lit L1 $\left.\sigma^{\prime}\right) \in($ subst－cl（cl－ecl $\left.C) \sigma^{\prime}\right)$ by auto
from this and 〈ground－clause（subst－cl（cl－ecl C）$\sigma^{\prime}$ ）〉
have vars－of－lit（subst－lit L1 $\sigma^{\prime}$ ）$=\{ \}$ by auto
from this and o1 have vars－of（subst $\left.t \sigma^{\prime}\right)=\{ \}$
unfolding orient－lit－inst－def using vars－of－lit．simps vars－of－eq．simps by
force
from 〈vars－of－lit（subst－lit L1 $\sigma^{\prime}$ ）$\left.=\{ \}\right\rangle$
and o1 have vars－of（subst s $\sigma^{\prime}$ ）$=\{ \}$
unfolding orient－lit－inst－def using vars－of－lit．simps vars－of－eq．simps by
force
from 〈vars－of（subst $t \sigma^{\prime}$ ）$=\{ \}$ 〉 and 〈vars－of（subst $\left.v \sigma^{\prime}\right)=\{ \}$ 〉
and $r t$ have vars－of（subst $\left.t^{\prime} \sigma^{\prime}\right)=\{ \}$ using ground－replacement［of $t p$
$\left.v t^{\prime} \sigma^{\prime}\right]$
unfolding ground－term－def by blast
from 〈vars－of（subst $\left.t^{\prime} \sigma^{\prime}\right)=\{ \}$ and 〈vars－of（subst s $\sigma^{\prime}$ ）$\left.=\{ \}\right\rangle$
have vars－of－eq（subst－equation $\left.\left(E q t^{\prime} s\right) \sigma^{\prime}\right)=\{ \}$ by auto
from $l^{\prime}$ have $L^{\prime}=\left(\operatorname{Pos}\left(E q t^{\prime} s\right)\right) \vee L^{\prime}=\left(N e g\left(E q t^{\prime} s\right)\right)$ using $m k$－lit．elims by auto
from this and 〈vars－of－eq（subst－equation $\left.\left(E q t^{\prime} s\right) \sigma^{\prime}\right)=\{ \}>$
have vars－of－lit（subst－lit $\left.L^{\prime} \sigma^{\prime}\right)=\{ \}$ by auto
from $\langle C \in S\rangle$ and $\langle D \in S\rangle$ and
superposition $C D R \sigma^{\prime}$ Ground $((C l-C-\{L 1\}) \cup((C l-D-\{L 2\}) \cup$
$\left.\left\{L^{\prime}\right\}\right)$ ） ，
have derivable $R\{C, D\} S \sigma^{\prime}$ Ground $((C l-C-\{L 1\}) \cup((C l-D-\{$
$\left.L 2\}) \cup\left\{L^{\prime}\right\}\right)$ ）
unfolding derivable－def by auto
have ground－clause $C l-R$
proof（rule ccontr）
assume $\neg$ ground－clause $\mathrm{Cl}-\mathrm{R}$
then have vars－of－cl $C l-R \neq\{ \}$ by auto
then obtain $M$ where $M \in C l-R$ and vars－of－lit $M \neq\{ \}$ by auto
from $\langle M \in C l-R\rangle$ and clr obtain $M^{\prime}$ where $M=\left(\right.$ subst－lit $\left.M^{\prime} \sigma^{\prime}\right)$
and $M^{\prime} \in\left((C l-C-\{L 1\}) \cup\left((C l-D-\{L 2\}) \cup\left\{L^{\prime}\right\}\right)\right)$
by auto
show False
proof（cases）
assume $M^{\prime}=L^{\prime}$
from this and＜vars－of－lit（subst－lit $\left.\left.L^{\prime} \sigma^{\prime}\right)=\{ \}\right\rangle$ and $\langle v a r s-o f-l i t ~ M \neq$
and $\left\langle M=\left(\right.\right.$ subst－lit $\left.\left.M^{\prime} \sigma^{\prime}\right)\right\rangle$ show False by auto
next
assume $M^{\prime} \neq L^{\prime}$
from this and $l 1$ clc cld and $\left\langle M^{\prime} \in(C l-C-\{L 1\}) \cup((C l-D-\{L 2\right.$
\}) $\left.\cup\left\{L^{\prime}\right\}\right)$＞
have $M^{\prime} \in($ cl－ecl $C) \vee M^{\prime} \in($ cl－ecl $D)$
by auto
then show False
proof
assume $M^{\prime} \in($ cl－ecl $C)$
from this and 〈ground－clause（subst－cl（cl－ecl C）$\sigma^{\prime}$ ）〉 have
vars－of－lit（subst－lit $\left.M^{\prime} \sigma^{\prime}\right)=\{ \}$ by auto
from this and $\left\langle M=\left(\right.\right.$ subst－lit $\left.\left.M^{\prime} \sigma^{\prime}\right)\right\rangle$ and
$\langle$ vars－of－lit $M \neq\{ \}\rangle$ show False by auto
next
assume $M^{\prime} \in($ cl－ecl $D)$
from this and 〈ground－clause（subst－cl（cl－ecl D）$\sigma^{\prime}$ ）〉 have
vars－of－lit（subst－lit $\left.M^{\prime} \sigma^{\prime}\right)=\{ \}$ by auto
from this and $\left\langle M=\left(\right.\right.$ subst－lit $\left.\left.M^{\prime} \sigma^{\prime}\right)\right\rangle$ and
$\langle$ vars－of－lit $M \neq\{ \}$ show False by auto
qed
qed
qed
from 〈ground－clause（subst－cl（cl－ecl C）$\sigma^{\prime}$ ）〉 and $\langle g r o u n d-c l a u s e ~(s u b s t-c l$ （cl－ecl D）$\sigma^{\prime}$ ）＞
have grounding－set $\{C, D\} \sigma^{\prime}$ unfolding grounding－set－def by auto
from $\langle g r o u n d-c l a u s e ~ C l-R\rangle$ and $\langle R=E c l C l-R n t-R\rangle$ have ground－clause （cl－ecl $R$ ）by auto
from this and＜derivable $R\{C, D\} S \sigma^{\prime}$ Ground $((C l-C-\{L 1\}) \cup$ $\left.\left((C l-D-\{L 2\}) \cup\left\{L^{\prime}\right\}\right)\right)$＞
and 〈ground－inference－saturated $S\rangle\left\langle\right.$ grounding－set $\left.\{C, D\} \sigma^{\prime}\right\rangle$
have（redundant－inference $R S\{C, D\} \sigma^{\prime}$ ）unfolding ground－inference－saturated－def
by blast
from this obtain $S^{\prime}$ where $S^{\prime} \subseteq($ instances $S$ ）and
（set－entails－clause（clset－instances $\left.S^{\prime}\right)($ cl－ecl $R)$ ）
and order：$\forall x \in S^{\prime} .\left((\operatorname{cl-ecl}(f s t x)\right.$ ，snd $x)$ ，cl－ecl $\left.C, \sigma^{\prime}\right) \in$ cl－ord $\vee$
$\left((\right.$ cl－ecl $(f s t x)$ ，snd $x)$ ，cl－ecl $\left.D, \sigma^{\prime}\right) \in$ cl－ord
and all－normalized－term－included：$\left(\forall x \in S^{\prime}\right.$ ．
（subterms－inclusion（subst－set（trms－ecl（fst x））（snd $x)$ ）
(trms-ecl $R)$ ))
unfolding redundant-inference-def by auto
have all-smaller: $\left(\forall x \in S^{\prime} .(((f s t x),(\right.$ snd $x)),(C, \sigma)) \in$ ecl-ord $)$
proof (rule ccontr)
assume $\neg\left(\forall x \in S^{\prime} .(((\right.$ fst $x),($ snd $x)),(C, \sigma)) \in$ ecl-ord $)$
then obtain $x$ where $x \in S^{\prime}$ and $(((f s t x),($ snd $x)),(C, \sigma)) \notin$ ecl-ord by auto
from order[rule-format, $O F\left\langle x \in S^{\prime}\right\rangle$ ]
have ( $($ cl-ecl $(f s t x)$, snd $x)$, cl-ecl $\left.C, \sigma^{\prime}\right) \in$ cl-ord
proof (elim disjE)
assume $\left((c l-e c l(f s t ~ x)\right.$, snd $x)$, cl-ecl $\left.C, \sigma^{\prime}\right) \in$ cl-ord
thus ?thesis by assumption
next
assume $\left((c l-e c l(f s t)\right.$, snd $x)$, cl-ecl $\left.D, \sigma^{\prime}\right) \in$ cl-ord
thus $\left((\right.$ cl-ecl $(f s t x)$, snd $x)$, cl-ecl $\left.C, \sigma^{\prime}\right) \in$ cl-ord
using $\left\langle\left(\left(D, \sigma^{\prime}\right),\left(C, \sigma^{\prime}\right)\right) \in\right.$ ecl-ord $\rangle[$ unfolded member-ecl-ord-iff]
by (rule cl-ord-trans[THEN transD])
qed
from 〈(coincide-on $\sigma \sigma^{\prime}($ vars-of-cl $($ cl-ecl $\left.\left.C))\right)\right\rangle$
have $\left(\right.$ mset-ecl $\left.\left(C, \sigma^{\prime}\right)\right)=($ mset-ecl $(C, \sigma))$
using ecl-ord-coincide $\left[\right.$ of $\left.\sigma \sigma^{\prime} C\right]$ by auto
with 〈( $($ cl-ecl $(f s t x)$, snd $x)$, cl-ecl $\left.C, \sigma^{\prime}\right) \in c l$-ord $\rangle$
have $(($ mset-ecl $x),($ mset-ecl $(C, \sigma))) \in($ mult (mult trm-ord $))$
by (metis (no-types, lifting) CollectD case-prodD cl-ord-def mset-ecl-conv prod.collapse)
from this and $\prec \neg(((f s t x),($ snd $x)),(C, \sigma)) \in$ ecl-ord $\rangle$ show False unfolding ecl-ord-def by auto
qed
have validate-clause-set ?I (clset-instances $S^{\prime}$ )
proof (rule ccontr)
assume $\neg$ validate-clause-set ?I (clset-instances $S^{\prime}$ )
then obtain $x$ where $x \in\left(\right.$ clset-instances $\left.S^{\prime}\right)$ and $\neg$ validate-clause ?I $x$ using validate-clause-set.simps by blast
from $\left\langle x \in\left(\right.\right.$ clset-instances $\left.\left.S^{\prime}\right)\right\rangle$ obtain pair' where pair' $\in S^{\prime}$
and $x=\left(\right.$ subst-cl $\left(\right.$ cl-ecl $\left(\right.$ fst pair $\left.\left.^{\prime}\right)\right)\left(\right.$ snd pair $\left.\left.{ }^{\prime}\right)\right)$
unfolding clset-instances-def
by auto
from all-smaller and $\left\langle\right.$ pair $\left.{ }^{\prime} \in S^{\prime}\right\rangle$ have $\left(\right.$ pair $\left.^{\prime},(C, \sigma)\right) \in$ ecl-ord
by auto
from this and $\langle C=$ fst pair $\rangle$ and $\langle\sigma=$ snd pair $\rangle$ have (pair ${ }^{\prime}$, pair $) \in$ ecl-ord
by auto
from this and hyp-ind have ?P pair' by blast
from $\left\langle\right.$ pair $\left.{ }^{\prime} \in S^{\prime}\right\rangle$ and all-normalized-term-included have
(subterms-inclusion (subst-set (trms-ecl (fst pair')) (snd pair' $)$ )
（trms－ecl $R$ ））by auto
have $i$ ：（all－trms－irreducible（subst－set（trms－ecl（fst pair＇））（snd pair＇））
$(\lambda t .($ trm－rep $t S)))$
proof（rule ccontr）
assume $\neg\left(\right.$ all－trms－irreducible（subst－set（trms－ecl $\left(\right.$ fst pair $\left.\left.^{\prime}\right)\right)$（snd pair $\left.{ }^{\prime}\right)$ ）
$(\lambda t .($ trm－rep $t S)))$
then obtain $w w^{\prime}$ where $w \in$ subst－set（trms－ecl（fst pair＇））（snd pair＇）
and occurs－in $w^{\prime} w$
trm－rep $w^{\prime} S \neq w^{\prime}$
unfolding all－trms－irreducible－def by blast
from $\langle w \in$ subst－set（trms－ecl（fst pair＇$)$ ）（snd pair＇$)\rangle$ and
$\left\langle\left(\right.\right.$ subterms－inclusion（subst－set（trms－ecl（fst pair $\left.\left.{ }^{\prime}\right)\right)$（snd pair $\left.{ }^{\prime}\right)$ ）
（trms－ecl $R$ ））＞obtain $w^{\prime \prime}$ where $w^{\prime \prime} \in$ trms－ecl $R$ and occurs－in $w w^{\prime \prime}$ unfolding subterms－inclusion－def by auto
from 〈occurs－in $w w^{\prime \prime}$ 〉 and «occurs－in $\left.w^{\prime} w\right\rangle$ have occurs－in $w^{\prime} w^{\prime \prime}$ using occur－in－subterm by blast
from ntr have nt－R $\subseteq$（subst－set $(($ trms－ecl $C) \cup($ trms－ecl $D) \cup$ $\{r . \exists q .(q, p) \in($ pos－ord $C t) \wedge($ subterm $\left.t q r)\}) \sigma^{\prime}\right)$
using dom－trms－subset［of Cl－R（subst－set（（trms－ecl $C) \cup($ trms－ecl $D) \cup$ $\{r . \exists q .(q, p) \in($ pos－ord $C t) \wedge($ subterm $\left.\left.t q r)\}) \sigma^{\prime}\right)\right]$ by blast
from this and $r$ have trms－ecl $R \subseteq$（subst－set（ trms－ecl $C) \cup$（trms－ecl
$\{r . \exists q .(q, p) \in($ pos－ord $C t) \wedge($ subterm $\left.t q r)\}) \sigma^{\prime}\right)$ by auto
from this and $\left\langle w^{\prime \prime} \in\right.$ trms－ecl $\left.R\right\rangle$ have $w^{\prime \prime} \in($ subst－set $(($ trms－ecl $C) \cup($ trms－ecl $D) \cup$
$\{r . \exists q .(q, p) \in($ pos－ord $C t) \wedge($ subterm $\left.t q r)\}) \sigma^{\prime}\right)$ by blast
from this obtain $w^{\prime \prime \prime}$
where $w^{\prime \prime \prime} \in(($ trms－ecl $C) \cup($ trms－ecl $D) \cup$ $\{r . \exists q .(q, p) \in($ pos－ord $C t) \wedge($ subterm $t q r)\})$ and $w^{\prime \prime}=$ subst
$w^{\prime \prime \prime} \sigma^{\prime}$
by auto
from this and 〈occurs－in $\left.w^{\prime} w^{\prime \prime}\right\rangle$ have occurs－in $w^{\prime}\left(\right.$ subst $\left.w^{\prime \prime \prime} \sigma^{\prime}\right)$ by auto
have $\neg\left(w^{\prime \prime \prime} \in\right.$ trms－ecl $\left.C\right)$
proof
assume $w^{\prime \prime \prime} \in$ trms－ecl $C$
from this and «occurs－in $\left.w^{\prime} w^{\prime \prime}\right\rangle$ and $\left\langle w^{\prime \prime}=\right.$ subst $\left.w^{\prime \prime \prime} \sigma^{\prime}\right\rangle$ have occurs－in $w^{\prime}\left(\right.$ subst $\left.w^{\prime \prime \prime} \sigma^{\prime}\right)$ by auto
from $\operatorname{assms}(3)$ and $\langle C \in S\rangle$ and $\left\langle w^{\prime \prime \prime} \in\right.$ trms－ecl $\left.C\right\rangle$ have vars－of $w^{\prime \prime \prime} \subseteq$ vars－of－cl（cl－ecl C）using dom－trm－vars unfolding Ball－def well－constrained－def by blast
from this and＜coincide－on $\sigma \sigma^{\prime}(v a r s-o f-c l(c l-e c l ~ C))$ ）have coincide－on $\sigma \sigma^{\prime}\left(\right.$ vars－of $\left.w^{\prime \prime \prime}\right)$
unfolding coincide－on－def by auto
from this have subst $w^{\prime \prime \prime} \sigma=$ subst $w^{\prime \prime \prime} \sigma^{\prime}$ using coincide－on－term by auto
from this
and＜occurs－in $w^{\prime}\left(\right.$ subst $\left.w^{\prime \prime \prime} \sigma^{\prime}\right)$ 〉
have occurs－in $w^{\prime}\left(\right.$ subst $\left.w^{\prime \prime \prime} \sigma\right)$ by auto
from this and $\left\langle w^{\prime \prime \prime} \in\right.$ trms－ecl $\left.C\right\rangle$
〈（all－trms－irreducible（subst－set（trms－ecl C）$\sigma$ ）
$(\lambda t$ ．（trm－rep $t S))$ ）＞
have trm－rep $w^{\prime} S=w^{\prime}$
unfolding all－trms－irreducible－def using subst－set．simps by blast from this and «trm－rep $w^{\prime} S \neq w^{\prime}$ 〉 show False by blast
qed
have $\neg\left(w^{\prime \prime \prime} \in\right.$ trms－ecl $\left.D\right)$
proof
assume $w^{\prime \prime \prime} \in$ trms－ecl $D$
from this and «occurs－in $\left.w^{\prime} w^{\prime \prime}\right\rangle$ and $\left\langle w^{\prime \prime}=\right.$ subst $\left.w^{\prime \prime \prime} \sigma^{\prime}\right\rangle$ have occurs－in $w^{\prime}\left(\right.$ subst $\left.w^{\prime \prime \prime} \sigma^{\prime}\right)$ by auto
from this and $\left\langle w^{\prime \prime \prime} \in\right.$ trms－ecl $\left.D\right\rangle$〈（all－trms－irreducible（subst－set（trms－ecl D）$\sigma^{\prime}$ ）
（ $\lambda t$ ．（trm－rep $t S)$ ））＞ have trm－rep $w^{\prime} S=w^{\prime}$
unfolding all－trms－irreducible－def using subst－set．simps by blast
from this and «trm－rep $\left.w^{\prime} S \neq w^{\prime}\right\rangle$ show False by blast
qed
from this and
$\left\langle w^{\prime \prime \prime} \in((\right.$ trms－ecl $C) \cup($ trms－ecl $D)$
$\cup\{r . \exists q .(q, p) \in($ pos－ord $C t) \wedge($ subterm $t q r)\})\rangle$
and $\left\langle\neg\left(w^{\prime \prime \prime} \in\right.\right.$ trms－ecl $\left.\left.C\right)\right\rangle$
obtain $q$－$w$ where（subterm $\left.t q-w w^{\prime \prime \prime}\right)$ and $(q-w, p) \in($ pos－ord $C t)$
by auto

```
from 〈subterm \(\left.t q-w w^{\prime \prime \prime}\right\rangle\) have subterm (subst \(\left.t \sigma^{\prime}\right) q-w\left(\right.\) subst \(\left.w^{\prime \prime \prime} \sigma^{\prime}\right)\)
    using substs-preserve-subterms by auto
from 〈occurs-in \(w^{\prime}\) (subst \(w^{\prime \prime \prime} \sigma^{\prime}\) )〉 obtain \(q\)
    where (subterm (subst \(\left.w^{\prime \prime \prime} \sigma^{\prime}\right) q w^{\prime}\) ) unfolding occurs-in-def by blast
    from this and 〈subterm (subst \(t \sigma^{\prime}\) ) q-w (subst \(w^{\prime \prime \prime} \sigma^{\prime}\) )〉
have subterm (subst \(t \sigma^{\prime}\) ) (append \(q\) - \(w q\) ) \(w^{\prime}\) using subterm-of-a-subterm-is- \(a\)-subterm
    by blast
    from this and \(\left\langle(\right.\) subst \(t \sigma)=\left(\right.\) subst \(\left.\left.t \sigma^{\prime}\right)\right\rangle\)
    have subterm (subst \(t \sigma\) ) (append \(q-w q\) ) \(w^{\prime}\) by auto
    from \(\langle(q-w, p) \in(\) pos-ord \(C t)\rangle\) have \(((\) append \(q-w q), p) \in(\) pos-ord \(C t)\)
    using pos-ord-prefix by blast
from this and 〈minimal-redex \(p(\) subst \(t \sigma) C S t\rangle\)
    and «subterm (subst \(t \sigma\) ) (append \(q-w q\) ) \(w^{\prime}\) 〉
    have trm-rep \(w^{\prime} S=w^{\prime}\)
    unfolding minimal-redex-def by blast
from this and \(\left\langle\right.\) trm-rep \(\left.w^{\prime} S \neq w^{\prime}\right\rangle\) show False by blast
qed
from \(\left\langle S^{\prime} \subseteq(\right.\) instances \(\left.S)\right\rangle\) and \(\left\langle\right.\) pair \(\left.{ }^{\prime} \in S^{\prime}\right\rangle\) have
    ii: ground-clause (subst-cl (cl-ecl (fst pair')) (snd pair'))
    unfolding instances-def [of \(S\) ] by fastforce
from \(\left\langle S^{\prime} \subseteq(\right.\) instances \(\left.S)\right\rangle\) and \(\left\langle\right.\) pair \(\left.{ }^{\prime} \in S^{\prime}\right\rangle\) have
```

```
    iii: (fst pair') \inS unfolding instances-def [of S] by fastforce
    from 〈?P pair'〉 and i ii ioi have validate-ground-clause ?I
        (subst-cl (cl-ecl (fst pair')) (snd pair')) unfolding int-clset-def by blast
    from this and <x = (subst-cl (cl-ecl (fst pair'}))(\mathrm{ snd pair}\mp@subsup{}{}{\prime}))
    and 〈\negvalidate-clause ?I x〉 show False
        by (metis ii substs-preserve-ground-clause validate-clause.simps)
    qed
    from this and〈fo-interpretation ?I> and
    <(set-entails-clause (clset-instances S') (cl-ecl R))>
    have validate-clause ?I (cl-ecl R) unfolding set-entails-clause-def by blast
    from this have validate-ground-clause ?I (cl-ecl R)
        by (metis }\langleR=Ecl Cl-R nt-R\rangle\langleground-clause Cl-R>
            cl-ecl.simps substs-preserve-ground-clause validate-clause.simps)
    from this obtain }\mp@subsup{L}{}{\prime\prime}\mathrm{ where }\mp@subsup{L}{}{\prime\prime}\in(\mathrm{ cl-ecl R) and validate-ground-lit ?I L'"
        using validate-ground-clause.simps by blast
    from}\langle\mp@subsup{L}{}{\prime\prime}\in(cl-ecl R)\rangle and <R = Ecl Cl-R nt-R\rangle and
        <Cl-R = (subst-cl ((Cl-C - {L1 }) \cup ((Cl-D - {L2 }) \cup{ L'} )) \sigma')\rangle
        obtain M where m:M\in((Cl-C-{L1 })\cup((Cl-D - {L2 }) \cup{L'
} ))
        and }\mp@subsup{L}{}{\prime\prime}=\mathrm{ subst-lit M 尔 by auto
    have M& cl-ecl C
    proof
        assume M \in cl-ecl C
        from this have vars-of-lit M\subseteqvars-of-cl (cl-ecl C) by auto
        from this and <coincide-on \sigma \sigma' (vars-of-cl (cl-ecl C))>
        have coincide-on \sigma \sigma' (vars-of-lit M) unfolding coincide-on-def by auto
        from this have subst-lit M \sigma = subst-lit M \sigma' using coincide-on-lit by
auto
    from this and }\langle\mp@subsup{L}{}{\prime\prime}=\mathrm{ subst-lit M }\mp@subsup{\sigma}{}{\prime}\rangle\mathrm{ have }\mp@subsup{L}{}{\prime\prime}=\mathrm{ subst-lit M }\sigma\mathrm{ by auto
    from}\langleM\in\mathrm{ cl-ecl C> and <L'' = subst-lit M }\sigma>\mathrm{ and <validate-ground-lit ?I
L'>
    have validate-ground-clause ?I (subst-cl (cl-ecl C) \sigma)
    by (metis (mono-tags, lifting) subst-cl.simps mem-Collect-eq
                validate-ground-clause.simps)
    from this and cm show False by blast
    qed
    have M\not\inCl-D - {L2 }
    proof
        assume M GCl-D - {L2 }
        from this and cld and }\langle\mp@subsup{L}{}{\prime\prime}=\mathrm{ subst-lit M 片〉 and «validate-ground-lit ?I
L'>
            have validate-ground-clause ?I (subst-cl ((cl-ecl D) - { L2 }) \sigma')
                by (metis (mono-tags, lifting) subst-cl.simps mem-Collect-eq vali-
date-ground-clause.simps)
    from this and <\negvalidate-ground-clause?I (subst-cl ((cl-ecl D) - { L2 }
) }\mp@subsup{\sigma}{}{\prime}\mathrm{ )>
            show False by blast
    qed
    have M\not= L'
```


## proof

assume $M=L^{\prime}$
from 〈？ （subst $u \sigma^{\prime}$ ）（subst $v \sigma^{\prime}$ ）〉
and $e$ have ？I（subst $u^{\prime} \sigma^{\prime}$ ）（subst $v \sigma^{\prime}$ ）by metis
from $r t$ and st 〈fo－interpretation ？I 〉〈？I（subst $u^{\prime} \sigma^{\prime}$ ）（subst $\left.\left.v \sigma^{\prime}\right)\right\rangle$
have ？I（subst $t \sigma^{\prime}$ ）（subst $t^{\prime} \sigma^{\prime}$ ）
using replacement－preserves－congruences［of ？I $\left.u^{\prime} \sigma^{\prime} v t p t^{\prime}\right]$
unfolding fo－interpretation－def by metis
from $l 1$ and $c m$ have $\neg\left(\right.$ validate－ground－lit ？I（subst－lit L1 $\left.\sigma^{\prime}\right)$ ）
using 〈subst－cl（cl－ecl C）$\sigma^{\prime}=$ subst－cl $($ cl－ecl $\left.C) \sigma\right\rangle$
$\left\langle\right.$ subst－lit L1 $\sigma^{\prime} \in$ subst－cl $\left(\right.$ cl－ecl C）$\left.\sigma^{\prime}\right\rangle$
validate－ground－clause．simps by blast
from this and 〈？ （subst $t \sigma^{\prime}$ ）（subst $\left.\left.t^{\prime} \sigma^{\prime}\right)\right\rangle$ and 〈fo－interpretation ？I〉
and $l^{\prime}\left\langle\right.$ orient－lit－inst L1 $t$ s polarity $\left.\sigma^{\prime}\right\rangle$
have $\neg$ validate－ground－lit ？I（subst－lit $L^{\prime} \sigma^{\prime}$ ）
using trm－rep－preserves－lit－semantics［of ？I $t \sigma^{\prime} t^{\prime} L 1$ s polarity $\sigma^{\prime}$ ］by metis
from this and $\left\langle M=L^{\prime}\right\rangle$ and $\left\langle\right.$ validate－ground－lit ？$\left.L^{\prime \prime}\right\rangle$ and $\left\langle L^{\prime \prime}=\right.$ subst－lit $M \sigma^{\prime}$＞
show False by blast
qed
from this and $\langle M \notin C l-D-\{L 2\}\rangle\langle M \notin($ cl－ecl $C)\rangle$ and $m$ clc show False by auto
qed
Second，we show that the clause contains no contradictory literal（otherwise the reflexion rule applies）．

```
let ?no-cont \(=\forall L t s .(L \in(\) cl-ecl \(C)) \longrightarrow(\) eligible-literal \(L C \sigma)\)
    \(\longrightarrow(\) orient-lit-inst \(L t\) s neg \(\sigma) \longrightarrow(\) trm-rep \((\) subst \(t \sigma) S)=(\) subst \(t \sigma)\)
    \(\longrightarrow(\) subst \(t \sigma) \neq(\) subst \(s \sigma)\)
    have ?no-cont
    proof (rule ccontr)
    assume \(\neg\) ?no-cont
    then obtain \(L t s\) where \(l: L \in(\) cl-ecl \(C)\) and \(e:(\) eligible-literal \(L C \sigma)\)
        and \(o\) : orient-lit-inst \(L t\) s neg \(\sigma\)
        and (trm-rep (subst \(t \sigma) S)=(\) subst \(t \sigma)\)
        and \((\) subst \(t \sigma)=(\) subst \(s \sigma)\) by blast
    from 〈(subst \(t \sigma)=(\) subst \(s \sigma)\rangle\)
        have \(t\) : ck-unifier ts \(\sigma\) Ground unfolding ck-unifier-def Unifier-def by
auto
    from \(l\) and \(e\) and \(o\) and \(t\) obtain \(R C l-R n t-R\) where
        \(R=E c l C l-R n t-R\) and \(C l-R=(\) subst-cl \(((c l-e c l C)-\{L\}) \sigma)\) and
        reflexion \(C R \sigma\) Ground \(((\) cl-ecl \(C)-\{L\})\)
        and trms-ecl \(R=(\) dom-trms \((\) cl-ecl \(R)(\) subst-set \(((\) trms-ecl \(C) \cup\{t\})\)
        unfolding reflexion-def get-trms-def
        by fastforce
    from \(\langle C \in S\rangle\) and \(\langle\) reflexion \(C R \sigma\) Ground \(((\) cl-ecl \(C)-\{L\})\rangle\)
    have derivable \(R\{C\} S \sigma\) Ground \(((\) cl-ecl \(C)-\{L\})\)
```

$\sigma)$ )
unfolding derivable－def by auto
from 〈ground－clause（subst－cl（cl－ecl C）$\sigma$ ）〉
and $\langle C l-R=($ subst－cl $(($ cl－ecl $C)-\{L\}) \sigma)\rangle$
have ground－clause Cl－R using ground－clause．simps by auto
from this and $\langle R=E c l C l-R$ nt－$R\rangle$ have ground－clause（cl－ecl $R$ ）by auto
from 〈ground－clause（subst－cl（cl－ecl C）$\sigma$ ）〉
have grounding－set $\{C\} \sigma$ unfolding grounding－set－def by auto
from this and 〈ground－clause（cl－ecl $R$ ）〉
〈derivable $R\{C\} S \sigma$ Ground $(($ cl－ecl $C)-\{L\})\rangle$ and $<$ ground－inference－saturated
have（redundant－inference $R S\{C\} \sigma$ ）unfolding ground－inference－saturated－def
by blast
from this obtain $S^{\prime}$ where $S^{\prime} \subseteq($ instances $S)$ and
（set－entails－clause（clset－instances $\left.S^{\prime}\right)($ cl－ecl $R)$ ）
and all－smaller：$\forall x \in S^{\prime} .(($ cl－ecl $(f s t x)$ ，snd $x)$ ，cl－ecl $C, \sigma) \in$ cl－ord
and all－normalized－term－included：$\left(\forall x \in S^{\prime}\right.$ ．
（subterms－inclusion（subst－set（trms－ecl（fst x））（snd x）） （trms－ecl $R)$ ））
unfolding redundant－inference－def by auto
have validate－clause－set ？I（clset－instances $S^{\prime}$ ）
proof（rule ccontr）
assume $\neg$ validate－clause－set ？I（clset－instances $S^{\prime}$ ）
then obtain $x$ where $x \in\left(\right.$ clset－instances $\left.S^{\prime}\right)$ and $\neg$ validate－clause ？I $x$ using validate－clause－set．simps by blast
from $\left\langle x \in\left(\right.\right.$ clset－instances $\left.\left.S^{\prime}\right)\right\rangle$ obtain pair＇where pair＇$\in S^{\prime}$
and $x=\left(\right.$ subst－cl（cl－ecl（fst pair $\left.\left.{ }^{\prime}\right)\right)\left(\right.$ snd pair $\left.\left.{ }^{\prime}\right)\right)$
unfolding clset－instances－def
by auto
from all－smaller and $\left\langle\right.$ pair $\left.{ }^{\prime} \in S^{\prime}\right\rangle$ have $\left(\right.$ pair $\left.^{\prime},(C, \sigma)\right) \in$ ecl－ord
by（metis member－ecl－ord－iff prod．collapse）
from this and $\langle C=$ fst pair $\rangle$ and $\langle\sigma=$ snd pair $\rangle$ have（pair ${ }^{\prime}$, pair $) \in$ ecl－ord
by auto
from this and hyp－ind have ？P pair＇by blast
from $\langle$ trms－ecl $R=($ dom－trms $($ cl－ecl $R)($ subst－set $(($ trms－ecl $C) \cup\{t\})$
$\sigma)$ ）＞
have trms－ecl $R \subseteq($ subst－set $((t r m s$－ecl $C) \cup\{t\}) \sigma)$
using dom－trms－subset by metis
from $\left\langle\right.$ pair $\left.{ }^{\prime} \in S^{\prime}\right\rangle$ and all－normalized－term－included have
（subterms－inclusion（subst－set（trms－ecl（fst pair＇$)$ ）（snd pair $\left.{ }^{\prime}\right)$ ）
（trms－ecl $R$ ））by auto
have $i$ ：（all－trms－irreducible（subst－set（trms－ecl（fst pair＇））（snd pair＇$)$ ）
（ $\lambda t$ ．（trm－rep $t S))$ ）
proof（rule ccontr）
assume $\neg$（all－trms－irreducible（subst－set（trms－ecl（fst pair $\left.{ }^{\prime}\right)$ ）（snd pair $\left.{ }^{\prime}\right)$ ） $(\lambda t$ ．（trm－rep $t S))$ ）
then obtain $t^{\prime} t^{\prime \prime}$ where $t^{\prime} \in($ subst－set（trms－ecl（fst pair $\left.)\right)\left(\right.$ snd pair $\left.\left.{ }^{\prime}\right)\right)$

```
    occurs-in t'\prime t' and trm-rep t' }S\not=\mp@subsup{t}{}{\prime\prime}\mathrm{ unfolding all-trms-irreducible-def
by blast
    from <t' }\in(\mathrm{ subst-set (trms-ecl (fst pair}\mp@subsup{}{}{\prime}))(\mathrm{ snd pair}\mp@subsup{}{}{\prime}))
            and<(subterms-inclusion (subst-set (trms-ecl (fst pair')) (snd pair'))
(trms-ecl R))>
    obtain s' where s'\in(trms-ecl R) and occurs-in t' s'
    unfolding subterms-inclusion-def by auto
    from <s'\in (trms-ecl R)\rangle and <trms-ecl R\subseteq(subst-set ((trms-ecl C) \cup{
t }) \sigma)>
    obtain s"}\mathrm{ where s'= subst s'|}\sigma\mathrm{ and }\mp@subsup{s}{}{\prime\prime}\in((trms-ecl C)\cup{t}) by
auto
    from}<\mp@subsup{s}{}{\prime\prime}\in((trms-ecl C)\cup{t})> have s"'\intrms-ecl C\vee ''\prime = t by
auto
    thus False
    proof
        assume s" }\in\mathrm{ trms-ecl C
        from this and \langles' = subst s" }\sigma\rangle\mathrm{ have }\mp@subsup{s}{}{\prime}\in(\mathrm{ subst-set (trms-ecl C) }\sigma\mathrm{ ) by
auto
    from this and <(all-trms-irreducible (subst-set (trms-ecl C) \sigma)
                    (\lambdat. (trm-rep t S)))> and <occurs-in t' s'> have trm-rep t'S = t'
            unfolding all-trms-irreducible-def by blast
        from this and <occurs-in t'\prime t'〉 and <trm-rep t" S\not= t'`show False
            using occurs-in-def subts-of-irred-trms-are-irred by blast
    next
        assume s"}=
        from this and }\langle\mp@subsup{s}{}{\prime}=\mathrm{ subst }\mp@subsup{s}{}{\prime\prime}\sigma\rangle\mathrm{ have }\mp@subsup{s}{}{\prime}=\mathrm{ subst t }\sigma\mathrm{ by auto
        from this and «(trm-rep (subst t \sigma) S)=(subst t \sigma)>
            have trm-rep s'}S=\mp@subsup{s}{}{\prime}\mathrm{ by blast
            from〈trm-rep s'S= s'\rangle\langletrm-rep t'\prime}S\not=\mp@subsup{t}{}{\prime\prime}\rangle\langleoccurs-in t' s'\rangle\langleoccurs-in
t'\prime t'>
            show False using occurs-in-def subts-of-irred-trms-are-irred by blast
        qed
    qed
    from \langleS'\subseteq(instances S)\rangle and <pair ' \in S'> have
        ii: ground-clause (subst-cl (cl-ecl (fst pair')) (snd pair'))
        unfolding instances-def [of S] by fastforce
        from \langleS'\subseteq(\mathrm{ instances S) > and <pair' }\in\mp@subsup{S}{}{\prime}\rangle have
        iii:(fst pair') \inS unfolding instances-def [of S] by fastforce
        from 〈?P pair'〉 and i ii iii have validate-ground-clause ?I
        (subst-cl (cl-ecl (fst pair')) (snd pair')) unfolding int-clset-def by blast
        from this and <x = (subst-cl (cl-ecl (fst pair')) (snd pair'))>
            and 〈\negvalidate-clause ?I x\rangle show False
            by (metis ii substs-preserve-ground-clause validate-clause.simps)
    qed
    from this and〈fo-interpretation ?I> and
    <(set-entails-clause (clset-instances S') (cl-ecl R))〉
    have validate-clause ?I (cl-ecl R) unfolding set-entails-clause-def by blast
    from this have validate-ground-clause ?I (cl-ecl R)
        by (metis <R = Ecl Cl-R nt-R\rangle\langleground-clause Cl-R>
```

cl-ecl.simps substs-preserve-ground-clause validate-clause.simps)
from this obtain $L^{\prime}$ where $L^{\prime} \in(c l-e c l ~ R)$ and validate-ground-lit ?I $L^{\prime}$ using validate-ground-clause.simps by blast
from $\left\langle L^{\prime} \in(\right.$ cl-ecl $\left.R)\right\rangle$ and $\langle R=E c l C l-R n t-R\rangle$ and $\langle C l-R=($ subst-cl $(($ cl-ecl $C)-\{L\}) \sigma)\rangle$
obtain $L^{\prime \prime}$ where $L^{\prime \prime} \in$ cl-ecl $C$ and $L^{\prime}=$ subst-lit $L^{\prime \prime} \sigma$
by auto
from $\left\langle L^{\prime \prime} \in\right.$ cl-ecl $\left.C\right\rangle$ and $\left\langle L^{\prime}=\right.$ subst-lit $\left.L^{\prime \prime} \sigma\right\rangle$ and $\langle v a l i d a t e-g r o u n d-l i t ? I$ $L^{\prime}$ ’
have validate-ground-clause ?I (subst-cl (cl-ecl C) $\sigma$ )
by (metis (mono-tags, lifting) subst-cl.simps mem-Collect-eq validate-ground-clause.simps)
from this and cm show False unfolding int-clset-def by blast qed

Third, we prove that the clause contains no pair of equations with the same left-hand side and equivalent right-hand sides (otherwise the factorization rule applies and a smaller false clause is derived).
let ?no-fact $=\forall$ L1 L2 tsuv. $($ L1 $\in($ cl-ecl $C)) \longrightarrow($ eligible-literal L1 C $\sigma$ )
$\longrightarrow(L 2 \in($ cl-ecl $C)-\{L 1\}) \longrightarrow($ orient-lit-inst L1 t s pos $\sigma)$
$\longrightarrow($ orient-lit-inst L2 $u$ v pos $\sigma) \longrightarrow($ subst $t \sigma)=($ subst $u \sigma)$
$\longrightarrow(\neg($ proper-subterm-red $t S \sigma))$
$\longrightarrow($ trm-rep $(($ subst s) $\sigma) S) \neq($ trm-rep $(($ subst $v) \sigma) S)$
have ?no-fact
proof (rule ccontr)
assume $\neg$ ?no-fact
then obtain L1 L2 ts $u v$ where l1: L1 $\in($ cl-ecl $C)$ and l2: L2 $\in(c l-e c l$ C) $-\{L 1\}$
and e1: (eligible-literal L1 $C \sigma$ ) and o1: (orient-lit-inst L1 ts pos $\sigma$ )
and o2: (orient-lit-inst L2 $u$ v pos $\sigma$ ) and $e:($ subst $t \sigma)=($ subst $u \sigma)$
and $(\neg($ proper-subterm-red $t S \sigma))$
and $i:($ trm-rep $(($ subst s) $\sigma) S)=($ trm-rep $(($ subst v) $\sigma) S)$
by blast
from $e$ have $t$ :ck-unifier t $u \sigma$ Ground unfolding ck-unifier-def Unifier-def
using inferences.distinct by metis
from $\langle L 1 \in($ cl-ecl $C)\rangle$ o1 $\langle\neg($ validate-ground-clause $($ int-clset $S)$ (subst-cl $(c l-e c l C) \sigma)$ ) have trm-rep (subst $t \sigma$ ) $S \neq$ trm-rep (subst s $\sigma$ ) $S$
using no-valid-literal by metis
then have subst $t \sigma \neq$ subst $s \sigma$ by metis
from $\langle L 2 \in($ cl-ecl $C)-\{L 1\}\rangle$ have $L 2 \in($ cl-ecl $C)$ by auto
from this o2 $\downarrow \neg($ validate-ground-clause $($ int-clset $S)($ subst-cl $($ cl-ecl $C) \sigma))$ 〉 have trm-rep (subst $u \sigma$ ) $S \neq$ trm-rep (subst $v \sigma$ ) $S$
using no-valid-literal by metis
from this and $e$ have subst $t \sigma \neq$ subst $v \sigma$ by metis
obtain $R C l-R$ nt- $R L^{\prime}$ where
$n t r: n t-R=($ dom－trms $C l-R($ subst－set $((t r m s-e c l ~ C) \cup($ proper－subterms－of t））$\sigma$ ）
and $r: R=E c l C l-R n t-R$
and clr：$C l-R=\left(\right.$ subst－cl $\left.\left(((c l-e c l C)-\{L 2\}) \cup\left\{L^{\prime}\right\}\right) \sigma\right)$
and $l^{\prime}: L^{\prime}=N e g(E q s v)$ by auto
from ntr $l^{\prime}$ clr l1 l2 o1 o2 e1 $t$
〈subst $t \sigma \neq$ subst $s$ $\sigma$ 〈subst $t \sigma \neq$ subst $v \sigma\rangle$
have factorization $C R \sigma$ Ground $\left(((\right.$ cl－ecl $\left.C)-\{L 2\}) \cup\left\{L^{\prime}\right\}\right)$
unfolding factorization－def get－trms－def using inferences．distinct
by（metis cl－ecl．simps）
from l2 have（subst－lit L2 $\sigma$ ）$\in($ subst－cl（cl－ecl C）$\sigma$ ）by auto
from this and 〈ground－clause（subst－cl（cl－ecl C）$\sigma$ ）〉
have vars－of－lit（subst－lit L2 $\sigma$ ）$=\{ \}$
by auto
from this and o2 have vars－of（subst $v \sigma$ ）$=\{ \}$
unfolding orient－lit－inst－def using vars－of－lit．simps vars－of－eq．simps by
from 11 have（subst－lit L1 $\sigma$ ）$\in($ subst－cl（cl－ecl C）$\sigma$ ）by auto
from this and 〈ground－clause（subst－cl（cl－ecl C）$\sigma$ ）〉
have vars－of－lit（subst－lit L1 $\sigma$ ）$=\{ \}$ by auto
from this and o1 have vars－of（subst s $\sigma$ ）$=\{ \}$
unfolding orient－lit－inst－def using vars－of－lit．simps vars－of－eq．simps by
force
from «vars－of（subst v $\sigma$ ）$=\{ \}$ 〉 and 〈vars－of（subst s $\sigma$ ）$=\{ \}\rangle$
and $l^{\prime}$ have vars－of－lit（subst－lit $L^{\prime} \sigma$ ）$=\{ \}$ by auto
from $\langle C \in S\rangle$ and $\langle$ factorization $C R \sigma$ Ground $((($ cl－ecl $C)-\{L 2\}) \cup$ $\left\{L^{\prime}\right\}$ ）＞
have derivable $R\{C\} S \sigma$ Ground $\left(((\right.$ cl－ecl $\left.C)-\{L 2\}) \cup\left\{L^{\prime}\right\}\right)$
unfolding derivable－def by auto
have ground－clause Cl －$R$
proof（rule ccontr）
assume $\neg$ ground－clause $\mathrm{Cl}-\mathrm{R}$
then have vars－of－cl $C l-R \neq\{ \}$ by auto
then obtain $M$ where $M \in C l-R$ and vars－of－lit $M \neq\{ \}$ by auto
from $\langle M \in C l-R\rangle$ and $c l r$ obtain $M^{\prime}$ where $M=\left(\right.$ subst－lit $\left.M^{\prime} \sigma\right)$
and $M^{\prime} \in(($ cl－ecl $C)-\{L 2, L 1\}) \cup\left\{L^{\prime}, L 1\right\}$
by auto
show False
proof（cases）
assume $M^{\prime}=L^{\prime}$
from this and 〈vars－of－lit（subst－lit $\left.\left.L^{\prime} \sigma\right)=\{ \}\right\rangle$ and «vars－of－lit $M \neq$
and $\left\langle M=\left(\right.\right.$ subst-lit $\left.\left.M^{\prime} \sigma\right)\right\rangle$ show False by auto
next
assume $M^{\prime} \neq L^{\prime}$
from this and $l 1$ and $\left\langle M^{\prime} \in((\right.$ cl-ecl $\left.C)-\{L 2, L 1\}) \cup\left\{L^{\prime}, L 1\right\}\right\rangle$
have $M^{\prime} \in($ cl－ecl $C)$ by auto
from this and 〈ground－clause（subst－cl（cl－ecl C）$\sigma$ ）〉 have vars－of－lit（subst－lit $M^{\prime} \sigma$ ）$=\{ \}$ by auto
from this and $\left\langle M=\left(\right.\right.$ subst－lit $\left.\left.M^{\prime} \sigma\right)\right\rangle$ and〈vars－of－lit $M \neq\{ \}$ show False by auto
qed
qed
from $\langle g r o u n d-c l a u s e ~ C l-R\rangle$ and $\langle R=E c l C l-R n t-R\rangle$ have ground－clause
（cl－ecl $R$ ）by auto
from 〈ground－clause（subst－cl（cl－ecl C）$\sigma$ ）〉
have grounding－set $\{C\} \sigma$ unfolding grounding－set－def by auto
from this 〈ground－clause（cl－ecl $R$ ）〉
and «derivable $R\{C\} S \sigma$ Ground $\left(((\right.$ cl－ecl $\left.\left.C)-\{L 2\}) \cup\left\{L^{\prime}\right\}\right)\right\rangle$
and 〈ground－inference－saturated $S$ 〉
have（redundant－inference $R S\{C\} \sigma$ ）unfolding ground－inference－saturated－def
by blast
from this obtain $S^{\prime}$ where $S^{\prime} \subseteq($ instances $S$ ）and
（set－entails－clause（clset－instances $\left.S^{\prime}\right)($ cl－ecl $R)$ ）
and all－smaller：$\forall x \in S^{\prime}$ ．（ $($ cl－ecl $(f s t x)$ ，snd $x)$ ，cl－ecl $\left.C, \sigma\right) \in$ cl－ord
and all－normalized－term－included：$\left(\forall x \in S^{\prime}\right.$ ．
（subterms－inclusion（subst－set（trms－ecl（fst x））（snd x）） （trms－ecl $R$ ）））
unfolding redundant－inference－def by auto
have validate－clause－set ？I（clset－instances $S^{\prime}$ ）
proof（rule ccontr）
assume $\neg$ validate－clause－set ？I（clset－instances $S^{\prime}$ ）
then obtain $x$ where $x \in\left(\right.$ clset－instances $\left.S^{\prime}\right)$ and $\neg$ validate－clause ？$I x$ using validate－clause－set．simps by blast
from $\left\langle x \in\left(\right.\right.$ clset－instances $\left.\left.S^{\prime}\right)\right\rangle$ obtain pair＇where pair＇$\in S^{\prime}$
and $x=\left(\right.$ subst－cl $\left(\right.$ cl－ecl $\left(\right.$ fst pair $\left.\left.^{\prime}\right)\right)($ snd pair $\left.')\right)$
unfolding clset－instances－def
by auto
from all－smaller and $\left\langle\right.$ pair $\left.{ }^{\prime} \in S^{\prime}\right\rangle$ have $\left(\right.$ pair $\left.^{\prime},(C, \sigma)\right) \in$ ecl－ord
by（metis member－ecl－ord－iff prod．collapse）
from this and $\langle C=$ fst pair $\rangle$ and $\left\langle\sigma=\right.$ snd pair〉 have（pair ${ }^{\prime}$, pair $) \in$
ecl－ord
by auto
from this and hyp－ind have ？P pair＇by blast
from $r$ ntr have trms－ecl $R=($ dom－trms（cl－ecl $R)$
$($ subst－set $(($ trms－ecl $C) \cup($ proper－subterms－of $t)) \sigma))$
by auto
from this have trms－ecl $R \subseteq$（subst－set（（trms－ecl $C) \cup$（proper－subterms－of t））$\sigma$ ）
using dom－trms－subset by metis
from $\left\langle\right.$ pair $\left.{ }^{\prime} \in S^{\prime}\right\rangle$ and all-normalized-term-included have
(subterms-inclusion (subst-set (trms-ecl (fst pair')) (snd pair' $\left.{ }^{\prime}\right)$
(trms-ecl $R$ )) by auto
have $i$ : (all-trms-irreducible (subst-set (trms-ecl (fst pair$\left.\left.{ }^{\prime}\right)\right)\left(\right.$ snd pair $\left.{ }^{\prime}\right)$ )
$(\lambda t .($ trm-rep $t S)))$
proof (rule ccontr)
assume $\neg$ (all-trms-irreducible (subst-set (trms-ecl (fst pair $\left.{ }^{\prime}\right)$ ) (snd pair $\left.{ }^{\prime}\right)$ ) ( $\lambda t$. (trm-rep $t S))$ )
then obtain $t^{\prime} t^{\prime \prime}$ where $t^{\prime} \in($ subst-set (trms-ecl (fst pair $)$ ) (snd pair $\left.{ }^{\prime}\right)$ )
occurs-in $t^{\prime \prime} t^{\prime}$ and trm-rep $t^{\prime \prime} S \neq t^{\prime \prime}$ unfolding all-trms-irreducible-def by blast
from $\left\langle t^{\prime} \in\left(\right.\right.$ subst-set $\left(\right.$ trms-ecl $\left(\right.$ fst pair $\left.\left.{ }^{\prime}\right)\right)\left(\right.$ snd pair $\left.\left.\left.{ }^{\prime}\right)\right)\right\rangle$
and «(subterms-inclusion (subst-set (trms-ecl (fst pair')) (snd pair'))
(trms-ecl R))>
obtain $s^{\prime}$ where $s^{\prime} \in($ trms-ecl $R)$ and occurs-in $t^{\prime} s^{\prime}$
unfolding subterms-inclusion-def by auto
from $\left\langle s^{\prime} \in(\right.$ trms-ecl $\left.R)\right\rangle$ and $\langle$ trms-ecl $R \subseteq$ (subst-set $(($ trms-ecl $C) \cup$ (proper-subterms-of $t$ )) $\sigma$ ) >
have $s^{\prime} \in($ subst-set $(($ trms-ecl $C) \cup($ proper-subterms-of $t)) \sigma)$ by auto
then obtain $s^{\prime \prime}$
where $s^{\prime}=$ subst $s^{\prime \prime} \sigma$ and $s^{\prime \prime} \in(($ trms-ecl $C) \cup($ proper-subterms-of $t))$ by auto
from $\left\langle s^{\prime \prime} \in((\right.$ trms-ecl $C) \cup($ proper-subterms-of $\left.t))\right\rangle$ have $s^{\prime \prime} \in$ trms-ecl C
$\vee s^{\prime \prime} \in($ proper-subterms-of $t)$ by auto
thus False
proof
assume $s^{\prime \prime} \in$ trms-ecl $C$
from this and $\left\langle s^{\prime}=\right.$ subst $\left.s^{\prime \prime} \sigma\right\rangle$ have $s^{\prime} \in($ subst-set $($ trms-ecl $C) \sigma$ ) by auto
from this and $\langle($ all-trms-irreducible (subst-set (trms-ecl C) $\sigma$ )
$(\lambda t$. $($ trm-rep $t S))$ ) > and $\left\langle\right.$ occurs-in $\left.t^{\prime} s^{\prime}\right\rangle$ have trm-rep $t^{\prime} S=t^{\prime}$ unfolding all-trms-irreducible-def by blast
from this and $\left\langle o c c u r s\right.$-in $\left.t^{\prime \prime} t^{\prime}\right\rangle$ and $\left\langle\right.$ trm-rep $\left.t^{\prime \prime} S \neq t^{\prime \prime}\right\rangle$ show False using occurs-in-def subts-of-irred-trms-are-irred by blast
next
assume $s^{\prime \prime} \in($ proper-subterms-of $t)$
from 〈occurs-in $\left.t^{\prime} s^{\prime}\right\rangle\left\langle o c c u r s-i n t^{\prime \prime} t^{\prime}\right\rangle\left\langle s^{\prime}=s^{\prime \prime} \triangleleft \sigma\right\rangle\left\langle\right.$ trm-rep $\left.t^{\prime \prime} S \neq t^{\prime \prime}\right\rangle$ have trm-rep (subst $\left.s^{\prime \prime} \sigma\right) S \neq\left(\right.$ subst $\left.s^{\prime \prime} \sigma\right)$
using occurs-in-def subts-of-irred-trms-are-irred by blast
from this and $\left\langle s^{\prime \prime} \in\right.$ (proper-subterms-of $t$ ) and $\langle\neg$ (proper-subterm-red $t S \sigma)>$
show False using proper-subterm-red-def proper-subterms-of.simps by
blast
qed
qed
from $\left\langle S^{\prime} \subseteq(\right.$ instances $\left.S)\right\rangle$ and $\left\langle\right.$ pair $\left.{ }^{\prime} \in S^{\prime}\right\rangle$ have

```
        ii: ground-clause (subst-cl (cl-ecl (fst pair \(\left.{ }^{\prime}\right)\) ) (snd pair \(\left.{ }^{\prime}\right)\) )
        unfolding instances-def [of \(S\) ] by fastforce
        from \(\left\langle S^{\prime} \subseteq(\right.\) instances \(\left.S)\right\rangle\) and \(\left\langle\right.\) pair \(\left.{ }^{\prime} \in S^{\prime}\right\rangle\) have
        iii: \(\left(\right.\) fst pair \(\left.{ }^{\prime}\right) \in S\) unfolding instances-def \([\) of \(S]\) by fastforce
    from 〈? P pair'〉 and \(i\) ii iii have validate-ground-clause ?I
        (subst-cl (cl-ecl (fst pair')) (snd pair'))
        unfolding int-clset-def by blast
    from this and \(\left\langle x=\left(\right.\right.\) subst-cl \(\left(\right.\) cl-ecl \(\left(\right.\) fst pair \(\left.\left.{ }^{\prime}\right)\right)\left(\right.\) snd pair \(\left.\left.\left.{ }^{\prime}\right)\right)\right\rangle\)
        and 〈حvalidate-clause ? \(1 x\rangle\) show False
        by (metis ii substs-preserve-ground-clause validate-clause.simps)
    qed
    from this and 〈fo-interpretation ?I〉 and
    «(set-entails-clause (clset-instances \(\left.S^{\prime}\right)(\) cl-ecl \(\left.R)\right)\) 〉
    have validate-clause ?I (cl-ecl \(R\) ) unfolding set-entails-clause-def by blast
    from this have validate-ground-clause ?I (cl-ecl R)
        by (metis \(\langle R=\) Ecl Cl-R nt-R〉〈ground-clause \(C l-R\rangle\)
            cl-ecl.simps substs-preserve-ground-clause validate-clause.simps)
    from this obtain \(L^{\prime \prime}\) where \(L^{\prime \prime} \in(\) cl-ecl \(R)\) and validate-ground-lit ?I \(L^{\prime \prime}\)
        using validate-ground-clause.simps by blast
    from \(\left\langle L^{\prime \prime} \in(\right.\) cl-ecl \(\left.R)\right\rangle\) and \(\langle R=E c l C l-R n t-R\rangle\) and
        \(\left\langle C l-R=\left(\right.\right.\) subst-cl \(\left(((\right.\) cl-ecl \(\left.\left.\left.C)-\{L 2\}) \cup\left\{L^{\prime}\right\}\right) \sigma\right)\right\rangle\)
        obtain \(M\) where \(m: M \in\left(((\right.\) cl-ecl \(\left.C)-\{L 2, L 1\}) \cup\left\{L^{\prime}, L 1\right\}\right)\)
        and \(L^{\prime \prime}=\) subst-lit \(M \sigma\)
        by auto
    have \(M \in\) cl-ecl \(C\)
    proof (rule ccontr)
        assume \(M \notin c l\)-ecl \(C\)
        from this and \(m\) and \(l 1\) have \(M=L^{\prime}\) by auto
        from this and \(\left\langle L^{\prime \prime}=\right.\) subst-lit \(\left.M \sigma\right\rangle\) and \(\left\langle L^{\prime}=(N e g(E q s v))\right\rangle\)
            have \(L^{\prime \prime}=(\operatorname{Neg}(E q(\) subst \(s \sigma)(\) subst \(v \sigma)))\) by auto
        from this and <validate-ground-lit?I \(\left.L^{\prime \prime}\right\rangle\)
            have \(\neg(? I\) (subst \(s \sigma)(\) subst \(v \sigma)\) )
                using validate-ground-lit.simps(2) validate-ground-eq.simps by metis
                from this and \(i\) show False unfolding same-values-def int-clset-def by
blast
            qed
            from \(\langle M \in\) cl-ecl \(C\rangle\) and \(\left\langle L^{\prime \prime}=\right.\) subst-lit \(\left.M \sigma\right\rangle\) and \(\langle\) validate-ground-lit ?I
\(L^{\prime \prime}>\)
            have validate-ground-clause ?I (subst-cl (cl-ecl C) \(\sigma\) )
            by (metis (mono-tags, lifting) subst-cl.simps mem-Collect-eq
                validate-ground-clause.simps)
            from this and cm show False by blast
qed
```

Now，it remains to prove that the considered clause yields a rule which can be used to reduce the left－hand side of the maximal equation，which（since no reduction is possible）entails that the left－hand side must be equivalent to the right－hand side（thus contradicting the fact that the clause is false）．
have（finite（cl－ecl C））by（simp add：〈C $\in S\rangle$ all－finite）
have $($ cl－ecl $C) \neq\{ \}$ by（simp add：$\langle C \in S\rangle$ all－non－empty）
from〈finite $($ cl－ecl $C)\rangle\langle($ cl－ecl $C) \neq\{ \} 〉\langle$ ground－clause（subst－cl（cl－ecl C） $\sigma)\rangle$
obtain $L$ where $L \in($ cl－ecl $C)$ eligible－literal $L C \sigma$ using eligible－lit－exists by metis
obtain $t s p$ where orient－lit－inst $L$ tspousing literal．exhaust equa－ tion．exhaust
using trm－ord－irrefl trm－ord－trans
unfolding orient－lit－inst－def irrefl－def trans－def by metis
We first show that the terms occurring inside variables are irreducible．To this aim，we need to consider the normal form of the substitution $\sigma$ ，obtained by replacing the image of each variable by its normal form．

```
    have \(\forall x y\).
    \(((x \in\) vars-of-cl \((\) cl-ecl \(C)) \longrightarrow\) occurs-in \(y(\) subst \((\) Var \(x) \sigma) \longrightarrow\) trm-rep \(y\)
\(S=y)\)
    proof (rule ccontr)
    assume \(\neg(\forall x y .(x \in\) vars-of-cl \((\) cl-ecl \(C)) \longrightarrow\) occurs-in \(y\) (subst (Var \(x)\)
\(\sigma) \longrightarrow\) trm-rep y \(S=y\) )
    then obtain \(x y\) where \((x \in\) vars-of-cl (cl-ecl \(C))\) and
        occurs-in \(y\) (subst \((\operatorname{Var} x) \sigma\) ) and trm-rep \(y \$ \neq y\) by blast
    from <occurs-in y (subst (Var \(x) \sigma\) ) 〉obtain \(p\) where subterm (subst (Var
x) \(\sigma) p y\)
            unfolding occurs-in-def by auto
            from 〈subterm (subst (Var x) \(\sigma\) ) p y〉 and 〈trm-rep y \(S \neq y\rangle\)
                have trm-rep (subst (Var x) \(\sigma) S \neq(\) subst \((\operatorname{Var} x) \sigma)\)
                using subts-of-irred-trms-are-irred by blast
    let \(? \vartheta=\) map-subst \((\lambda x\). \((\) trm-rep \(x S)) \sigma\)
    have equivalent-on \(\sigma\) ? \(\vartheta(\) vars-of-cl \((\) cl-ecl \(C))\) ?I
    proof (rule ccontr)
        assume \(\neg\) equivalent-on \(\sigma\) ? ७ (vars-of-cl (cl-ecl C)) ?I
        then obtain \(z\) where \(z \in\) vars-of-cl (cl-ecl C)
            and \(\neg(? I(\) subst \((\operatorname{Var} z) \sigma)(\) subst \((\operatorname{Var} z) ? \vartheta))\)
            unfolding equivalent-on-def by blast
        from \(\langle\neg(? I\) (subst (Var z) \(\sigma\) ) (subst (Var z) ? \(\vartheta\) ) ) 〉
            have trm-rep (subst (Var z) \(\sigma\) ) \(S \neq\) trm-rep (subst (Var z) ?V) \(S\)
            unfolding same-values-def int-clset-def by blast
            from this have trm-rep (trm-rep (subst (Var z) \(\sigma\) ) \(S\) ) \(S \neq\) trm-rep (subst
(Var z) ?V) \(S\)
            using trm-rep-involutive by metis
        from this have (subst (Var z) \(\sigma\) ) \(=(\operatorname{subst}(\operatorname{Var} z)\) ? \(\vartheta)\)
            using map-subst-lemma [of z \(\sigma \lambda\). (trm-rep \(x\) S)] by metis
            from this and \(\langle\neg(? I(\operatorname{subst}(\operatorname{Var} z) \sigma)(\operatorname{subst}(\operatorname{Var} z)\) ? \(\vartheta))\rangle\)
            show False using 〈fo-interpretation ?I〉
            unfolding fo-interpretation-def congruence-def equivalence-relation-def
reflexive-def
            by metis
    qed
```

```
    from this and \(\prec \neg\) validate-ground-clause ?I (subst-cl (cl-ecl C) \(\sigma\) )〉
    〈fo-interpretation? ? 〉
    have \(\neg\) validate-ground-clause ?I (subst-cl (cl-ecl C) ?७)
    using equivalent-on-cl by metis
    have lower-on ? \(\vartheta \sigma\) (vars-of-cl (cl-ecl C))
    proof (rule ccontr)
    assume \(\neg l o w e r-o n ~ ? \vartheta ~ \sigma(v a r s-o f-c l(c l-e c l ~ C)) ~\)
    then obtain \(z\) where \(z \in\) vars-of-cl (cl-ecl \(C\) )
        and (subst (Var z) \(\sigma\) ) \(\neq(\) subst \((\operatorname{Var} z)\) ? \(\vartheta)\)
        and \(((\) subst \((\operatorname{Var} z) ? \vartheta),(\) subst \((\operatorname{Var} z) \sigma)) \notin\) trm-ord
        unfolding lower-on-def lower-or-eq-def by metis
    from «(subst \((\operatorname{Var} z) \sigma) \neq(\) subst \((\operatorname{Var} z)\) ? \(\vartheta)\rangle\) have
        (trm-rep (subst \((\operatorname{Var} z) \sigma) S)=(\) subst \((\operatorname{Var} z)\) ? \(\vartheta)\)
        using map-subst-lemma \([\) of \(z \sigma \lambda x\). (trm-rep \(x S)]\) by metis
    from this and 〈(subst \((\operatorname{Var} z) \sigma) \neq(\) subst \((\operatorname{Var} z)\) ? \(\vartheta)\) 〉
        and «((subst \((\operatorname{Var} z)\) ? \(\vartheta),(\) subst \((\operatorname{Var} z) \sigma)) \notin\) trm-ord \(\rangle\)
            show False using trm-rep-is-lower by metis
qed
have subst \((\operatorname{Var} x) \sigma \neq(\operatorname{Var} x)\)
proof
    assume subst \((\operatorname{Var} x) \sigma=(\operatorname{Var} x)\)
    from this and \(\langle x \in\) vars-of-cl (cl-ecl \(C\) ) 〉 have \(\neg\) (ground-on (vars-of-cl
\((\) cl-ecl \(C)) \sigma\) )
        unfolding ground-on-def ground-term-def by auto
        from this and 〈ground-clause (subst-cl (cl-ecl C) \(\sigma\) ) 〉
        show False using ground-clauses-and-ground-substs by metis
    qed
    from 〈subst \((\operatorname{Var} x) \sigma \neq(\operatorname{Var} x)\rangle\)
        have \((\) trm-rep \((\) subst \((\operatorname{Var} x) \sigma) S)=(\) subst \((\operatorname{Var} x)\) ? \(\vartheta)\)
            using map-subst-lemma [of \(x \sigma \lambda\). (trm-rep \(x S)\) ] by metis
    from this and 〈trm-rep (subst (Var x) \(\sigma\) ) \(S \neq(\) subst \((\operatorname{Var} x) \sigma)\rangle\)
        have \(((\) subst \((\operatorname{Var} x)\) ? \(\vartheta),(\) subst \((\operatorname{Var} x) \sigma)) \in\) trm-ord
        using trm-rep-is-lower by metis
    from 〈lower-on ? \(\vartheta \sigma(\) vars-of-cl \((\) cl-ecl \(C))\) ) and \(\langle x \in \operatorname{vars-of-cl~(cl-ecl~C)~}\rangle\)
        〈finite (cl-ecl C)〉
        and \(\langle((\) subst \((\operatorname{Var} x)\) ? \(\vartheta),(\) subst \((\operatorname{Var} x) \sigma)) \in\) trm-ord \(\rangle\)
    have \(((C, ? \vartheta),(C, \sigma)) \in\) ecl-ord
        using lower-on-cl by blast
    from \(\langle C=\) fst pair \(\langle\langle\sigma=\) snd pair \(\rangle\) have pair \(=(C, \sigma)\) by auto
    from this and \(\langle((C, ? \vartheta),(C, \sigma)) \in\) ecl-ord \(\rangle\) have
        \(((C, ? \vartheta), p a i r) \in\) ecl-ord
        by metis
    from this and hyp-ind have ?P \((C, ? \vartheta)\) by blast
    from 〈(all-trms-irreducible (subst-set (trms-ecl C) \(\sigma\) )
                \((\lambda t\). (trm-rep \(t S))\) )
        〈lower-on? \(\sigma\) (vars-of-cl (cl-ecl \(C)\) )〉〈C \(\in S\rangle\langle f o-i n t e r p r e t a t i o n ? ~ ? ~\rangle ~\)
        〈equivalent-on \(\sigma\) ? \(\vartheta\) (vars-of-cl (cl-ecl C)) ?I〉assms(3)
        have (all-trms-irreducible (subst-set (trms-ecl C) ?V)
                \((\lambda t .(\) trm-rep \(t S)))\)
```

using irred－terms－and－reduced－subst unfolding Ball－def well－constrained－def
by metis
have ground－clause（subst－cl（cl－ecl C）？V）
proof－
from 〈ground－clause（subst－cl（cl－ecl C）$\sigma$ ）〉
have ground－on（vars－of－cl（cl－ecl C））$\sigma$ using ground－clauses－and－ground－substs by auto
from this and＜lower－on？$\sigma$（vars－of－cl（cl－ecl C））＞
have ground－on（vars－of－cl（cl－ecl C））？$\vartheta$
using lower－on－ground by meson
from this show ？thesis using ground－substs－yield－ground－clause by metis
qed
from this
〈（all－trms－irreducible（subst－set（trms－ecl C）？$\vartheta$ ）$(\lambda t$ ．（trm－rep $t S))$ ）＞
$\langle ? P(C, ? \vartheta)\rangle\langle\neg$ validate－ground－clause ？I（subst－cl（cl－ecl C）？V）〉
$\langle C \in S\rangle$ show False unfolding int－clset－def by（metis fst－conv snd－conv）
qed
Next，we show that the eligible term $t$ is in normal form．We first need to establish the result for proper subterms of $t$ before considering the general case．

```
    have }\neg(\mathrm{ proper-subterm-red t S }\sigma
        proof
        assume (proper-subterm-red t S \sigma)
        from this have trm-rep (subst t \sigma) S\not= subst t \sigma
    using proper-subterm-red-def substs-preserve-subterms subts-of-irred-trms-are-irred
            by blast
        from〈(proper-subterm-red t S \sigma)\rangle
            \forallxy.
            ((x\invars-of-cl (cl-ecl C)) \longrightarrow occurs-in y (subst (Var x)\sigma)\longrightarrowtrm-rep y
S=y)>
    <eligible-literal L C \sigma`
    <trm-rep (subst t \sigma) S\not= subst t \sigma\rangle\langleL\in cl-ecl C>
    <orient-lit-inst L tspa>< \forallx\inS. finite (cl-ecl x)\rangle
    <ground-clause (subst-cl (cl-ecl C) \sigma)\rangle
    <fo-interpretation (int-clset S)〉
    \Ball S well-constrained\rangle}\langleC\inS
    <all-trms-irreducible (subst-set (trms-ecl C) \sigma) (\lambdat. trm-rep t S)>
    \checkmark validate-ground-clause (int-clset S) (subst-cl (cl-ecl C)\sigma)\rangle
    <closed-under-renaming S`
```


## have

```
\(\exists \sigma^{\prime \prime} u u^{\prime}\) pa v D L2．（reduction L C \(\sigma^{\prime \prime} t\) s pL2 u u＇pa v D
            (same-values ( }\lambdat.\mathrm{ trm-rep t S)) S 
            using reduction-exists [of pts C S \sigma L] unfolding int-clset-def by
blast
```

from this and 〈？nored〉 show False unfolding int－clset－def by blast

## qed

have $p=$ neg $\vee \neg$ equivalent－eq－existsts（cl－ecl $C$ ）（same－values（ $\lambda x$ ．trm－rep x $S$ ））$\sigma$
$L$
proof（rule ccontr）
assume neg：$\neg(p=n e g \vee \neg$ equivalent－eq－exists $t s(c l-e c l C)$
（same－values $(\lambda x$ ．trm－rep $x S)) \sigma L$ ）
then have $p \neq$ neg by metis
from neg have equivalent－eq－exists $t s$（cl－ecl C）（same－values（ $\lambda$ x．trm－rep xS））$\sigma L$
by metis
from $\langle p \neq n e g\rangle$ have $p=$ pos using sign．exhaust by auto
from＜equivalent－eq－exists ts（cl－ecl C）（same－values（ $\lambda$ x．trm－rep x $S$ ））$\sigma$ L＞
obtain L2 where $L 2 \in($ cl－ecl $C)-\{L\}$ and $f: \exists u$ v．orient－lit－inst L2 uv pos $\sigma \wedge$
subst $t \sigma=$ subst $u \sigma \wedge$ trm－rep（subst s $\sigma$ ）$S=$ trm－rep（subst $v \sigma$ ）
$S$
unfolding equivalent－eq－exists－def unfolding same－values－def by metis
from $f$ obtain $u v$ where $f^{\prime}$ ：orient－lit－inst L2 $u v \operatorname{pos} \sigma \wedge$ subst $t \sigma=$ subst $u \sigma$
$\wedge$ trm－rep（subst s $\sigma$ ）$S=$ trm－rep $($ subst $v \sigma) S$
by blast
from $f^{\prime}$ have orient－lit－inst L2 $u v$ pos $\sigma$ by metis
from $f^{\prime}$ have subst $t \sigma=$ subst $u \sigma$ by metis
from $f^{\prime}$ have trm－rep（subst s $\sigma$ ）$S=$ trm－rep（subst $v \sigma$ ）$S$ by metis
from 〈orient－lit－inst L2 u v pos $\sigma\rangle\langle$ subst $t \sigma=$ subst $u \sigma\rangle$
$\langle$ trm－rep（subst s $\sigma$ ）$S=$ trm－rep（subst $v \sigma$ ）$S\rangle$
$\langle$ orient－lit－inst $L$ tsp $\sigma\rangle\langle p=$ pos $\langle L \in($ cl－ecl $C)\rangle\langle L 2 \in($ cl－ecl $C)-$
$\{L\}$＞
$\langle$ eligible－literal L $C \sigma\rangle$
$\langle\neg($ proper－subterm－red $t S \sigma)$ 〉
and 〈？no－fact〉 show False by blast
qed

```
have（trm－rep（subst \(t \sigma) S)=(\) subst \(t \sigma)\)
```

proof（rule ccontr）
assume（trm－rep（subst $t \sigma) S) \neq($ subst $t \sigma)$
from $\langle p=$ neg $\vee \neg$ equivalent－eq－exists $t s$（cl－ecl $C$ ）（same－values $(\lambda x$. trm－rep $x S$ ））$\sigma L>$
$\langle\forall x y$ ．
$((x \in$ vars－of－cl $($ cl－ecl $C)) \longrightarrow$ occurs－in $y($ subst $($ Var $x) \sigma) \longrightarrow$ trm－rep $y$ $S=y)$＞
$\langle$ eligible－literal L $C \quad \sigma\rangle$
$\langle$ trm－rep（subst $t \sigma) S \neq$ subst $t \sigma\rangle\langle L \in$ cl－ecl $C\rangle$
〈orient－lit－inst Ltspo〉〈 $\forall x \in S$ ．finite（cl－ecl x）〉
〈ground－clause（subst－cl（cl－ecl C）$\sigma$ ）〉

```
<fo-interpretation (int-clset S)>
<Ball S well-constrained\rangle}\langleC\inS
<all-trms-irreducible (subst-set (trms-ecl C) \sigma) (\lambdat. trm-rep t S)>
\imath validate-ground-clause (int-clset S) (subst-cl (cl-ecl C) \sigma)〉
<closed-under-renaming S`
```


## have



```
                (same-values ( }\lambdat\mathrm{ . trm-rep t S))S S ^ variable-disjoint C D)
                using reduction-exists [of pts C S \sigma L] unfolding int-clset-def by
```

blast
from this and 〈?nored〉 show False unfolding int-clset-def by blast
qed
from «orient-lit-inst Ltsp $\sigma$ have $(($ subst $t \sigma),($ subst s $\sigma)) \notin$ trm-ord
unfolding orient-lit-inst-def by auto
from 〈ground-clause (subst-cl (cl-ecl C) $\sigma$ )〉
have vars-of-cl (subst-cl (cl-ecl C) $\sigma$ ) $=\{ \}$ by auto
from this and $\langle L \in($ cl-ecl $C)\rangle$ have vars-of-lit (subst-lit $L \sigma)=\{ \}$ by auto
from <orient-lit-inst $L$ tspo
have orient-lit (subst-lit L $\sigma$ ) (subst $t \sigma)($ subst s $\sigma) p$
using lift-orient-lit by auto
from 〈orient-lit (subst-lit L $\sigma$ ) (subst t $\sigma$ ) (subst s $\sigma$ ) p〉
have vars-of (subst $t \sigma) \subseteq$ vars-of-lit (subst-lit L $\sigma$ )
using orient-lit-vars by auto
from this and «vars-of-lit (subst-lit $L \sigma$ ) $=\{ \}$ 〉 have vars-of (subst $t \sigma$ ) $=$
\{\}
by auto
from 〈orient-lit (subst-lit $L \sigma$ ) (subst $t \sigma)($ subst $s \sigma) p\rangle$
have vars-of (subst s $\sigma$ ) $\subseteq$ vars-of-lit (subst-lit $L \sigma$ )
using orient-lit-vars by auto
from this and 〈vars-of-lit (subst-lit $L \sigma$ ) $=\{ \}$ 〉have vars-of (subst s $\sigma$ ) $=$
\{\}
by auto
from
$\langle(($ subst $t \sigma),($ subst $s \sigma)) \notin$ trm-ord $\rangle$
$\langle$ vars-of (subst $t \sigma)=\{ \}\rangle\langle v a r s-o f($ subst $s \sigma)=\{ \}\rangle$
have $($ subst $t \sigma)=($ subst $s \sigma) \vee(($ subst $s \sigma),($ subst $t \sigma)) \in$ trm-ord
using trm-ord-ground-total unfolding ground-term-def by blast
from $\langle L \in($ cl-ecl $C)\rangle$ have $($ subst-lit $L \sigma) \in($ subst-cl $($ cl-ecl $C) \sigma)$ by auto

Using the fact that the eligible term is in normal form and that the eligible literal is false in the considered interpretation but is not a contradiction，we deduce that this literal must be positive．

```
have p=pos
proof (rule ccontr)
    assume p}=\mathrm{ pos
    from this have p=neg using sign.exhaust by auto
    from <trm-rep (subst t \sigma)S=(subst t \sigma)\rangle\langleL\in(cl-ecl C)\rangle\langleeligible-literal
```

$L C \sigma\rangle$
and 〈orient－lit－inst Ltspo〉〈p＝neg〉〈？no－cont〉
have（subst $t \sigma) \neq($ subst $s \sigma)$ by blast
from this and
$\langle($ subst $t \sigma)=($ subst $s \sigma) \vee(($ subst s $\sigma),($ subst $t \sigma)) \in$ trm－ord $\rangle$
and $\langle$ trm－rep（subst $t \sigma$ ）$S=$ subst $t \sigma\rangle$
have $(($ trm－rep $($ subst $s \sigma) S),($ trm－rep $($ subst $t \sigma) S)) \in$ trm－ord
using trm－rep－is－lower［of（subst s $\sigma$ ）$S$ ］trm－ord－trans unfolding trans－def by metis
from this have（trm－rep（subst s $\sigma$ ）$S$ ）$\neq($ trm－rep $($ subst $t \sigma) S)$
using trm－ord－irrefl irrefl－def by metis
from this have $\neg$ validate－ground－eq ？I（Eq（subst $t \sigma)($ subst $s \sigma)$ ）
unfolding same－values－def int－clset－def using validate－ground－eq．simps by（metis（mono－tags，lifting））
from 〈（trm－rep（subst s $\sigma$ ）$S) \neq($ trm－rep $($ subst $t \sigma) S)$ 〉
have $\neg$ validate－ground－eq ？I（Eq（subst s $\sigma$ ）（subst $t \sigma)$ ）
unfolding same－values－def int－clset－def using validate－ground－eq．simps by（metis（mono－tags，lifting））
from 〈orient－lit－inst $L t s p \sigma\rangle$ and $\langle p=n e g\rangle$ have $L=(N e g(E q t s)) \vee L$ $=(N e g(E q s t))$
unfolding orient－lit－inst－def by auto
from this have subst－lit $L \sigma=(\operatorname{Neg}(E q($ subst $t \sigma)($ subst s $\sigma)))$
$\vee$ subst－lit $L \sigma=(\operatorname{Neg}(E q($ subst s $\sigma)($ subst $t \sigma)))$ by auto
from this and «ᄀvalidate－ground－eq ？I（Eq（subst s $\sigma$ ）（subst $t \sigma)$ ）＞
and «ᄀvalidate－ground－eq ？I（Eq（subst $t \sigma)($ subst $s \sigma)$ ）＞
have validate－ground－lit？I（subst－lit L $\sigma$ ）using validate－ground－lit．simps（2）
by metis
from 〈（subst－lit $L \sigma) \in($ subst－cl $($ cl－ecl $C) \sigma)\rangle$
and 〈validate－ground－lit ？I（subst－lit L $\sigma$ ）〉
have validate－ground－clause ？I（subst－cl（cl－ecl C）$\sigma$ ）
using validate－ground－clause．elims（3）by blast
from this and $\langle\neg$ validate－ground－clause ？I（subst－cl（cl－ecl C）$\sigma$ ）〉 show
False by blast
qed
This entails that the right－hand side of the eligible literal occurs in the set of possible values for the left－hand side $t$ ，which is impossible since this term is irreducible．
from $\langle L \in($ cl－ecl $C)\rangle\langle o r i e n t-l i t-i n s t L t s p\rangle\langle p=p o s\rangle$
$\langle\neg$ validate－ground－clause ？I（subst－cl（cl－ecl C）$\sigma$ ）〉
have trm－rep（subst $t \sigma$ ）$S \neq$ trm－rep（subst s $\sigma$ ）$S$
using no－valid－literal by metis
from this have（subst $t \sigma) \neq($ subst $s \sigma$ ）by metis
from this and «（subst $t \sigma)=($ subst $s \sigma) \vee(($ subst s $\sigma),($ subst $t \sigma)) \in$ trm－ord〉 have $(($ subst $s \sigma),($ subst $t \sigma)) \in$ trm－ord
using trm－ord－ground－total unfolding ground－term－def by blast
from $\langle p=p o s\rangle$ and $\langle o r i e n t-l i t-i n s t L t s p r\rangle$ have $\neg$ negative－literal $L$ unfolding orient－lit－inst－def by auto
from this and «eligible－literal $L C \sigma$ 〉 have sel（cl－ecl C）$=\{ \}$ and maximal－literal（subst－lit L $\sigma$ ）（subst－cl（cl－ecl
C）$\sigma$ ）
using sel－neg unfolding eligible－literal－def by auto
from $\langle\neg$ validate－ground－clause ？I（subst－cl（cl－ecl C）$\sigma$ ）〉
have smaller－lits－are－false（subst t $\sigma$ ）（subst－cl（cl－ecl C）$\sigma$ ）$S$
using smaller－lits－are－false－if－cl－not－valid $[$ of $S($ subst－cl $(c l-e c l ~ C) \sigma)]$ by
blast
from $\langle p=p o s\rangle$ and $\langle p=n e g \vee \neg$ equivalent－eq－exists $t s(c l-e c l C)$
（same－values $(\lambda x$. trm－rep $x S)) \sigma L\rangle$
have $\neg$ equivalent－eq－exists $t s(c l-e c l C)($ int－clset $S) \sigma L$ unfolding int－clset－def
using sign．distinct by metis
from this $\langle p=p o s\rangle$ have maximal－literal－is－unique（subst $t \sigma$ ）（subst $s \sigma$ ）（cl－ecl C）$L S \sigma$
using maximal－literal－is－unique－lemma［of ts（cl－ecl C）S $S$ L］by blast from «all－trms－irreducible（subst－set（trms－ecl C）$\sigma$ ）（ $\lambda t$ ．trm－rep $t S)$ 〉
have trms－irreducible C $\sigma S$（subst $t \sigma$ ）
using trms－irreducible－lemma by blast
have（subst t $\sigma$ ）$\notin$ subst－set $($ trms－ecl $C) \sigma$
proof
assume（subst $t \sigma) \in$ subst－set（trms－ecl C）$\sigma$
from this obtain $t^{\prime}$ where $t^{\prime} \in \operatorname{trms}$－ecl $C$ and（subst $\left.t^{\prime} \sigma\right)=($ subst $t \sigma)$ by auto
from $\left\langle t^{\prime} \in\right.$ trms－ecl $\left.C\right\rangle$ and assms（3）and $\langle C \in S\rangle$ have dom－trm $t^{\prime}($ cl－ecl C）
unfolding Ball－def well－constrained－def by auto
from this obtain $M u v q$ where $M \in($ cl－ecl $C)$ decompose－literal $M u v q$ and
$q=n e g \wedge\left(u=t^{\prime}\right) \vee\left(\left(t^{\prime}, u\right) \in\right.$ trm－ord $)$ unfolding dom－trm－def by blast obtain $u^{\prime} v^{\prime} q^{\prime}$ where orient－lit－inst $M u^{\prime} v^{\prime} q^{\prime} \sigma$ using literal．exhaust equation．exhaust using trm－ord－irrefl trm－ord－trans
unfolding orient－lit－inst－def irrefl－def trans－def by metis
from 〈decompose－literal $M u v q\rangle$ and $\left\langle o r i e n t-l i t-i n s t M u^{\prime} v^{\prime} q^{\prime} \sigma\right\rangle$
have $u=u^{\prime} \vee u=v^{\prime}$
unfolding decompose－literal－def orient－lit－inst－def
by（metis atom．simps（2）decompose－equation－def equation．inject lit－ eral．distinct（1）
literal．inject（1））
from 〈decompose－literal $M u v q\rangle$ and＜orient－lit－inst $\left.M u^{\prime} v^{\prime} q^{\prime} \sigma\right\rangle$
have $q=q^{\prime}$
unfolding decompose－literal－def orient－lit－inst－def by auto
from «vars－of－cl（subst－cl $($ cl－ecl $C) \sigma)=\{ \}\rangle$ and $\langle M \in($ cl－ecl $C)\rangle$
have vars－of－lit（subst－lit $M \sigma$ ）$=\{ \}$ by auto from＜orient－lit－inst $\left.M u^{\prime} v^{\prime} q^{\prime} \sigma\right\rangle$ have
orient－lit（subst－lit $M \sigma)\left(\right.$ subst $\left.u^{\prime} \sigma\right)\left(\right.$ subst $\left.v^{\prime} \sigma\right) q^{\prime}$
using lift－orient－lit by auto
from＜orient－lit－inst Ltspo＞have
orient－lit（subst－lit L $\sigma$ ）（subst $t \sigma)($ subst $s \sigma) p$
using lift－orient－lit by auto
have $\left(t^{\prime}, u\right) \notin$ trm－ord
proof
assume $\left(t^{\prime}, u\right) \in$ trm－ord
then have $\left(\left(\right.\right.$ subst $\left.t^{\prime} \sigma\right),($ subst $\left.u \sigma)\right) \in$ trm－ord
using trm－ord－subst by auto
from this and $\left\langle\left(\right.\right.$ subst $\left.t^{\prime} \sigma\right)=($ subst $\left.t \sigma)\right\rangle$ have
$(($ subst $t \sigma),($ subst $u \sigma)) \in$ trm－ord by auto
from＜orient－lit（subst－lit M $\sigma$ ）（subst $\left.u^{\prime} \sigma\right)\left(\right.$ subst $\left.v^{\prime} \sigma\right) q^{\prime}$ 〉
and＜orient－lit（subst－lit L $\sigma$ ）（subst $t \sigma)($ subst $s \sigma) p\rangle$
and $\langle(($ subst $t \sigma),($ subst $u \sigma)) \in$ trm－ord $\rangle$
and 〈vars－of－lit（subst－lit M $\sigma$ ）$=\{ \}\rangle$
and 〈vars－of－lit（subst－lit L $\sigma$ ）$=\{ \}\rangle$
and $\left\langle u=u^{\prime} \vee u=v^{\prime}\right.$ 〉
have $(($ subst－lit $L \sigma),($ subst－lit $M \sigma)) \in$ lit－ord
using lit－ord－dominating－term by metis
from this and «maximal－literal（subst－lit L $\sigma$ ）（subst－cl（cl－ecl C）$\sigma$ ）＞
and $\langle M \in($ cl－ecl $C)\rangle$ show False using maximal－literal－def by auto
qed
have $\neg\left(q=n e g \wedge\left(u=t^{\prime}\right)\right)$
proof
assume $q=n e g \wedge\left(u=t^{\prime}\right)$
then have $q=n e g$ and $u=t^{\prime}$ by auto
from＜orient－lit（subst－lit $M \sigma)\left(\right.$ subst $\left.u^{\prime} \sigma\right)\left(\right.$ subst $\left.\left.v^{\prime} \sigma\right) q^{\prime}\right\rangle$
and $\langle$ orient－lit（subst－lit L $\sigma$ ）（subst $t \sigma$ ）（subst s $\sigma$ ）p〉
and $\left\langle u=t^{\prime}\right\rangle$
and $\left\langle\left(\right.\right.$ subst $\left.t^{\prime} \sigma\right)=($ subst $\left.t \sigma)\right\rangle$
and $\langle q=n e g\rangle$ and $\left\langle q=q^{\prime}\right\rangle$
and $\langle p=p o s\rangle$
and 〈vars－of－lit（subst－lit $M \sigma$ ）$=\{ \}\rangle$
and 〈vars－of－lit（subst－lit L $\sigma$ ）$=\{ \}\rangle$
and $\left\langle u=u^{\prime} \vee u=v^{\prime}\right.$ 〉
have $(($ subst－lit $L \sigma),($ subst－lit $M \sigma)) \in$ lit－ord
using lit－ord－neg－lit－lhs lit－ord－neg－lit－rhs by metis
from this and «maximal－literal（subst－lit L $\sigma$ ）（subst－cl（cl－ecl C）$\sigma$ ）＞
and $\langle M \in($ cl－ecl $C)\rangle$ show False using maximal－literal－def by auto
qed
from this and $\left\langle\left(t^{\prime}, u\right) \notin\right.$ trm－ord $\rangle$ and $\left\langle q=n e g \wedge\left(u=t^{\prime}\right) \vee\left(\left(t^{\prime}, u\right) \in\right.\right.$ trm－ord）＞
show False by auto
qed
from $\langle C \in S\rangle\langle($ subst $s \sigma$, subst $t \sigma) \in$ trm－ord $\rangle$

```
    and «p=pos\rangle<orient-lit-inst L t s p \sigma\rangle and <sel (cl-ecl C)={}>
    and <L <cl-ecl C>
    and <maximal-literal (subst-lit L \sigma) (subst-cl (cl-ecl C) \sigma)>
    and <ground-clause (subst-cl (cl-ecl C) \sigma)\rangle
    and <finite (cl-ecl C)〉
    and «smaller-lits-are-false (subst t \sigma) (subst-cl (cl-ecl C) \sigma) S〉
    and <maximal-literal-is-unique (subst t \sigma) (subst s \sigma) (cl-ecl C) L S \sigma`
    and <trms-irreducible C\sigmaS (subst t \sigma)>
    and <(subst t \sigma)\not\in subst-set (trms-ecl C) \sigma〉
    have cv: (candidate-values (trm-rep (subst s \sigma) S) C (cl-ecl C)
    (subst-cl (cl-ecl C) \sigma) (subst s \sigma) (subst-lit L \sigma)L \sigma t s (subst t \sigma) S)
    unfolding candidate-values-def by blast
    from cv have (trm-rep (subst s \sigma)S,(subst s \sigma)) \in set-of-candidate-values
S (subst t \sigma)
            unfolding set-of-candidate-values-def by blast
    from <trm-rep (subst t \sigma) S=(subst t \sigma)〉
        have }\neg(\mathrm{ subterm-reduction-applicable S (subst t }\sigma\mathrm{ ))
        using trm-rep-is-lower-subt-red trm-ord-irrefl irrefl-def
        by metis
    from <(trm-rep (subst s \sigma) S, subst s \sigma)
    set-of-candidate-values S (subst t \sigma)>
    have set-of-candidate-values S (subst t \sigma)\not={} by blast
    from <(trm-rep (subst s \sigma) S, subst s \sigma)
    \epsilon set-of-candidate-values S (subst t \sigma)>
    have min-trms (set-of-candidate-values S (subst t \sigma))}\not={
    using min-trms-not-empty by blast
    from <\neg(subterm-reduction-applicable S (subst t \sigma))`
        <min-trms (set-of-candidate-values S (subst t \sigma)) \not= {}>
        have (trm-rep (subst t \sigma) S,(subst t \sigma)) \in trm-ord
        using trm-rep-is-lower-root-red [of S subst t \sigma] by blast
    from this and <(trm-rep (subst t \sigma)S)=(subst t \sigma)\rangle
    show False using trm-ord-irrefl irrefl-def by metis
qed
qed
```

As an immediate consequence of the previous lemma，we show that the set of clauses that are derivable from an unsatisfiable clause set must contain an empty clause（since this set is trivially saturated）．
lemma COMPLETENESS：
assumes $\forall x .(x \in S \longrightarrow($ trms－ecl $x=\{ \}))$
assumes $(\forall x \in S$ ．finite（ cl－ecl $x)$ ）
assumes $\neg($ satisfiable－clause－set（cl－ecl＇$S)$ ）
shows $\exists x$ ．（derivable－ecl $x S) \wedge$ cl－ecl $x=\{ \}$
proof（rule ccontr）
assume $\neg(\exists x .($ derivable－ecl $x S) \wedge$ cl－ecl $x=\{ \})$
let $? S=\{y$ ．（derivable－ecl $y S)\}$
let ？$I=$ same－values $(\lambda x$ ．（trm－rep $x$ ？$S$ ）$)$
have fo－interpretation？？using trm－rep－compatible－with－structure same－values－fo－int
by metis

```
have \(\forall x \in\) ?S. \((\) cl-ecl \(x) \neq\{ \}\)
proof (rule ccontr)
    assume \(\neg\) ?thesis
    then obtain \(x\) where \(x \in ? S\) and cl-ecl \(x=\{ \}\) by blast
    from \(\langle x \in ? S\rangle\) have derivable-ecl \(x S\) by (meson CollectD)
    from this \(\langle\) cl-ecl \(x=\{ \}\rangle\langle\neg(\exists x\). (derivable-ecl \(x S) \wedge\) cl-ecl \(x=\{ \})\rangle\)
        show False by metis
qed
have all-finite: \(\forall x \in\) ?S. (finite (cl-ecl \(x\) ))
proof (rule ccontr)
    assume \(\neg\) ?thesis
    then obtain \(x\) where \(x \in ? S\) and \(\neg\) finite (cl-ecl \(x\) ) by blast
    from \(\langle x \in ? S\rangle\) have derivable-ecl \(x S\) by (meson CollectD)
    from this assms(2) \(\neg\) finite (cl-ecl x)〉 show False using all-derived-clauses-are-finite
by metis
    qed
    have Ball \(S\) well-constrained
    proof
    fix \(x\) assume \(x \in S\)
    from this assms(1) have trms-ecl \(x=\{ \}\) by auto
    from this show well-constrained \(x\) unfolding well-constrained-def by blast
qed
have Ball ?S well-constrained
proof
    fix \(x\) assume \(x \in\) ? \(S\)
    from this have derivable-ecl \(x S\) by (meson CollectD)
    from this assms(2)〈Ball \(S\) well-constrained〉show well-constrained \(x\)
        using all-derived-clauses-are-wellconstrained
        by metis
    qed
    have closed-under-renaming ?S
    proof (rule ccontr)
    assume \(\neg\) ?thesis
    then obtain \(C D\) where \(C \in ? S\) renaming-cl \(C D D \notin ? S\)
        unfolding closed-under-renaming-def by metis
    from \(\langle C \in ? S\rangle\) have derivable-ecl \(C S\) by (meson CollectD)
    from 〈derivable-ecl \(C S\rangle\langle r e n a m i n g-c l ~ C D\rangle\) have (derivable-ecl \(D S\) )
        using derivable-ecl.intros(2) by metis
    from this and \(\langle D \notin ? S\rangle\) show False by blast
qed
have inference-closed ?S
proof (rule ccontr)
    assume \(\neg\) ?thesis
```

then obtain $D P \vartheta C^{\prime}$ where（derivable $D P ? S \vartheta$ FirstOrder $\left.C^{\prime}\right) D \notin ? S$ unfolding inference－closed－def by metis
from «derivable $D P ? S \vartheta$ FirstOrder $C^{\prime} 〉$ have $P \subseteq ? S$ using derivable－premisses

## by metis

have $\forall x . x \in P \longrightarrow$ derivable－ecl $x S$
proof $(($ rule allI $),($ rule impI $))$
fix $x$ assume $x \in P$
from this and $\langle P \subseteq ? S\rangle$ have $x \in ? S$ by（meson rev－subsetD）
from this show derivable－ecl $x S$ by（meson CollectD）
qed
from this and «（derivable D P ？S v FirstOrder $\left.\left.C^{\prime}\right)\right\rangle$ have derivable－ecl D S
using derivable－ecl．intros（3）［of PSD？S $\left.\vartheta C^{\prime}\right]$ by meson
from this and $\langle D \notin ? S\rangle$ show False by blast
qed
from this all－finite have clause－saturated？S
using inference－closed－sets－are－saturated by meson
from this all－finite have inference－saturated？S
using clause－saturated－and－inference－saturated by meson
from this have ground－inference－saturated ？S
using lift－inference by metis
have validate－clause－set ？I（cl－ecl＇S）
proof（rule ccontr）
assume $\neg$ ？thesis
from this obtain $C l-C$ where clc：$C l-C \in(c l-e c l ' S)$ and $\neg$（validate－clause
？I $\mathrm{Cl}-\mathrm{C}$ ）
using validate－clause－set．simps by metis
from clc obtain $C$ where $C \in S$ and $C l-C=($ cl－ecl $C)$ by blast
from $\langle C \in S\rangle$ have derivable－ecl $C S$
using derivable－ecl．intros（1）by metis
from this have $C \in$ ？S by blast
from 〈 $\neg$（validate－clause ？I Cl －$C$ ）〉 obtain $\sigma$
where $\neg($ validate－ground－clause ？I（subst－cl Cl－C $\sigma$ ））
and ground－clause（subst－cl Cl－C $\sigma$ ）
using validate－clause．simps by metis
let ？pair $=(C, \sigma)$
have $f$ st ？pair $=C$ by auto
have snd ？pair $=\sigma$ by auto
from $\langle C \in S\rangle \operatorname{assms}(1)$ have trms－ecl $C=\{ \}$ by auto
then have（subst－set（trms－ecl C）$\sigma$ ）$=\{ \}$ by auto
then have $n$ ：all－trms－irreducible（subst－set（trms－ecl C）$\sigma$ ）
（ $\lambda t$ ．trm－rep $t\{y$ ．derivable－ecl y $S\}$ ）
unfolding all－trms－irreducible－def by blast
from〈ground－inference－saturated ？S〉 all－finite 〈Ball ？S well－constrained〉〈closed－under－renaming ？$S\rangle\langle\forall x \in ? S .($ cl－ecl $x) \neq\{ \}\rangle$
have $\forall C \sigma . f s t$ ？pair $=C \longrightarrow$
$\sigma=$ snd ？pair $\longrightarrow$
$C \in\{y$ ．derivable－ecl $y S\} \longrightarrow$

```
        ground-clause (subst-cl (cl-ecl C) \sigma)}
        all-trms-irreducible (subst-set (trms-ecl C) \sigma) (\lambdat. trm-rep t {y.derivable-ecl
y S})
            \longrightarrow \text { validate-ground-clause ?I (subst-cl (cl-ecl C) \%)}
            using int-clset-is-a-model [of ?S ?pair] by blast
    from this <fst ?pair = C\rangle\langleC E?S\rangle\langlesnd ?pair = \sigma\rangle\langleCl-C= (cl-ecl C)\rangle
                <ground-clause (subst-cl Cl-C \sigma)〉n
            have validate-ground-clause ?I (subst-cl (cl-ecl C) \sigma) by metis
    from this and }\neg\neg(validate-ground-clause ?I (subst-cl Cl-C \sigma))\rangle\langleCl-C=(cl-ecl
C)>
    show False by metis
qed
from this and assms(3)<fo-interpretation ?I` show False using satisfiable-clause-set-def
by metis
qed
end
end
```


## References

[1] L. Bachmair and H. Ganzinger. Rewrite-based equational theorem proving with selection and simplification. Journal of Logic and Computation, $3(4): 217-247,1994$.
[2] L. Bachmair, H. Ganzinger, C. Lynch, and W. Snyder. Basic paramodulation and superposition. In D. Kapur, editor, Automated Deduction - CADE-11, 11th International Conference on Automated Deduction, Saratoga Springs, NY, USA, June 15-18, 1992, Proceedings, volume 607 of Lecture Notes in Computer Science, pages 462-476. Springer, 1992.
[3] L. Bachmair, H. Ganzinger, C. Lynch, and W. Snyder. Basic paramodulation. Inf. Comput., 121(2):172-192, 1995.
[4] R. Nieuwenhuis and A. Rubio. Paramodulation-based theorem proving. In J. A. Robinson and A. Voronkov, editors, Handbook of Automated Reasoning, pages 371-443. Elsevier and MIT Press, 2001.

