Reducing Rewrite Properties to Properties on Ground Terms*

Alexander Lochmann

September 13, 2023

Abstract

This AFP entry relates important rewriting properties between the set of terms and the set of ground terms induced by a given signature. The properties considered are confluence, strong/local confluence, the normal form property, unique normal forms with respect to reduction and conversion, commutation, conversion equivalence, and normalization equivalence.

Contents

*Supported by FWF (Austrian Science Fund) projects P30301.

1 Introduction

Rewriting is an abstract model of computation. Among other things, it studies important properties including the following:

$$\text{CR}: \quad \forall s \forall t \forall u\,(s \to^* t \wedge s \to^* u \implies t \downarrow u) \qquad \text{confluence}$$

$$\text{SCR}: \quad \forall s \forall t \forall u\,(s \to t \wedge s \to u \implies \exists v\ (t \to^= v \wedge u \to^* v)) \qquad \text{strong confluence}$$

$$\text{WCR}: \quad \forall s \forall t \forall u\,(s \to t \wedge s \to u \implies t \downarrow u) \qquad \text{local confluence}$$

$$\text{NFP}: \quad \forall s \forall t \forall u\,(s \to^* t \wedge s \to^! u \implies t \to^! u) \qquad \text{normal form property}$$

$$\text{UNR}: \quad \forall s \forall t \forall u\,(s \to^! t \wedge s \to^! u \implies t = u) \qquad \text{unique normal forms with respect to reduction}$$

$$\text{UNC}: \quad \forall t \forall u\,(t \leftrightarrow^* u \wedge \text{NF}(t) \wedge \text{NF}(u) \implies t = u) \qquad \text{unique normal forms with respect to conversion}$$

We also consider the following properties involving two TRSs $\mathcal{R}$ and $\mathcal{S}$:

$$\text{COM}: \quad \forall s \forall t \forall u\,(s \to^*_{\mathcal{R}} t \wedge s \to^*_{\mathcal{S}} u \implies \exists v\,(t \to^*_{\mathcal{S}} v \wedge u \to^*_{\mathcal{R}} v)) \qquad \text{commutation}$$

$$\text{CE}: \quad \forall s \forall t\,(s \leftrightarrow^*_{\mathcal{R}} t \iff s \leftrightarrow^*_{\mathcal{S}} t) \qquad \text{conversion equivalence}$$

$$\text{NE}: \quad \forall s \forall t\,(s \to^!_{\mathcal{R}} t \iff s \to^!_{\mathcal{S}} t) \qquad \text{normalization equivalence}$$

An interesting observation is that for each of these properties there exists a rewrite system that satisfies the property when restricted to ground terms but not when arbitrary terms are allowed. Consider the left-linear right-ground TRS $\mathcal{R}$ consisting of the rules

$$\mathrm{a} \to \mathrm{b} \qquad \mathrm{f}(\mathrm{a}, x) \to \mathrm{b} \qquad \mathrm{f}(\mathrm{b}, \mathrm{b}) \to \mathrm{b}$$

over the signature $\mathcal{F} = \{\mathrm{a}, \mathrm{b}, \mathrm{f}\}$. It is ground-confluent because every ground term in $\mathcal{T}(\mathcal{F})$ rewrites to b. Confluence does not hold; the term $\mathrm{f}(\mathrm{a}, x)$ rewrites to the different normal forms b and $\mathrm{f}(\mathrm{b}, x)$.

In this AFP entry, properties on arbitrary terms are reduced to the corresponding properties on ground terms, for left-linear right-ground rewrite systems and for linear variable-separated systems. To do this, I formalized fundamental term rewriting operations that include the root step and the one step rewriting relations. Also, I added definitions for conversion equivalence, normalization equivalence, strong confluence and the normal from property extending the list of important rewriting properties of the AFP entry "Abstract Rewriting" [1].

Rewrite sequences that contain a root step play an important role in the formalization. The table of contents should give the reader a good overview of the content of this entry.

2 Preliminaries

theory *Terms-Positions*
imports *Regular-Tree-Relations.Ground-Terms*
begin

2.1 Additional operations on terms and positions

2.1.1 Linearity

fun *linear-term* :: ('*f*, '*v*) *term* ⇒ *bool* **where**
linear-term (*Var* -) = *True* |
linear-term (*Fun* - *ts*) = (*is-partition* (*map vars-term ts*) ∧ (∀ *t*∈*set ts*. *linear-term t*))
abbreviation *linear-sys* $\mathcal{R}$ ≡ ∀ (*l*, *r*) ∈ $\mathcal{R}$. *linear-term l* ∧ *linear-term r*

2.1.2 Positions induced by contexts, by variables and by given subterms

definition *possc C* = {*p* | *p t*. *p* ∈ *poss C*⟨*t*⟩}
definition *varposs s* = {*p* | *p*. *p* ∈ *poss s* ∧ *is-Var* (*s* |- *p*)}
definition *poss-of-term u t* = {*p*. *p* ∈ *poss t* ∧ *t* |- *p* = *u*}

2.1.3 Replacing functions symbols that aren't specified in the signature by variables

definition *funas-rel* $\mathcal{R}$ = (⋃ (*l*, *r*) ∈ $\mathcal{R}$. *funas-term l* ∪ *funas-term r*)

fun *term-to-sig* **where**
term-to-sig $\mathcal{F}$ *v* (*Var x*) = *Var x*
| *term-to-sig* $\mathcal{F}$ *v* (*Fun f ts*) =
(*if* (*f*, *length ts*) ∈ $\mathcal{F}$ *then Fun f* (*map* (*term-to-sig* $\mathcal{F}$ *v*) *ts*) *else Var v*)

fun *ctxt-well-def-hole-path* **where**
ctxt-well-def-hole-path $\mathcal{F}$ *Hole* ⟷ *True*
| *ctxt-well-def-hole-path* $\mathcal{F}$ (*More f ss C ts*) ⟷ (*f*, *Suc* (*length ss* + *length ts*)) ∈ $\mathcal{F}$ ∧ *ctxt-well-def-hole-path* $\mathcal{F}$ *C*

fun *inv-const-ctxt* **where**
inv-const-ctxt $\mathcal{F}$ *v Hole* = *Hole*
| *inv-const-ctxt* $\mathcal{F}$ *v* ((*More f ss C ts*))
= (*More f* (*map* (*term-to-sig* $\mathcal{F}$ *v*) *ss*) (*inv-const-ctxt* $\mathcal{F}$ *v C*) (*map* (*term-to-sig* $\mathcal{F}$ *v*) *ts*))

fun *inv-const-ctxt'* **where**
inv-const-ctxt' $\mathcal{F}$ *v Hole* = *Var v*
| *inv-const-ctxt'* $\mathcal{F}$ *v* ((*More f ss C ts*))
= (*if* (*f*, *Suc* (*length ss* + *length ts*)) ∈ $\mathcal{F}$ *then Fun f* (*map* (*term-to-sig* $\mathcal{F}$ *v*) *ss* @ *inv-const-ctxt'* $\mathcal{F}$ *v C* # *map* (*term-to-sig* $\mathcal{F}$ *v*) *ts*) *else Var v*)

2.1.4 Replace term at a given position in contexts

fun *replace-term-context-at* :: ('*f*, '*v*) *ctxt* ⇒ *pos* ⇒ ('*f*, '*v*) *term* ⇒ ('*f*, '*v*) *ctxt*
(-[- ← -]$_C$ [*1000*, *0*] *1000*) **where**
replace-term-context-at □ *p u* = □
| *replace-term-context-at* (*More f ss C ts*) (*i* # *ps*) *u* =
(*if i* < *length ss then More f* (*ss*[*i* := (*ss* ! *i*)[*ps* ← *u*]]) *C ts*
else if i = *length ss then More f ss* (*replace-term-context-at C ps u*) *ts*
else More f ss C (*ts*[(*i* − *Suc* (*length ss*)) := (*ts* ! (*i* − *Suc* (*length ss*)))[*ps* ← *u*]]))

abbreviation *constT c* ≡ *Fun c* []

2.1.5 Multihole context closure of a term relation as inductive set

definition *all-ctxt-closed* **where**
all-ctxt-closed F r ⟷ (∀ *f ts ss*. (*f*, *length ss*) ∈ *F* ⟶ *length ts* = *length ss* ⟶
(∀ *i*. *i* < *length ts* ⟶ (*ts* ! *i*, *ss* ! *i*) ∈ *r*) ⟶
(∀ *i*. *i* < *length ts* ⟶ *funas-term* (*ts* ! *i*) ∪ *funas-term* (*ss* ! *i*) ⊆ *F*) ⟶ (*Fun f ts*, *Fun f ss*) ∈ *r*) ∧
(∀ *x*. (*Var x*, *Var x*) ∈ *r*)

2.2 Destruction and introduction of *all-ctxt-closed*

lemma *all-ctxt-closedD*: *all-ctxt-closed F r* ⟹ (*f*,*length ss*) ∈ *F* ⟹ *length ts* = *length ss*
⟹ ⟦⋀ *i*. *i* < *length ts* ⟹ (*ts* ! *i*, *ss* ! *i*) ∈ *r* ⟧
⟹ ⟦⋀ *i*. *i* < *length ts* ⟹ *funas-term* (*ts* ! *i*) ⊆ *F* ⟧
⟹ ⟦⋀ *i*. *i* < *length ts* ⟹ *funas-term* (*ss* ! *i*) ⊆ *F* ⟧
⟹ (*Fun f ts*, *Fun f ss*) ∈ *r*
unfolding *all-ctxt-closed-def* **by** *auto*

lemma *trans-ctxt-sig-imp-all-ctxt-closed*: **assumes** *tran*: *trans r*
and *refl*: ⋀ *t*. *funas-term t* ⊆ *F* ⟹ (*t*,*t*) ∈ *r*
and *ctxt*: ⋀ *C s t*. *funas-ctxt C* ⊆ *F* ⟹ *funas-term s* ⊆ *F* ⟹ *funas-term t* ⊆ *F* ⟹ (*s*,*t*) ∈ *r* ⟹ (*C* ⟨ *s* ⟩, *C* ⟨ *t* ⟩) ∈ *r*
shows *all-ctxt-closed F r* **unfolding** *all-ctxt-closed-def*
proof (*rule*, *intro allI impI*)
fix *f ts ss*
assume *f*: (*f*,*length ss*) ∈ *F* **and**
l: *length ts* = *length ss* **and**
steps: ∀ *i* < *length ts*. (*ts* ! *i*, *ss* ! *i*) ∈ *r* **and**
sig: ∀ *i* < *length ts*. *funas-term* (*ts* ! *i*) ∪ *funas-term* (*ss* ! *i*) ⊆ *F*
from *sig* **have** *sig-ts*: ⋀ *t*. *t* ∈ *set ts* ⟹ *funas-term t* ⊆ *F* **unfolding** *set-conv-nth* **by** *auto*
let *?p* = λ *ss*. (*Fun f ts*, *Fun f ss*) ∈ *r* ∧ *funas-term* (*Fun f ss*) ⊆ *F*
let *?r* = λ *xsi ysi*. (*xsi*, *ysi*) ∈ *r* ∧ *funas-term ysi* ⊆ *F*
have *init*: *?p ts* **by** (*rule conjI*[*OF refl*], *insert f sig-ts l*, *auto*)
have *?p ss*

proof (*rule parallel-list-update*[**where** p = *?p* **and** r = *?r*, *OF - HOL.refl init l*[*symmetric*]])
fix *xs i y*
assume *len*: *length xs* = *length ts*
and *i*: $i <$ *length ts*
and *r*: *?r* (*xs* ! *i*) *y*
and *p*: *?p xs*
let *?C* = *More f* (*take i xs*) *Hole* (*drop* (*Suc i*) *xs*)
have *id1*: *Fun f xs* = *?C* ⟨ *xs* ! *i*⟩ **using** *id-take-nth-drop*[*OF i*[*folded len*]] **by** *simp*
have *id2*: *Fun f* (*xs*[*i* := *y*]) = *?C* ⟨ *y* ⟩ **using** *upd-conv-take-nth-drop*[*OF i*[*folded len*]] **by** *simp*
from *p*[*unfolded id1*] **have** *C*: *funas-ctxt ?C* ⊆ *F* **and** *xi*: *funas-term* (*xs* ! *i*) ⊆ *F* **by** *auto*
from *r* **have** *funas-term y* ⊆ *F* (*xs* ! *i*, *y*) ∈ *r* **by** *auto*
with *ctxt*[*OF C xi this*] *C* **have** *r*: (*Fun f xs*, *Fun f* (*xs*[*i* := *y*])) ∈ *r*
and *f*: *funas-term* (*Fun f* (*xs*[*i* := *y*])) ⊆ *F* **unfolding** *id1 id2* **by** *auto*
from *p r tran* **have** (*Fun f ts*, *Fun f* (*xs*[*i* := *y*])) ∈ *r* **unfolding** *trans-def* **by** *auto*
with *f*
show *?p* (*xs*[*i* := *y*]) **by** *auto*
qed (*insert sig steps*, *auto*)
then show (*Fun f ts*, *Fun f ss*) ∈ *r* **..**
qed (*insert refl*, *auto*)

2.3 Lemmas for *poss* and ordering of positions

lemma *subst-poss-mono*: *poss s* ⊆ *poss* (*s* · *σ*)
by (*induct s*) *force*+

lemma *par-pos-prefix* [*simp*]:
(*i* # *p*) ⊥ (*i* # *q*) ⟹ *p* ⊥ *q*
by (*simp add*: *par-Cons-iff*)

lemma *pos-diff-itself* [*simp*]: *p* −$_p$ *p* = []
by (*simp add*: *pos-diff-def*)

lemma *pos-les-eq-append-diff* [*simp*]:
p ≤$_p$ *q* ⟹ *p* @ (*q* −$_p$ *p*) = *q*
by (*metis option.sel pos-diff-def position-less-eq-def remove-prefix-append*)

lemma *pos-diff-append-itself* [*simp*]: (*p* @ *q*) −$_p$ *p* = *q*
by (*simp add*: *pos-diff-def remove-prefix-append*)

lemma *poss-pos-diffI*:
p ≤$_p$ *q* ⟹ *q* ∈ *poss s* ⟹ *q* −$_p$ *p* ∈ *poss* (*s* |- *p*)
using *poss-append-poss* **by** *fastforce*

lemma *less-eq-poss-append-itself* [*simp*]: *p* ≤$_p$ (*p* @ *q*)

using *position-less-eq-def* **by** *blast*

lemma *poss-ctxt-apply* [*simp*]:
hole-pos C @ *p* ∈ *poss C*⟨*s*⟩ ⟷ *p* ∈ *poss s*
by (*induct C*) *auto*

lemma *pos-replace-at-pres*:
p ∈ *poss s* ⟹ *p* ∈ *poss s*[*p* ← *t*]
proof (*induct p arbitrary*: *s*)
case (*Cons i p*)
show *?case* **using** *Cons*(*1*)[*of args s* ! *i*] *Cons*(*2*−)
by (*cases s*) *auto*
qed *auto*

lemma *par-pos-replace-pres*:
p ∈ *poss s* ⟹ *p* ⊥ *q* ⟹ *p* ∈ *poss s*[*q* ← *t*]
proof (*induct p arbitrary*: *s q*)
case (*Cons i p*)
show *?case* **using** *Cons*(*1*)[*of args s* ! *i tl q*] *Cons*(*2*−)
by (*cases s*; *cases q*) (*auto simp add*: *nth-list-update par-Cons-iff*)
qed *auto*

lemma *poss-of-termE* [*elim*]:
assumes *p* ∈ *poss-of-term u s*
and *p* ∈ *poss s* ⟹ *s* |- *p* = *u* ⟹ *P*
shows *P* **using** *assms* **unfolding** *poss-of-term-def*
by *blast*

lemma *poss-of-term-Cons*:
i # *p* ∈ *poss-of-term u* (*Fun f ts*) ⟹ *p* ∈ *poss-of-term u* (*ts* ! *i*)
unfolding *poss-of-term-def* **by** *auto*

lemma *poss-of-term-const-ctxt-apply*:
assumes *p* ∈ *poss-of-term* (*constT c*) *C*⟨*s*⟩
shows *p* ⊥ (*hole-pos C*) ∨ (*hole-pos C*) ≤$_p$ *p* **using** *assms*
proof (*induct p arbitrary*: *C*)
case *Nil* **then show** *?case*
by (*cases C*) *auto*
next
case (*Cons i p*) **then show** *?case*
by (*cases C*) (*fastforce simp add*: *par-Cons-iff dest*!: *poss-of-term-Cons*)+
qed

2.4 Lemmas for (|-) and *replace-term-at*

lemma *subt-at-append-dist*:
p @ *q* ∈ *poss s* ⟹ *s* |- (*p* @ *q*) = (*s* |- *p*) |- *q*
proof (*induct p arbitrary*: *s*)
case (*Cons i p*) **then show** *?case*

by (*cases s*) *auto*
qed *auto*

lemma *ctxt-apply-term-subt-at-hole-pos* [*simp*]:
C⟨*s*⟩ |- (*hole-pos C* @ *q*) = *s* |- *q*
by (*induct C*) *auto*

lemma *subst-subt-at-dist*:
p ∈ *poss s* ⟹ *s* · *σ* |- *p* = *s* |- *p* · *σ*
proof (*induct p arbitrary*: *s*)
case (*Cons i p*) **then show** *?case*
by (*cases s*) *auto*
qed *auto*

lemma *replace-term-at-subt-at-id* [*simp*]: *s*[*p* ← (*s* |- *p*)] = *s*
proof (*induct p arbitrary*: *s*)
case (*Cons i p*) **then show** *?case*
by (*cases s*) *auto*
qed *auto*

lemma *replace-term-at-same-pos* [*simp*]:
s[*p* ← *u*][*p* ← *t*] = *s*[*p* ← *t*]
using *position-less-refl replace-term-at-above* **by** *blast*

— Replacement at under substitution
lemma *subt-at-vars-term*:
p ∈ *poss s* ⟹ *s* |- *p* = *Var x* ⟹ *x* ∈ *vars-term s*
by (*metis UnCI ctxt-at-pos-subt-at-id term.set-intros*(*3*) *vars-term-ctxt-apply*)

lemma *linear-term-varposs-subst-replace-term*:
linear-term s ⟹ *p* ∈ *varposs s* ⟹ *p* ≤$_p$ *q* ⟹
(*s* · *σ*)[*q* ← *u*] = *s* · (λ *x*. *if Var x* = *s* |- *p then* (*σ x*)[*q* −$_p$ *p* ← *u*] *else* (*σ x*))
proof (*induct q arbitrary*: *s p*)
case (*Cons i q*)
show *?case* **using** *Cons*(*1*)[*of args s* ! *i tl p*] *Cons*(*2*−)
by (*cases s*) (*auto simp*: *varposs-def nth-list-update term-subst-eq-conv*
is-partition-alt is-partition-alt-def disjoint-iff subt-at-vars-term intro!: *nth-equalityI*)
qed (*auto simp*: *varposs-def*)

— Replacement at context parallel to the hole position
lemma *par-hole-pos-replace-term-context-at*:
p ⊥ *hole-pos C* ⟹ *C*⟨*s*⟩[*p* ← *u*] = (*C*[*p* ← *u*]$_C$)⟨*s*⟩
proof (*induct p arbitrary*: *C*)
case (*Cons i p*)
from *Cons*(*2*) **obtain** *f ss D ts* **where** [*simp*]: *C* = *More f ss D ts* **by** (*cases C*)
auto
show *?case* **using** *Cons*(*1*)[*of D*] *Cons*(*2*)
by (*auto simp*: *list-update-append nth-append-Cons minus-nat.simps*(*2*) *split*:

nat.splits)
qed *auto*

lemma *par-pos-replace-term-at*:
  $p \in poss\ s \Longrightarrow p \perp q \Longrightarrow s[q \leftarrow t]\ |\text{-}\ p = s\ |\text{-}\ p$
proof (*induct p arbitrary*: *s q*)
  case (*Cons i p*)
  show *?case* **using** *Cons*(*1*)[*of args s* ! *i tl q*] *Cons*(*2*−)
    by (*cases s*; *cases q*) (*auto*, *metis nth-list-update par-Cons-iff*)
qed *auto*

lemma *less-eq-subt-at-replace*:
  $p \in poss\ s \Longrightarrow p \leq_p q \Longrightarrow s[q \leftarrow t]\ |\text{-}\ p = (s\ |\text{-}\ p)[q -_p p \leftarrow t]$
proof (*induct p arbitrary*: *s q*)
  case (*Cons i p*)
  show *?case* **using** *Cons*(*1*)[*of args s* ! *i tl q*] *Cons*(*2*−)
    by (*cases s*; *cases q*) *auto*
qed *auto*

lemma *greater-eq-subt-at-replace*:
  $p \in poss\ s \Longrightarrow q \leq_p p \Longrightarrow s[q \leftarrow t]\ |\text{-}\ p = t\ |\text{-}\ (p -_p q)$
proof (*induct p arbitrary*: *s q*)
  case (*Cons i p*)
  show *?case* **using** *Cons*(*1*)[*of args s* ! *i tl q*] *Cons*(*2*−)
    by (*cases s*; *cases q*) *auto*
qed *auto*

lemma *replace-subterm-at-itself* [*simp*]:
  $s[p \leftarrow (s\ |\text{-}\ p)[q \leftarrow t]] = s[p\ @\ q \leftarrow t]$
proof (*induct p arbitrary*: *s*)
  case (*Cons i p*)
  show *?case* **using** *Cons*(*1*)[*of args s* ! *i*]
    by (*cases s*) *auto*
qed *auto*

lemma *hole-pos-replace-term-at* [*simp*]:
  $hole\text{-}pos\ C \leq_p p \Longrightarrow C\langle s\rangle[p \leftarrow u] = C\langle s[p -_p hole\text{-}pos\ C \leftarrow u]\rangle$
proof (*induct C arbitrary*: *p*)
  case (*More f ss C ts*) **then show** *?case*
    by (*cases p*) *auto*
qed *auto*

lemma *ctxt-of-pos-term-apply-replace-at-ident*:
  assumes $p \in poss\ s$
  shows $(ctxt\text{-}at\text{-}pos\ s\ p)\langle t\rangle = s[p \leftarrow t]$
  using *assms*

proof (*induct p arbitrary*: *s*)
case (*Cons i p*)
show *?case* **using** *Cons*(*1*)[*of args s* ! *i*] *Cons*(*2*−)
by (*cases s*) (*auto simp*: *nth-append-Cons intro*!: *nth-equalityI*)
qed *auto*

lemma *ctxt-apply-term-replace-term-hole-pos* [*simp*]:
$C\langle s\rangle$[*hole-pos C* @ *q* ← *u*] = $C\langle s$[*q* ← *u*]$\rangle$
by (*simp add*: *pos-diff-def position-less-eq-def remove-prefix-append*)

lemma *ctxt-apply-subt-at-hole-pos* [*simp*]: $C\langle s\rangle$ |- *hole-pos C* = *s*
by (*induct C*) *auto*

lemma *subt-at-imp-supteq′*:
assumes *p* ∈ *poss s* **and** *s*|-*p* = *t* **shows** *s* ⊵ *t* **using** *assms*
proof (*induct p arbitrary*: *s*)
case (*Cons i p*)
from *Cons*(*2*−) **show** *?case* **using** *Cons*(*1*)[*of args s* ! *i*]
by (*cases s*) *force*+
qed *auto*

lemma *subt-at-imp-supteq*:
assumes *p* ∈ *poss s* **shows** *s* ⊵ *s*|-*p*
proof −
have *s*|-*p* = *s*|-*p* **by** *auto*
with *assms* **show** *?thesis* **by** (*rule subt-at-imp-supteq′*)
qed

2.5 *term-to-sig* invariants and distributions

lemma *fuans-term-term-to-sig* [*simp*]: *funas-term* (*term-to-sig* $\mathcal{F}$ *v t*) ⊆ $\mathcal{F}$
by (*induct t*) *auto*

lemma *term-to-sig-id* [*simp*]:
funas-term t ⊆ $\mathcal{F}$ ⟹ *term-to-sig* $\mathcal{F}$ *v t* = *t*
by (*induct t*) (*auto simp add*: *UN-subset-iff map-idI*)

lemma *term-to-sig-subst-sig* [*simp*]:
funas-term t ⊆ $\mathcal{F}$ ⟹ *term-to-sig* $\mathcal{F}$ *v* (*t* · *σ*) = *t* · (λ *x*. *term-to-sig* $\mathcal{F}$ *v* (*σ x*))
by (*induct t*) *auto*

lemma *funas-ctxt-ctxt-inv-const-ctxt-ind* [*simp*]:
funas-ctxt C ⊆ $\mathcal{F}$ ⟹ *inv-const-ctxt* $\mathcal{F}$ *v C* = *C*
by (*induct C*) (*auto simp add*: *UN-subset-iff intro*!: *nth-equalityI*)

lemma *term-to-sig-ctxt-apply* [*simp*]:
ctxt-well-def-hole-path $\mathcal{F}$ *C* ⟹ *term-to-sig* $\mathcal{F}$ *v* $C\langle s\rangle$ = (*inv-const-ctxt* $\mathcal{F}$ *v* *C*)$\langle$*term-to-sig* $\mathcal{F}$ *v s*$\rangle$
by (*induct C*) *auto*

lemma *term-to-sig-ctxt-apply'* [*simp*]:
  ¬ *ctxt-well-def-hole-path* $\mathcal{F}$ *C* ⟹ *term-to-sig* $\mathcal{F}$ *v* *C*⟨*s*⟩ = *inv-const-ctxt'* $\mathcal{F}$ *v* *C*
  by (*induct C*) *auto*

lemma *funas-ctxt-ctxt-well-def-hole-path*:
  funas-ctxt C ⊆ $\mathcal{F}$ ⟹ *ctxt-well-def-hole-path* $\mathcal{F}$ *C*
  by (*induct C*) *auto*

2.6 Misc

lemma *funas-term-subt-at*:
  (*f*, *n*) ∈ *funas-term t* ⟹ (∃ *p ts*. *p* ∈ *poss t* ∧ *t* |- *p* = *Fun f ts* ∧ *length ts* = *n*)
proof (*induct t*)
  case (*Fun g ts*) **note** *IH* = *this*
  show *?case*
  proof (*cases g* = *f* ∧ *length ts* = *n*)
    case *False*
    then obtain *i* **where** *i*: *i* < *length ts* (*f*, *n*) ∈ *funas-term* (*ts* ! *i*) **using** *IH*(*2*)
      using *in-set-idx* **by** *force*
    from *IH*(*1*)[*OF nth-mem*[*OF this*(*1*)] *this*(*2*)] **show** *?thesis* **using** *i*(*1*)
      by (*metis poss-Cons-poss subt-at.simps*(*2*) *term.sel*(*4*))
  qed *auto*
qed *simp*

lemma *finite-poss*: *finite* (*poss s*)
proof (*induct s*)
  case (*Fun f ts*)
  have *poss* (*Fun f ts*) = *insert* [] (⋃ (*set* (*map2* (λ *i p*. ((#) *i*) ‘ *p*) [*0*..< *length ts*] (*map poss ts*))))
    by (*auto simp*: *image-iff set-zip split*: *prod.splits*)
  then show *?case* **using** *Fun*
    by (*auto simp del*: *poss.simps dest*!: *set-zip-rightD*)
qed *simp*

lemma *finite-varposs*: *finite* (*varposs s*)
  by (*intro finite-subset*[*of varposs s poss s*]) (*auto simp*: *varposs-def finite-poss*)

lemma *gound-linear* [*simp*]: *ground t* ⟹ *linear-term t*
  by (*induct t*) (*auto simp*: *is-partition-alt is-partition-alt-def*)

declare *ground-substI*[*intro*, *simp*]
lemma *ground-ctxt-substI*:
  (⋀ *x*. *x* ∈ *vars-ctxt C* ⟹ *ground* (*σ x*)) ⟹ *ground-ctxt* (*C* ·$_c$ *σ*)
  by (*induct C*) *auto*

lemma *funas-ctxt-subst-apply-ctxt*:
  funas-ctxt (*C* ·$_c$ *σ*) = *funas-ctxt C* ∪ (⋃ (*funas-term* ‘ *σ* ‘ *vars-ctxt C*))

proof (*induct C*)
  case (*More f ss C ts*)
  then show *?case*
    by (*fastforce simp add*: *funas-term-subst*)
qed *simp*

lemma *varposs-Var*[*simp*]:
  varposs (*Var x*) = {[]}
  by (*auto simp*: *varposs-def*)

lemma *varposs-Fun*[*simp*]:
  varposs (*Fun f ts*) = { *i* # *p*| *i p*. *i* < *length ts* ∧ *p* ∈ *varposs* (*ts* ! *i*)}
  by (*auto simp*: *varposs-def*)

lemma *vars-term-varposs-iff*:
  x ∈ *vars-term s* ⟷ (∃ *p* ∈ *varposs s*. *s* |- *p* = *Var x*)
proof (*induct s*)
  case (*Fun f ts*)
  show *?case* **using** *Fun*[*OF nth-mem*]
    by (*force simp*: *in-set-conv-nth Bex-def*)
qed *auto*

lemma *vars-term-empty-ground*:
  vars-term s = {} ⟹ *ground s*
  by (*metis equals0D ground-substI subst-ident*)

lemma *ground-subst-apply*: *ground t* ⟹ *t* · *σ* = *t*
  by (*induct t*) (*auto intro*: *nth-equalityI*)

lemma *varposs-imp-poss*:
  p ∈ *varposs s* ⟹ *p* ∈ *poss s* **by** (*auto simp*: *varposs-def*)

lemma *varposs-empty-gound*:
 varposs s = {} ⟷ *ground s*
  by (*induct s*) (*fastforce simp*: *in-set-conv-nth*)+

lemma *funas-term-subterm-atI* [*intro*]:
  p ∈ *poss s* ⟹ *funas-term s* ⊆ *F* ⟹ *funas-term* (*s* |- *p*) ⊆ *F*
  by (*metis ctxt-at-pos-subt-at-id funas-ctxt-apply le-sup-iff*)

lemma *varposs-ground-replace-at*:
  p ∈ *varposs s* ⟹ *ground u* ⟹ *varposs s*[*p* ← *u*] = *varposs s* − {*p*}
proof (*induct p arbitrary*: *s*)
  case *Nil* **then show** *?case*
    by (*cases s*) (*auto simp*: *varposs-empty-gound*)
next
  case (*Cons i p*)
  from *Cons*(*2*) **obtain** *f ts* **where** [*simp*]: *s* = *Fun f ts* **by** (*cases s*) *auto*
  from *Cons*(*2*) **have** *var*: *p* ∈ *varposs* (*ts* ! *i*) **by** *auto*

from *Cons(1)*[*OF var Cons(3)*] **have** $j < \mathit{length}\ ts \Longrightarrow \{j \# q|\ q.\ q \in \mathit{varposs}\ (ts[i := (ts\ !\ i)[p \leftarrow u]]\ !\ j)\} =$
$\{j \# q\ |q.\ q \in \mathit{varposs}\ (ts\ !\ j)\} - \{i \# p\}$ **for** j
by (*cases* $j = i$) (*auto simp add*: *nth-list-update*)
then show *?case* **by** *auto blast*
qed

lemma *funas-term-replace-at-upper*:
$\mathit{funas\text{-}term}\ s[p \leftarrow t] \subseteq \mathit{funas\text{-}term}\ s \cup \mathit{funas\text{-}term}\ t$
proof (*induct p arbitrary*: *s*)
case (*Cons i p*)
show *?case* **using** *Cons(1)*[*of args s ! i*]
by (*cases s*) (*fastforce simp*: *in-set-conv-nth nth-list-update split*!: *if-splits*)+
qed *simp*

lemma *funas-term-replace-at-lower*:
$p \in \mathit{poss}\ s \Longrightarrow \mathit{funas\text{-}term}\ t \subseteq \mathit{funas\text{-}term}\ (s[p \leftarrow t])$
proof (*induct p arbitrary*: *s*)
case (*Cons i p*)
show *?case* **using** *Cons(1)*[*of args s ! i*] *Cons(2−)*
by (*cases s*) (*fastforce simp*: *in-set-conv-nth nth-list-update split*!: *if-splits*)+
qed *simp*

lemma *poss-of-term-possI* [*intro*!]:
$p \in \mathit{poss}\ s \Longrightarrow s\ |\text{-}\ p = u \Longrightarrow p \in \mathit{poss\text{-}of\text{-}term}\ u\ s$
unfolding *poss-of-term-def* **by** *blast*

lemma *poss-of-term-replace-term-at*:
$p \in \mathit{poss}\ s \Longrightarrow p \in \mathit{poss\text{-}of\text{-}term}\ u\ s[p \leftarrow u]$
proof (*induct p arbitrary*: *s*)
case (*Cons i p*) **then show** *?case*
by (*cases s*) (*auto simp*: *poss-of-term-def*)
qed *auto*

lemma *constT-nfunas-term-poss-of-term-empty*:
$(c,\ 0) \notin \mathit{funas\text{-}term}\ t \longleftrightarrow \mathit{poss\text{-}of\text{-}term}\ (\mathit{constT}\ c)\ t = \{\}$
unfolding *poss-of-term-def*
using *funas-term-subt-at*[*of c 0 t*]
using *funas-term-subterm-atI*[**where** *?F* =*funas-term t* **and** *?s* = *t*, *THEN subsetD*]
by *auto*

lemma *poss-of-term-poss-emptyD*:
assumes *poss-of-term u s* = {}
shows $p \in \mathit{poss}\ s \Longrightarrow s\ |\text{-}\ p \neq u$ **using** *assms*
unfolding *poss-of-term-def* **by** *blast*

lemma *possc-subt-at-ctxt-apply*:
$p \in \mathit{possc}\ C \Longrightarrow p \perp \mathit{hole\text{-}pos}\ C \Longrightarrow C\langle s\rangle\ |\text{-}\ p = C\langle t\rangle\ |\text{-}\ p$

proof (*induct p arbitrary*: *C*)
case (*Cons i p*)
have [*dest*]: *length ss # p* ∈ *possc* (*More f ss D ts*) ⟹ *p* ∈ *possc D* **for** *f ss D ts*
by (*auto simp*: *possc-def*)
show *?case* **using** *Cons*
by (*cases C*) (*auto simp*: *nth-append-Cons*)
qed *simp*

end

3 Rewriting

theory *Rewriting*
imports *Terms-Positions*
begin

3.1 Basic rewrite definitions

3.1.1 Rewrite steps with implicit signature declaration (encoded in the type)

inductive-set *rrstep* :: ('*f*, '*v*) *term rel* ⇒ ('*f*, '*v*) *term rel* **for** $\mathcal{R}$ **where**
[*intro*]: (*l*, *r*) ∈ $\mathcal{R}$ ⟹ (*l* · *σ*, *r* · *σ*) ∈ *rrstep* $\mathcal{R}$

inductive-set *rstep* :: ('*f*, '*v*) *term rel* ⇒ ('*f*, '*v*) *term rel* **for** $\mathcal{R}$ **where**
(*s*, *t*) ∈ *rrstep* $\mathcal{R}$ ⟹ (*C*⟨*s*⟩, *C*⟨*t*⟩) ∈ *rstep* $\mathcal{R}$

3.1.2 Restrict relations to terms induced by a given signature

definition *sig-step* $\mathcal{F}$ $\mathcal{R}$ = *Restr* $\mathcal{R}$ (*Collect* (λ *s*. *funas-term s* ⊆ $\mathcal{F}$))

3.1.3 Rewriting under a given signature/restricted to ground terms

abbreviation *srrstep* $\mathcal{F}$ $\mathcal{R}$ ≡ *sig-step* $\mathcal{F}$ (*rrstep* $\mathcal{R}$)
abbreviation *srstep* $\mathcal{F}$ $\mathcal{R}$ ≡ *sig-step* $\mathcal{F}$ (*rstep* $\mathcal{R}$)
abbreviation *gsrstep* $\mathcal{F}$ $\mathcal{R}$ ≡ *Restr* (*sig-step* $\mathcal{F}$ (*rstep* $\mathcal{R}$)) (*Collect ground*)

3.1.4 Rewriting sequences involving a root step

abbreviation (*input*) *relto* :: '*a rel* ⇒ '*a rel* ⇒ '*a rel* **where**
relto R S ≡ *S*^* *O R O S*^*
definition *srsteps-with-root-step* $\mathcal{F}$ $\mathcal{R}$ ≡ *relto* (*sig-step* $\mathcal{F}$ (*rrstep* $\mathcal{R}$)) (*srstep* $\mathcal{F}$ $\mathcal{R}$)

3.2 Monotonicity laws

lemma *Restr-mono*: *Restr r A* ⊆ *r* **by** *auto*

lemma *Restr-trancl-mono-set*: (*Restr r A*)$^+$ ⊆ *A* × *A*

by (*simp add*: *trancl-subset-Sigma*)

lemma *rrstep-rstep-mono*: *rrstep* $\mathcal{R} \subseteq$ *rstep* $\mathcal{R}$
by (*auto intro*: *rstep.intros*[**where** *?C* = $\square$, *simplified*])

lemma *sig-step-mono*:
$\mathcal{F} \subseteq \mathcal{G} \Longrightarrow$ *sig-step* $\mathcal{F}$ $\mathcal{R} \subseteq$ *sig-step* $\mathcal{G}$ $\mathcal{R}$
by (*auto simp*: *sig-step-def*)

lemma *sig-step-mono2*:
$\mathcal{R} \subseteq \mathcal{L} \Longrightarrow$ *sig-step* $\mathcal{F}$ $\mathcal{R} \subseteq$ *sig-step* $\mathcal{F}$ $\mathcal{L}$
by (*auto simp*: *sig-step-def*)

lemma *srrstep-monp*:
$\mathcal{F} \subseteq \mathcal{G} \Longrightarrow$ *srrstep* $\mathcal{F}$ $\mathcal{R} \subseteq$ *srrstep* $\mathcal{G}$ $\mathcal{R}$
by (*simp add*: *sig-step-mono*)

lemma *srstep-monp*:
$\mathcal{F} \subseteq \mathcal{G} \Longrightarrow$ *srstep* $\mathcal{F}$ $\mathcal{R} \subseteq$ *srstep* $\mathcal{G}$ $\mathcal{R}$
by (*simp add*: *sig-step-mono*)

lemma *srsteps-monp*:
$\mathcal{F} \subseteq \mathcal{G} \Longrightarrow$ (*srstep* $\mathcal{F}$ $\mathcal{R}$)$^+ \subseteq$ (*srstep* $\mathcal{G}$ $\mathcal{R}$)$^+$
by (*simp add*: *sig-step-mono trancl-mono-set*)

lemma *srsteps-eq-monp*:
$\mathcal{F} \subseteq \mathcal{G} \Longrightarrow$ (*srstep* $\mathcal{F}$ $\mathcal{R}$)$^* \subseteq$ (*srstep* $\mathcal{G}$ $\mathcal{R}$)*
by (*meson rtrancl-mono sig-step-mono subrelI subsetD trancl-into-rtrancl*)

lemma *srsteps-with-root-step-sig-mono*:
$\mathcal{F} \subseteq \mathcal{G} \Longrightarrow$ *srsteps-with-root-step* $\mathcal{F}$ $\mathcal{R} \subseteq$ *srsteps-with-root-step* $\mathcal{G}$ $\mathcal{R}$
unfolding *srsteps-with-root-step-def*
by (*simp add*: *relcomp-mono srrstep-monp srsteps-eq-monp*)

3.3 Introduction, elimination, and destruction rules for *sig-step*, *rstep*, *rrstep*, *srrstep*, and *srstep*

lemma *sig-stepE* [*elim*, *consumes 1*]:
(s, t) $\in$ *sig-step* $\mathcal{F}$ $\mathcal{R} \Longrightarrow$ $[\![(s, t) \in \mathcal{R} \Longrightarrow$ *funas-term* $s \subseteq \mathcal{F} \Longrightarrow$ *funas-term* $t \subseteq \mathcal{F} \Longrightarrow P]\!] \Longrightarrow P$
by (*auto simp*: *sig-step-def*)

lemma *sig-stepI* [*intro*]:
funas-term $s \subseteq \mathcal{F} \Longrightarrow$ *funas-term* $t \subseteq \mathcal{F} \Longrightarrow (s, t) \in \mathcal{R} \Longrightarrow (s, t) \in$ *sig-step* $\mathcal{F}$ $\mathcal{R}$
by (*auto simp*: *sig-step-def*)

lemma *rrstep-subst* [*elim*, *consumes 1*]:
assumes (s, t) $\in$ *rrstep* $\mathcal{R}$

obtains $l\ r\ \sigma$ **where** $(l,\ r) \in \mathcal{R}\ s = l \cdot \sigma\ t = r \cdot \sigma$ **using** *assms*
by (*meson rrstep.simps*)

lemma *rstep-imp-C-s-r*:
assumes $(s,\ t) \in rstep\ \mathcal{R}$
shows $\exists\, C\ \sigma\ l\ r.\ (l,r) \in \mathcal{R} \wedge s = C\langle l{\cdot}\sigma\rangle \wedge t = C\langle r{\cdot}\sigma\rangle$**using** *assms*
by (*metis rrstep.cases rstep.simps*)

lemma *rstep-imp-C-s-r'* [*elim*, *consumes 1*]:
assumes $(s,\ t) \in rstep\ \mathcal{R}$
obtains $C\ l\ r\ \sigma$ **where** $(l,r) \in \mathcal{R}\ s = C\langle l{\cdot}\sigma\rangle\ t = C\langle r{\cdot}\sigma\rangle$ **using** *assms*
using *rstep-imp-C-s-r* **by** *blast*

lemma *rrstep-basicI* [*intro*]:
$(l,\ r) \in \mathcal{R} \Longrightarrow (l,\ r) \in rrstep\ \mathcal{R}$
by (*metis rrstepp.intros rrstepp-rrstep-eq subst-apply-term-empty*)

lemma *rstep-ruleI* [*intro*]:
$(l,\ r) \in \mathcal{R} \Longrightarrow (l,\ r) \in rstep\ \mathcal{R}$
using *rrstep-rstep-mono* **by** *blast*

lemma *rstepI* [*intro*]:
$(l,\ r) \in \mathcal{R} \Longrightarrow s = C\langle l \cdot \sigma\rangle \Longrightarrow t = C\langle r \cdot \sigma\rangle \Longrightarrow (s,\ t) \in rstep\ \mathcal{R}$
by (*simp add*: *rrstep.intros rstep.intros*)

lemma *rstep-substI* [*intro*]:
$(s,\ t) \in rstep\ \mathcal{R} \Longrightarrow (s \cdot \sigma,\ t \cdot \sigma) \in rstep\ \mathcal{R}$
by (*auto elim*!: *rstep-imp-C-s-r' simp flip*: *subst-subst-compose*)

lemma *rstep-ctxtI* [*intro*]:
$(s,\ t) \in rstep\ \mathcal{R} \Longrightarrow (C\langle s\rangle,\ C\langle t\rangle) \in rstep\ \mathcal{R}$
by (*auto elim*!: *rstep-imp-C-s-r' simp flip*: *ctxt-ctxt-compose*)

lemma *srrstepD*:
$(s,\ t) \in srrstep\ \mathcal{F}\ \mathcal{R} \Longrightarrow (s,\ t) \in rrstep\ \mathcal{R} \wedge funas\text{-}term\ s \subseteq \mathcal{F} \wedge funas\text{-}term\ t \subseteq \mathcal{F}$
by (*auto simp*: *sig-step-def*)

lemma *srstepD*:
$(s,\ t) \in (srstep\ \mathcal{F}\ \mathcal{R}) \Longrightarrow (s,\ t) \in rstep\ \mathcal{R} \wedge funas\text{-}term\ s \subseteq \mathcal{F} \wedge funas\text{-}term\ t \subseteq \mathcal{F}$
by (*auto simp*: *sig-step-def*)

lemma *srstepsD*:
$(s,\ t) \in (srstep\ \mathcal{F}\ \mathcal{R})^+ \Longrightarrow (s,\ t) \in (rstep\ \mathcal{R})^+ \wedge funas\text{-}term\ s \subseteq \mathcal{F} \wedge funas\text{-}term\ t \subseteq \mathcal{F}$
unfolding *sig-step-def* **using** *trancl-mono-set*[*OF Restr-mono*]
by (*auto simp*: *sig-step-def dest*: *subsetD*[*OF Restr-trancl-mono-set*])

3.3.1 Transitive and relfexive closure distribution over *sig-step*

lemma *funas-rel-converse*:
funas-rel $\mathcal{R} \subseteq \mathcal{F} \Longrightarrow$ *funas-rel* $(\mathcal{R}^{-1}) \subseteq \mathcal{F}$ **unfolding** *funas-rel-def*
by *auto*

lemma *rstep-term-to-sig-r*:
assumes $(s, t) \in$ *rstep* $\mathcal{R}$ **and** *funas-rel* $\mathcal{R} \subseteq \mathcal{F}$ **and** *funas-term* $s \subseteq \mathcal{F}$
shows $(s,$ *term-to-sig* $\mathcal{F}$ v $t) \in$ *rstep* $\mathcal{R}$
proof −
from *assms*(*1*) **obtain** C l r σ **where**
$*$: $s = C\langle l \cdot \sigma\rangle$ $t = C\langle r \cdot \sigma\rangle$ $(l, r) \in \mathcal{R}$ **by** *auto*
from *assms*(*2*, *3*) $*(3)$ **have** *funas-ctxt* $C \subseteq \mathcal{F}$ *funas-term* $l \subseteq \mathcal{F}$ *funas-term* $r \subseteq \mathcal{F}$
by (*auto simp*: $*(1)$ *funas-rel-def funas-term-subst subset-eq*)
then have (*term-to-sig* $\mathcal{F}$ v s, *term-to-sig* $\mathcal{F}$ v $t) \in$ *rstep* $\mathcal{R}$ **using** $*(3)$
by (*auto simp*: $*(1, 2)$ *funas-ctxt-ctxt-well-def-hole-path*)
then show *?thesis* **using** *assms*(*3*) **by** *auto*
qed

lemma *rstep-term-to-sig-l*:
assumes $(s, t) \in$ *rstep* $\mathcal{R}$ **and** *funas-rel* $\mathcal{R} \subseteq \mathcal{F}$ **and** *funas-term* $t \subseteq \mathcal{F}$
shows (*term-to-sig* $\mathcal{F}$ v s, $t) \in$ *rstep* $\mathcal{R}$
proof −
from *assms*(*1*) **obtain** C l r σ **where**
$*$: $s = C\langle l \cdot \sigma\rangle$ $t = C\langle r \cdot \sigma\rangle$ $(l, r) \in \mathcal{R}$ **by** *auto*
from *assms*(*2*, *3*) $*(3)$ **have** *funas-ctxt* $C \subseteq \mathcal{F}$ *funas-term* $l \subseteq \mathcal{F}$ *funas-term* $r \subseteq \mathcal{F}$
by (*auto simp*: $*(2)$ *funas-rel-def funas-term-subst subset-eq*)
then have (*term-to-sig* $\mathcal{F}$ v s, *term-to-sig* $\mathcal{F}$ v $t) \in$ *rstep* $\mathcal{R}$ **using** $*(3)$
by (*auto simp*: $*(1, 2)$ *funas-ctxt-ctxt-well-def-hole-path*)
then show *?thesis* **using** *assms*(*3*) **by** *auto*
qed

lemma *rstep-trancl-sig-step-r*:
assumes $(s, t) \in$ (*rstep* $\mathcal{R})^+$ **and** *funas-rel* $\mathcal{R} \subseteq \mathcal{F}$ **and** *funas-term* $s \subseteq \mathcal{F}$
shows $(s,$ *term-to-sig* $\mathcal{F}$ v $t) \in$ (*srstep* $\mathcal{F}$ $\mathcal{R})^+$ **using** *assms*
proof (*induct*)
case (*base* t)
then show *?case* **using** *subsetD*[*OF fuans-term-term-to-sig*, *of* - $\mathcal{F}$ v]
by (*auto simp*: *rstep-term-to-sig-r sig-step-def intro*!: *r-into-trancl*)
next
case (*step* t u)
then have *st*: $(s,$ *term-to-sig* $\mathcal{F}$ v $t) \in$ (*srstep* $\mathcal{F}$ $\mathcal{R})^+$ **by** *auto*
from *step*(*2*) **obtain** C l r σ **where**
$*$: $t = C\langle l \cdot \sigma\rangle$ $u = C\langle r \cdot \sigma\rangle$ $(l, r) \in \mathcal{R}$ **by** *auto*
show *?case*
proof (*cases ctxt-well-def-hole-path* $\mathcal{F}$ C)
case *True*
from $*(3)$ *step*(*4*) **have** *funas-term* $l \subseteq \mathcal{F}$ *funas-term* $r \subseteq \mathcal{F}$ **by** (*auto simp*:

funas-rel-def)
    then have (*term-to-sig* $\mathcal{F}$ *v t*, *term-to-sig* $\mathcal{F}$ *v u*) ∈ *rstep* $\mathcal{R}$
      using *True step*(*2*) ∗(*3*) **unfolding** ∗
      by *auto*
    then have (*term-to-sig* $\mathcal{F}$ *v t*, *term-to-sig* $\mathcal{F}$ *v u*) ∈ *srstep* $\mathcal{F}$ $\mathcal{R}$
      by (*auto simp*:- *sig-step-def*)
    then show *?thesis* **using** *st* **by** *auto*
  next
    case *False*
    then have *term-to-sig* $\mathcal{F}$ *v t* = *term-to-sig* $\mathcal{F}$ *v u* **unfolding** ∗ **by** *auto*
    then show *?thesis* **using** *st* **by** *auto*
  qed
qed

lemma *rstep-trancl-sig-step-l*:
  assumes (*s*, *t*) ∈ (*rstep* $\mathcal{R}$)$^+$ **and** *funas-rel* $\mathcal{R}$ ⊆ $\mathcal{F}$ **and** *funas-term t* ⊆ $\mathcal{F}$
  shows (*term-to-sig* $\mathcal{F}$ *v s*, *t*) ∈ (*srstep* $\mathcal{F}$ $\mathcal{R}$)$^+$ **using** *assms*
proof (*induct rule*: *converse-trancl-induct*)
  case (*base t*)
  then show *?case* **using** *subsetD*[*OF fuans-term-term-to-sig*, *of* - $\mathcal{F}$ *v*]
    by (*auto simp*: *rstep-term-to-sig-l sig-step-def intro*!: *r-into-trancl*)
next
  case (*step s u*)
  then have *st*: (*term-to-sig* $\mathcal{F}$ *v u*, *t*) ∈ (*srstep* $\mathcal{F}$ $\mathcal{R}$)$^+$ **by** *auto*
  from *step*(*1*) **obtain** *C l r* σ **where**
    ∗: *s* = *C*⟨*l* · σ⟩ *u* = *C*⟨*r* · σ⟩ (*l*, *r*) ∈ $\mathcal{R}$ **by** *auto*
  show *?case*
  proof (*cases ctxt-well-def-hole-path* $\mathcal{F}$ *C*)
    case *True*
    from ∗(*3*) *step*(*4*) **have** *funas-term l* ⊆ $\mathcal{F}$ *funas-term r* ⊆ $\mathcal{F}$ **by** (*auto simp*:
funas-rel-def)
    then have (*term-to-sig* $\mathcal{F}$ *v s*, *term-to-sig* $\mathcal{F}$ *v u*) ∈ *rstep* $\mathcal{R}$
      using *True step*(*2*) ∗(*3*) **unfolding** ∗
      by *auto*
    then have (*term-to-sig* $\mathcal{F}$ *v s*, *term-to-sig* $\mathcal{F}$ *v u*) ∈ *srstep* $\mathcal{F}$ $\mathcal{R}$
      by (*auto simp*:- *sig-step-def*)
    then show *?thesis* **using** *st* **by** *auto*
  next
    case *False*
    then have *term-to-sig* $\mathcal{F}$ *v s* = *term-to-sig* $\mathcal{F}$ *v u* **unfolding** ∗ **by** *auto*
    then show *?thesis* **using** *st* **by** *auto*
  qed
qed

lemma *rstep-srstepI* [*intro*]:
  funas-rel $\mathcal{R}$ ⊆ $\mathcal{F}$ ⟹ *funas-term s* ⊆ $\mathcal{F}$ ⟹ *funas-term t* ⊆ $\mathcal{F}$ ⟹ (*s*, *t*) ∈ *rstep*
$\mathcal{R}$ ⟹ (*s*, *t*) ∈ *srstep* $\mathcal{F}$ $\mathcal{R}$
  by *blast*

lemma *rsteps-srstepsI* [*intro*]:
funas-rel $\mathcal{R} \subseteq \mathcal{F} \Longrightarrow$ *funas-term* $s \subseteq \mathcal{F} \Longrightarrow$ *funas-term* $t \subseteq \mathcal{F} \Longrightarrow (s, t) \in (\mathit{rstep}\ \mathcal{R})^+ \Longrightarrow (s, t) \in (\mathit{srstep}\ \mathcal{F}\ \mathcal{R})^+$
using *rstep-trancl-sig-step-r*[*of s t* $\mathcal{R}$ $\mathcal{F}$]
by *auto*

lemma *rsteps-eq-srsteps-eqI* [*intro*]:
funas-rel $\mathcal{R} \subseteq \mathcal{F} \Longrightarrow$ *funas-term* $s \subseteq \mathcal{F} \Longrightarrow$ *funas-term* $t \subseteq \mathcal{F} \Longrightarrow (s, t) \in (\mathit{rstep}\ \mathcal{R})^* \Longrightarrow (s, t) \in (\mathit{srstep}\ \mathcal{F}\ \mathcal{R})^*$
by (*auto simp add*: *rtrancl-eq-or-trancl*)

lemma *rsteps-eq-relcomp-srsteps-eq-relcompI* [*intro*]:
assumes *funas-rel* $\mathcal{R} \subseteq \mathcal{F}$ *funas-rel* $\mathcal{S} \subseteq \mathcal{F}$
and *funas*: *funas-term* $s \subseteq \mathcal{F}$ *funas-term* $t \subseteq \mathcal{F}$
and *steps*: $(s, t) \in (\mathit{rstep}\ \mathcal{R})^*\ O\ (\mathit{rstep}\ \mathcal{S})^*$
shows $(s, t) \in (\mathit{srstep}\ \mathcal{F}\ \mathcal{R})^*\ O\ (\mathit{srstep}\ \mathcal{F}\ \mathcal{S})^*$
proof −
from *steps* **obtain** u **where** $(s, u) \in (\mathit{rstep}\ \mathcal{R})^*\ (u, t) \in (\mathit{rstep}\ \mathcal{S})^*$ **by** *auto*
then have $(s,\ \mathit{term\text{-}to\text{-}sig}\ \mathcal{F}\ v\ u) \in (\mathit{srstep}\ \mathcal{F}\ \mathcal{R})^*\ (\mathit{term\text{-}to\text{-}sig}\ \mathcal{F}\ v\ u,\ t) \in (\mathit{srstep}\ \mathcal{F}\ \mathcal{S})^*$
using *rstep-trancl-sig-step-l*[*OF - assms*(*2*) *funas*(*2*), *of u v*]
using *rstep-trancl-sig-step-r*[*OF - assms*(*1*) *funas*(*1*), *of u v*] *funas*
by (*auto simp*: *rtrancl-eq-or-trancl*)
then show *?thesis* **by** *auto*
qed

3.3.2 Distributivity laws

lemma *rstep-smycl-dist*:
$(\mathit{rstep}\ \mathcal{R})^{\leftrightarrow} = \mathit{rstep}\ (\mathcal{R}^{\leftrightarrow})$
by (*auto simp*: *sig-step-def*)

lemma *sig-step-symcl-dist*:
$(\mathit{sig\text{-}step}\ \mathcal{F}\ \mathcal{R})^{\leftrightarrow} = \mathit{sig\text{-}step}\ \mathcal{F}\ (\mathcal{R}^{\leftrightarrow})$
by (*auto simp*: *sig-step-def*)

lemma *srstep-symcl-dist*:
$(\mathit{srstep}\ \mathcal{F}\ \mathcal{R})^{\leftrightarrow} = \mathit{srstep}\ \mathcal{F}\ (\mathcal{R}^{\leftrightarrow})$
by (*auto simp*: *sig-step-def*)

lemma *Restr-smycl-dist*:
$(\mathit{Restr}\ \mathcal{R}\ \mathcal{A})^{\leftrightarrow} = \mathit{Restr}\ (\mathcal{R}^{\leftrightarrow})\ \mathcal{A}$
by *auto*

lemmas *rew-symcl-inwards* = *rstep-smycl-dist sig-step-symcl-dist srstep-symcl-dist Restr-smycl-dist*
lemmas *rew-symcl-outwards* = *rew-symcl-inwards*[*symmetric*]

lemma *rstep-converse-dist*:
(*rstep* $\mathcal{R}$)$^{-1}$ = *rstep* ($\mathcal{R}^{-1}$)
by *auto*

lemma *srrstep-converse-dist*:
(*srrstep* $\mathcal{F}$ $\mathcal{R}$)$^{-1}$ = *srrstep* $\mathcal{F}$ ($\mathcal{R}^{-1}$)
by (*fastforce simp*: *sig-step-def*)

lemma *sig-step-converse-rstep*:
(*srstep* $\mathcal{F}$ $\mathcal{R}$)$^{-1}$ = *sig-step* $\mathcal{F}$ ((*rstep* $\mathcal{R}$)$^{-1}$)
by (*meson converse.simps set-eq-subset sig-stepE*(*1*) *sig-stepE sig-stepI subrelI*)

lemma *srstep-converse-dist*:
(*srstep* $\mathcal{F}$ $\mathcal{R}$)$^{-1}$ = *srstep* $\mathcal{F}$ ($\mathcal{R}^{-1}$)
by (*auto simp*: *sig-step-def*)

lemma *Restr-converse*: (*Restr* $\mathcal{R}$ *A*)$^{-1}$ = *Restr* ($\mathcal{R}^{-1}$) *A*
by *auto*

lemmas *rew-converse-inwards* = *rstep-converse-dist srrstep-converse-dist sig-step-converse-rstep srstep-converse-dist Restr-converse trancl-converse*[*symmetric*] *rtrancl-converse*[*symmetric*]
lemmas *rew-converse-outwards* = *rew-converse-inwards*[*symmetric*]

lemma *sig-step-rsteps-dist*:
funas-rel $\mathcal{R}$ $\subseteq$ $\mathcal{F}$ $\Longrightarrow$ *sig-step* $\mathcal{F}$ ((*rstep* $\mathcal{R}$)$^+$) = (*srstep* $\mathcal{F}$ $\mathcal{R}$)$^+$
by (*auto elim*!: *sig-stepE dest*: *srstepsD*)

lemma *sig-step-rsteps-eq-dist*:
funas-rel $\mathcal{R}$ $\subseteq$ $\mathcal{F}$ $\Longrightarrow$ *sig-step* $\mathcal{F}$ ((*rstep* $\mathcal{R}$)$^+$) $\cup$ *Id* = (*srstep* $\mathcal{F}$ $\mathcal{R}$)*
by (*auto simp*: *rtrancl-eq-or-trancl sig-step-rsteps-dist*)

lemma *sig-step-conversion-dist*:
(*srstep* $\mathcal{F}$ $\mathcal{R}$)$^{\leftrightarrow *}$ = (*srstep* $\mathcal{F}$ ($\mathcal{R}^{\leftrightarrow}$))*
by (*auto simp*: *rtrancl-eq-or-trancl sig-step-rsteps-dist conversion-def srstep-symcl-dist*)

lemma *gsrstep-conversion-dist*:
(*gsrstep* $\mathcal{F}$ $\mathcal{R}$)$^{\leftrightarrow *}$ = (*gsrstep* $\mathcal{F}$ ($\mathcal{R}^{\leftrightarrow}$))*
by (*auto simp*: *conversion-def rew-symcl-inwards*)

lemma *sig-step-grstep-dist*:
gsrstep $\mathcal{F}$ $\mathcal{R}$ = *sig-step* $\mathcal{F}$ (*Restr* (*rstep* $\mathcal{R}$) (*Collect ground*))
by (*auto simp*: *sig-step-def*)

3.4 Substitution closure of *srstep*

lemma *srstep-subst-closed*:
assumes (*s*, *t*) $\in$ *srstep* $\mathcal{F}$ $\mathcal{R}$ $\bigwedge$ *x*. *funas-term* (σ *x*) $\subseteq$ $\mathcal{F}$
shows (*s* $\cdot$ σ, *t* $\cdot$ σ) $\in$ *srstep* $\mathcal{F}$ $\mathcal{R}$ **using** *assms*
by (*auto simp*: *sig-step-def funas-term-subst*)

lemma *srsteps-subst-closed*:
assumes $(s,\ t) \in (srstep\ \mathcal{F}\ \mathcal{R})^+ \bigwedge x.\ funas\text{-}term\ (\sigma\ x) \subseteq \mathcal{F}$
shows $(s \cdot \sigma,\ t \cdot \sigma) \in (srstep\ \mathcal{F}\ \mathcal{R})^+$ **using** *assms*(*1*)
proof (*induct rule*: *trancl.induct*)
case (*r-into-trancl s t*) **show** *?case*
using *srstep-subst-closed*[*OF r-into-trancl assms*(*2*)]
by *auto*
next
case (*trancl-into-trancl s t u*)
from *trancl-into-trancl*(*2*) **show** *?case*
using *srstep-subst-closed*[*OF trancl-into-trancl*(*3*) *assms*(*2*)]
by (*meson rtrancl-into-trancl1 trancl-into-rtrancl*)
qed

lemma *srsteps-eq-subst-closed*:
assumes $(s,\ t) \in (srstep\ \mathcal{F}\ \mathcal{R})^* \bigwedge x.\ funas\text{-}term\ (\sigma\ x) \subseteq \mathcal{F}$
shows $(s \cdot \sigma,\ t \cdot \sigma) \in (srstep\ \mathcal{F}\ \mathcal{R})^*$ **using** *assms srsteps-subst-closed*
by (*metis rtrancl-eq-or-trancl*)

lemma *srsteps-eq-subst-relcomp-closed*:
assumes $(s,\ t) \in (srstep\ \mathcal{F}\ \mathcal{R})^*\ O\ (srstep\ \mathcal{F}\ \mathcal{S})^* \bigwedge x.\ funas\text{-}term\ (\sigma\ x) \subseteq \mathcal{F}$
shows $(s \cdot \sigma,\ t \cdot \sigma) \in (srstep\ \mathcal{F}\ \mathcal{R})^*\ O\ (srstep\ \mathcal{F}\ \mathcal{S})^*$
proof −
from *assms*(*1*) **obtain** *u* **where** $(s,\ u) \in (srstep\ \mathcal{F}\ \mathcal{R})^*\ (u,\ t) \in (srstep\ \mathcal{F}\ \mathcal{S})^*$
by *auto*
then have $(s \cdot \sigma,\ u \cdot \sigma) \in (srstep\ \mathcal{F}\ \mathcal{R})^*\ (u \cdot \sigma,\ t \cdot \sigma) \in (srstep\ \mathcal{F}\ \mathcal{S})^*$
using *assms srsteps-eq-subst-closed*
by *metis+*
then show *?thesis* **by** *auto*
qed

3.5 Context closure of *srstep*

lemma *srstep-ctxt-closed*:
assumes $funas\text{-}ctxt\ C \subseteq \mathcal{F}$ **and** $(s,\ t) \in srstep\ \mathcal{F}\ \mathcal{R}$
shows $(C\langle s\rangle,\ C\langle t\rangle) \in srstep\ \mathcal{F}\ \mathcal{R}$ **using** *assms*
by (*intro sig-stepI*) (*auto dest*: *srstepD*)

lemma *srsteps-ctxt-closed*:
assumes $funas\text{-}ctxt\ C \subseteq \mathcal{F}$ **and** $(s,\ t) \in (srstep\ \mathcal{F}\ \mathcal{R})^+$
shows $(C\langle s\rangle,\ C\langle t\rangle) \in (srstep\ \mathcal{F}\ \mathcal{R})^+$ **using** *assms*(*2*) *srstep-ctxt-closed*[*OF assms*(*1*)]
by (*induct*) *force+*

lemma *srsteps-eq-ctxt-closed*:
assumes $funas\text{-}ctxt\ C \subseteq \mathcal{F}$ **and** $(s,\ t) \in (srstep\ \mathcal{F}\ \mathcal{R})^*$
shows $(C\langle s\rangle,\ C\langle t\rangle) \in (srstep\ \mathcal{F}\ \mathcal{R})^*$ **using** *srsteps-ctxt-closed*[*OF assms*(*1*)] *assms*(*2*)

by (*metis rtrancl-eq-or-trancl*)

lemma *sig-steps-join-ctxt-closed*:
assumes *funas-ctxt* $C \subseteq \mathcal{F}$ **and** $(s, t) \in (srstep\ \mathcal{F}\ \mathcal{R})^{\downarrow}$
shows $(C\langle s\rangle, C\langle t\rangle) \in (srstep\ \mathcal{F}\ \mathcal{R})^{\downarrow}$ **using** *srsteps-eq-ctxt-closed*[*OF assms*(*1*)] *assms*(*2*)
unfolding *join-def rew-converse-inwards*
by *auto*

The following lemma shows that every rewrite sequence either contains a root step or is root stable

lemma *nsrsteps-with-root-step-step-on-args*:
assumes $(s, t) \in (srstep\ \mathcal{F}\ \mathcal{R})^{+}$ $(s, t) \notin srsteps\text{-}with\text{-}root\text{-}step\ \mathcal{F}\ \mathcal{R}$
shows $\exists\ f\ ss\ ts.\ s = Fun\ f\ ss \wedge t = Fun\ f\ ts \wedge length\ ss = length\ ts \wedge$
$(\forall\ i < length\ ts.\ (ss\ !\ i, ts\ !\ i) \in (srstep\ \mathcal{F}\ \mathcal{R})^{*})$ **using** *assms*
proof (*induct*)
case (*base t*)
obtain *C l r* σ **where** [*simp*]: $s = C\langle l \cdot \sigma\rangle$ $t = C\langle r \cdot \sigma\rangle$ **and** *r*: $(l, r) \in \mathcal{R}$
using *base*(*1*) **unfolding** *sig-step-def*
by *blast*
then have *funas*: *funas-ctxt* $C \subseteq \mathcal{F}$ *funas-term* $(l \cdot \sigma) \subseteq \mathcal{F}$ *funas-term* $(r \cdot \sigma) \subseteq \mathcal{F}$
using *base*(*1*) **by** (*auto simp*: *sig-step-def*)
from *funas*(*2*−) *r* **have** $(l \cdot \sigma, r \cdot \sigma) \in srrstep\ \mathcal{F}\ \mathcal{R}$
by (*auto simp*: *sig-step-def*)
then have $C = Hole \Longrightarrow False$ **using** *base*(*2*) *r*
by (*auto simp*: *srsteps-with-root-step-def*)
then obtain *f ss D ts* **where** [*simp*]: $C = More\ f\ ss\ D\ ts$ **by** (*cases C*) *auto*
have $(D\langle l \cdot \sigma\rangle, D\langle r \cdot \sigma\rangle) \in (srstep\ \mathcal{F}\ \mathcal{R})$ **using** *base*(*1*) *r funas*
by (*auto simp*: *sig-step-def*)
then show *?case* **using** *funas* **by** (*auto simp*: *nth-append-Cons*)
next
case (*step t u*) **show** *?case*
proof (*cases* $(s, t) \in srsteps\text{-}with\text{-}root\text{-}step\ \mathcal{F}\ \mathcal{R} \vee (t, u) \in sig\text{-}step\ \mathcal{F}\ (rrstep\ \mathcal{R})$)
case *True* **then show** *?thesis* **using** *step*(*1*, *2*, *4*)
by (*auto simp add*: *relcomp3-I rtrancl.rtrancl-into-rtrancl srsteps-with-root-step-def*)
next
case *False*
obtain *C l r* σ **where** $*$[*simp*]: $t = C\langle l \cdot \sigma\rangle$ $u = C\langle r \cdot \sigma\rangle$ **and** *r*: $(l, r) \in \mathcal{R}$
using *step*(*2*) **unfolding** *sig-step-def* **by** *blast*
then have *funas*: *funas-ctxt* $C \subseteq \mathcal{F}$ *funas-term* $(l \cdot \sigma) \subseteq \mathcal{F}$ *funas-term* $(r \cdot \sigma) \subseteq \mathcal{F}$
using *step*(*2*) **by** (*auto simp*: *sig-step-def*)
from *False* **have** $C \neq Hole$ **using** *funas r* **by** (*force simp*: *sig-step-def*)
then obtain *f ss D ts* **where** *c*[*simp*]: $C = More\ f\ ss\ D\ ts$ **by** (*cases C*) *auto*
from *step*(*3*, *1*) *False* **obtain** *g sss tss* **where**
$**$[*simp*]: $s = Fun\ g\ sss\ t = Fun\ g\ tss$ **and** *l*: $length\ sss = length\ tss$ **and**
inv: $\forall\ i < length\ tss.\ (sss\ !\ i, tss\ !\ i) \in (srstep\ \mathcal{F}\ \mathcal{R})^{*}$

by *auto*
have [*simp*]: $g = f$ **and** *lc*: $Suc\ (length\ ss + length\ ts) = length\ sss$
using *l* ∗(*1*) **unfolding** *c* **using** ∗∗(*2*) **by** *auto*
then have $\forall\ i < Suc\ (length\ ss + length\ ts).\ ((ss\ @\ D\langle l \cdot \sigma\rangle\ \#\ ts)\ !\ i, (ss\ @\ D\langle r \cdot \sigma\rangle\ \#\ ts)\ !\ i) \in (srstep\ \mathcal{F}\ \mathcal{R})^*$
using ∗ *funas r* **by** (*auto simp*: *nth-append-Cons r-into-rtrancl rstep.intros rstepI sig-stepI*)
then have $i < length\ tss \Longrightarrow (sss\ !\ i, (ss\ @\ D\langle r \cdot \sigma\rangle\ \#\ ts)\ !\ i) \in (srstep\ \mathcal{F}\ \mathcal{R})^*$ **for** *i*
using *inv* ∗ *l lc funas* ∗∗
by (*auto simp*: *nth-append-Cons simp del*: ∗∗ ∗ *split*!: *if-splits*)
then show *?thesis* **using** *inv l lc* ∗ **unfolding** *c*
by *auto*
qed
qed

lemma *rstep-to-pos-replace*:
assumes $(s, t) \in rstep\ \mathcal{R}$
shows $\exists\ p\ l\ r\ \sigma.\ p \in poss\ s \wedge (l, r) \in \mathcal{R} \wedge s\ |\text{-}\ p = l \cdot \sigma \wedge t = s[p \leftarrow r \cdot \sigma]$
proof −
from *assms* **obtain** $C\ l\ r\ \sigma$ **where** *st*: $(l, r) \in \mathcal{R}\ s = C\langle l \cdot \sigma\rangle\ t = C\langle r \cdot \sigma\rangle$
using *rstep-imp-C-s-r* **by** *fastforce*
from *st*(*2*, *3*) **have** ∗: $t = s[hole\text{-}pos\ C \leftarrow r \cdot \sigma]$ **by** *simp*
from *this st* **show** *?thesis* **unfolding** ∗
by (*intro exI*[*of* - *hole-pos C*]) *auto*
qed

lemma *pos-replace-to-rstep*:
assumes $p \in poss\ s\ (l, r) \in \mathcal{R}$
and $s\ |\text{-}\ p = l \cdot \sigma\ t = s[p \leftarrow r \cdot \sigma]$
shows $(s, t) \in rstep\ \mathcal{R}$
using *assms*(*1*, *3*−) *replace-term-at-subt-at-id* [*of s p*]
by (*intro rstepI*[*OF assms*(*2*), *of s ctxt-at-pos s p* σ])
(*auto simp add*: *ctxt-of-pos-term-apply-replace-at-ident*)

end
theory *Replace-Constant*
imports *Rewriting*
begin

3.6 Removing/Replacing constants in a rewrite sequence that do not appear in the rewrite system

lemma *funas-term-const-subst-conv*:
$(c, 0) \notin funas\text{-}term\ l \longleftrightarrow \neg\ (l \trianglerighteq constT\ c)$
proof (*induct l*)
case (*Fun f ts*) **then show** *?case*
by *auto* (*metis Fun-supt supteq-supt-conv term.inject*(*2*))+
qed (*auto simp add*: *supteq-var-imp-eq*)

lemma *fresh-const-single-step-replace*:
assumes *lin*: *linear-sys* $\mathcal{R}$ **and** *fresh*: $(c, 0) \notin$ *funas-rel* $\mathcal{R}$
and *occ*: $p \in$ *poss-of-term* (*constT* c) s **and** *step*: $(s, t) \in$ *rstep* $\mathcal{R}$
shows $(s[p \leftarrow u], t) \in$ *rstep* $\mathcal{R} \vee$
$(\exists\ q.\ q \in$ *poss-of-term* (*constT* c) $t \wedge (s[p \leftarrow u], t[q \leftarrow u]) \in$ *rstep* $\mathcal{R})$
proof −
from *occ* **have** *const*: $p \in$ *poss* $s \wedge s$ |- $p =$ *constT* c **by** *auto*
from *step* **obtain** $C\ l\ r\ \sigma$ **where** t [*simp*]: $s = C\langle l \cdot \sigma\rangle\ t = C\langle r \cdot \sigma\rangle$
and *rule*: $(l, r) \in \mathcal{R}$ **by** *blast*
from *rule lin* **have** *lin*: *linear-term* l *linear-term* r **by** *fastforce+*
from *fresh rule* **have** *nt-lhs*: $(c, 0) \notin$ *funas-term* l **by** (*auto simp*: *funas-rel-def*)
consider (*par*) $p \perp$ (*hole-pos* C) | (*below*) *hole-pos* $C \leq_p p$ **using** *occ*
by (*auto dest*: *poss-of-term-const-ctxt-apply*)
then show *?thesis*
proof *cases*
case *par*
then have *possc*: $p \in$ *possc* C **using** *const t possc-def* **by** *blast*
then have $p \in$ *poss-of-term* (*constT* c) t $(s[p \leftarrow u], t[p \leftarrow u]) \in$ *rstep* $\mathcal{R}$
using *const par-hole-pos-replace-term-context-at*[*OF par*]
using *possc-subt-at-ctxt-apply*[*OF possc par*, *of* $r \cdot \sigma\ l \cdot \sigma$] *rule*
by *auto* (*metis par par-pos-replace-pres replace-at-hole-pos*)
then show *?thesis* **by** *blast*
next
case *below*
then obtain q **where** [*simp*]:$p =$ *hole-pos* C @ q **and** *poss*: $q \in$ *poss* $(l \cdot \sigma)$
using *const position-less-eq-def*
by (*metis* (*full-types*) *ctxt-at-pos-hole-pos ctxt-at-pos-subt-at-pos poss-append-poss t*(*1*))
have *const*: $l \cdot \sigma$ |- $q =$ *constT* c **using** *const* **by** *auto*
from *nt-lhs* **have** $\exists\ r.\ r \in$ *varposs* $l \wedge r \leq_p q$ **using** *const poss*
proof (*induct l arbitrary*: q)
case (*Var* x)
then show *?case* **by** *auto*
next
case (*Fun f ts*)
from *Fun*(*1*)[*OF nth-mem*, *of hd q tl q*] *Fun*(*2*−) **obtain** r **where**
$r \in$ *varposs* (*ts* ! *hd q*) $\wedge\ r \leq_p$ *tl q*
by (*cases q*) *auto*
then show *?case* **using** *Fun*(*2*− *4*)
by (*intro exI*[*of* - *hd q* # r]) *auto*
qed
then obtain $x\ v$ **where** *varposs*: $v \in$ *varposs* $l\ v \leq_p q\ l$ |- $v =$ *Var* x
unfolding *varposs-def* **by** *blast*
let *?*$\tau = \lambda x.$ *if Var* $x = l$ |- v *then* $(\sigma\ x)[q -_p v \leftarrow u]$ *else* $\sigma\ x$
show *?thesis*
proof (*cases* $x \in$ *vars-term* r)
case *True*
then obtain q' **where** *varposs-r*: $q' \in$ *varposs* $r\ r$ |- $q' =$ *Var* x

```
      by (metis vars-term-varposs-iff)
    have (s[p ← u], t[(hole-pos C) @ q' @ (q −p v) ← u]) ∈ rstep R
      using lin varposs rule varposs-r
      by (auto simp: linear-term-varposs-subst-replace-term intro!: rstep-ctxtI)
       (smt (verit, ccfv-SIG) pos-diff-append-itself rrstep.intros rrstep-rstep-mono
subset-eq term-subst-eq)
    moreover have (hole-pos C) @ q' @ q −p v ∈ poss-of-term (constT c) t
      using varposs-r varposs poss const poss-pos-diffI[OF varposs(2) poss]
      using subt-at-append-dist[of q' q −p v r · σ]
      by (auto simp: poss-append-poss varposs-imp-poss[THEN subst-subt-at-dist]
varposs-imp-poss[THEN subsetD[OF subst-poss-mono]])
     (metis pos-les-eq-append-diff eval-term.simps(1) subst-subt-at-dist subt-at-append-dist
varposs-imp-poss)
    ultimately show ?thesis by auto
  next
    case False
    then have [simp]: r · σ = r · ?τ using varposs
      by (auto simp add: term-subst-eq-conv)
    have (s[p ← u], t) ∈ rstep R using rule varposs lin
      by (auto simp: linear-term-varposs-subst-replace-term)
    then show ?thesis by auto
  qed
 qed
qed

lemma fresh-const-steps-replace:
 assumes lin: linear-sys R and fresh: (c, 0) ∉ funas-rel R
  and occ: p ∈ poss-of-term (constT c) s and steps: (s, t) ∈ (rstep R)+
 shows (s[p ← u], t) ∈ (rstep R)+ ∨
  (∃ q. q ∈ poss-of-term (constT c) t ∧ (s[p ← u], t[q ← u]) ∈ (rstep R)+)
 using steps occ
proof (induct arbitrary: p rule: converse-trancl-induct)
 case (base s)
 from fresh-const-single-step-replace[OF lin fresh base(2, 1)] show ?case
  by (meson r-into-trancl')
next
 case (step s t)
 from fresh-const-single-step-replace[OF lin fresh step(4, 1)]
 consider (a) (s[p ← u], t) ∈ rstep R | (b) ∃ q. q ∈ poss-of-term (constT c) t ∧
(s[p ← u], t[q ← u]) ∈ rstep R by blast
 then show ?case
 proof cases
  case a then show ?thesis using step(2)
   by auto
 next
  case b
  then obtain q where q ∈ poss-of-term (constT c) t (s[p ← u], t[q ← u]) ∈
rstep R by blast
  from step(3)[OF this(1)] this(2) show ?thesis
```

by (*metis trancl-into-trancl2*)
qed
qed

lemma *remove-const-lhs-steps*:
assumes *lin*: *linear-sys* $\mathcal{R}$ **and** *fresh*: $(c, 0) \notin$ *funas-rel* $\mathcal{R}$
and *const*: $(c, 0) \notin$ *funas-term* t
and *pos*: $p \in$ *poss-of-term* (*constT* c) s
and *steps*: $(s, t) \in (\mathit{rstep}\ \mathcal{R})^+$
shows $(s[p \leftarrow u], t) \in (\mathit{rstep}\ \mathcal{R})^+$ **using** *steps pos const fresh-const-steps-replace*
by (*metis fresh funas-term-const-subst-conv lin poss-of-termE subt-at-imp-supteq*)

Now we can show that we may remove a constant substitution

definition *const-replace-closed* **where**
const-replace-closed $c\ U = (\forall\ s\ t\ u\ p.$
$p \in$ *poss-of-term* (*constT* c) $s \longrightarrow (s, t) \in U \longrightarrow$
$(\exists\ q.\ q \in$ *poss-of-term* (*constT* c) $t \wedge (s[p \leftarrow u], t[q \leftarrow u]) \in U) \vee (s[p \leftarrow u], t) \in U)$

lemma *const-replace-closedD*:
assumes *const-replace-closed* $c\ U\ p \in$ *poss-of-term* (*constT* c) $s\ (s, t) \in U$
shows $(s[p \leftarrow u], t) \in U \vee (\exists\ q.\ q \in$ *poss-of-term* (*constT* c) $t \wedge (s[p \leftarrow u], t[q \leftarrow u]) \in U)$ **using** *assms*
unfolding *const-replace-closed-def* **by** *blast*

lemma *const-replace-closedI*:
assumes $\bigwedge s\ t\ u\ p.\ p \in$ *poss-of-term* (*constT* c) $s \Longrightarrow (s, t) \in U \Longrightarrow$
$(\exists\ q.\ q \in$ *poss-of-term* (*constT* c) $t \wedge (s[p \leftarrow u], t[q \leftarrow u]) \in U) \vee (s[p \leftarrow u], t) \in U$
shows *const-replace-closed* $c\ U$ **using** *assms*
unfolding *const-replace-closed-def*
by *auto*

abbreviation *const-subst* :: $'f \Rightarrow 'v \Rightarrow ('f, 'v)$ *Term.term* **where**
const-subst $c \equiv (\lambda\ x.$ *Fun* c [])

lemma *lin-fresh-rstep-const-replace-closed*:
linear-sys $\mathcal{R} \Longrightarrow (c, 0) \notin$ *funas-rel* $\mathcal{R} \Longrightarrow$ *const-replace-closed* c (*rstep* $\mathcal{R}$)
using *fresh-const-single-step-replace*[*of* $\mathcal{R}$ c]
by (*intro const-replace-closedI*) (*auto simp*: *constT-nfunas-term-poss-of-term-empty*, *blast*)

lemma *const-replace-closed-symcl*:
const-replace-closed $c\ U \Longrightarrow$ *const-replace-closed* $c\ (U^=)$
unfolding *const-replace-closed-def*
by (*metis Un-iff pair-in-Id-conv*)

lemma *const-replace-closed-trancl*:
const-replace-closed $c\ U \Longrightarrow$ *const-replace-closed* $c\ (U^+)$

proof (*intro const-replace-closedI*)
fix *s t u p*
assume *const*: *const-replace-closed c U* **and** *wit*: *p* ∈ *poss-of-term* (*constT c*) *s*
and *steps* :(*s*, *t*) ∈ U^+
show (∃ *q*. *q* ∈ *poss-of-term* (*constT c*) *t* ∧ (*s*[*p* ← *u*], *t*[*q* ← *u*]) ∈ U^+) ∨ (*s*[*p* ← *u*], *t*) ∈ U^+ **using** *steps wit*
proof (*induct arbitrary*: *p rule*: *converse-trancl-induct*)
case (*base s*)
show *?case* **using** *const-replace-closedD*[*OF const base*(*2*, *1*)]
by *blast*
next
case (*step s v*)
from *const-replace-closedD*[*OF const step*(*4*, *1*)]
consider (*a*) (*s*[*p* ← *u*], *v*) ∈ *U* | (*b*) ∃ *q*. *q* ∈ *poss-of-term* (*constT c*) *v* ∧ (*s*[*p* ← *u*], *v*[*q* ← *u*]) ∈ *U* **by** *auto*
then show *?case*
proof *cases*
case *a* **then show** *?thesis* **using** *step*(*2*)
by (*meson trancl-into-trancl2*)
next
case *b*
then show *?thesis* **using** *step*(*3*, *4*) **by** (*meson trancl-into-trancl2*)
qed
qed
qed

lemma *const-replace-closed-rtrancl*:
const-replace-closed c U ⟹ *const-replace-closed c* (U^*)
proof (*intro const-replace-closedI*)
fix *s t u p*
assume *const*: *const-replace-closed c U* **and** *wit*: *p* ∈ *poss-of-term* (*constT c*) *s*
and *steps* :(*s*, *t*) ∈ U^*
show (∃ *q*. *q* ∈ *poss-of-term* (*constT c*) *t* ∧ (*s*[*p* ← *u*], *t*[*q* ← *u*]) ∈ U^*) ∨ (*s*[*p* ← *u*], *t*) ∈ U^*
using *const-replace-closed-trancl*[*OF const*] *wit steps*
by (*metis const-replace-closedD rtrancl-eq-or-trancl*)
qed

lemma *const-replace-closed-relcomp*:
const-replace-closed c U ⟹ *const-replace-closed c V* ⟹ *const-replace-closed c* (*U O V*)
proof (*intro const-replace-closedI*)
fix *s t u p*
assume *const*: *const-replace-closed c U const-replace-closed c V*
and *wit*: *p* ∈ *poss-of-term* (*constT c*) *s* **and** *step*: (*s*, *t*) ∈ *U O V*
from *step* **obtain** *w* **where** *w*: (*s*, *w*) ∈ *U* (*w*, *t*) ∈ *V* **by** *auto*
from *const-replace-closedD*[*OF const*(*1*) *wit this*(*1*)]
consider (*a*) (*s*[*p* ← *u*], *w*) ∈ *U* | (*b*) (∃ *q*. *q* ∈ *poss-of-term* (*constT c*) *w* ∧ (*s*[*p* ← *u*], *w*[*q* ← *u*]) ∈ *U*)

by *auto*
then show $(\exists q.\ q \in \mathit{poss\text{-}of\text{-}term}\ (\mathit{constT}\ c)\ t \wedge (s[p \leftarrow u],\ t[q \leftarrow u]) \in U\ O\ V) \vee (s[p \leftarrow u],\ t) \in U\ O\ V$
proof *cases*
case *a*
then show *?thesis* **using** *w*(*2*) **by** *auto*
next
case *b*
then show *?thesis* **using** *const-replace-closedD*[*OF const*(*2*) - *w*(*2*)]
by (*meson relcomp.simps*)
qed
qed

const-replace-closed allow the removal of a fresh constant substitution

lemma *const-replace-closed-remove-subst-lhs*:
assumes *repcl*: *const-replace-closed c U*
and *const*: $(c,\ 0) \notin \mathit{funas\text{-}term}\ t$
and *steps*: $(s \cdot \mathit{const\text{-}subst}\ c,\ t) \in U$
shows $(s,\ t) \in U$ **using** *steps*
proof (*induct card* (*varposs s*) *arbitrary*: *s*)
case (*Suc n*)
obtain *p ps* **where** *vl*: *varposs s* = *insert p ps p* $\notin$ *ps* **using** *Suc*(*2*)
by (*metis card-le-Suc-iff dual-order.refl*)
let *?s* = *s*[*p* $\leftarrow$ *Fun c* []] **have** *vp*: *p* $\in$ *varposs s* **using** *vl* **by** *auto*
then have [*simp*]: *?s* $\cdot$ *const-subst c* = *s* $\cdot$ *const-subst c*
by (*induct s arbitrary*: *p*) (*auto simp*: *nth-list-update map-update intro*!: *nth-equalityI*)
have *varposs ?s* = *ps* **using** *vl varposs-ground-replace-at*[*of p s constT c*]
by *auto*
then have *n* = *card* (*varposs ?s*) **using** *vl Suc*(*2*) **by** (*auto simp*: *card-insert-if finite-varposs*)
from *Suc*(*1*)[*OF this*] **have** *IH*: $(s[p \leftarrow \mathit{constT}\ c],\ t) \in U\ p \in \mathit{poss\text{-}of\text{-}term}\ (\mathit{constT}\ c)\ s[p \leftarrow \mathit{constT}\ c]$
using *Suc*(*2*, *3*) *vl poss-of-term-replace-term-at varposs-imp-poss vp*
using ‹*s*[*p* $\leftarrow$ *constT c*] $\cdot$ *const-subst c* = *s* $\cdot$ *const-subst c*›
by *fastforce*+
show *?case* **using** *const-replace-closedD*[*OF repcl*] *const IH*(*2*, *1*)
by (*metis constT-nfunas-term-poss-of-term-empty empty-iff replace-term-at-same-pos replace-term-at-subt-at-id*)
qed (*auto simp*: *ground-subst-apply card-eq-0-iff finite-varposs varposs-empty-gound*)

3.6.1 Removal lemma applied to various rewrite relations

lemma *remove-const-subst-step-lhs*:
assumes *lin*: *linear-sys* $\mathcal{R}$ **and** *fresh*: $(c,\ 0) \notin \mathit{funas\text{-}rel}\ \mathcal{R}$
and *const*: $(c,\ 0) \notin \mathit{funas\text{-}term}\ t$
and *step*: $(s \cdot \mathit{const\text{-}subst}\ c,\ t) \in (\mathit{rstep}\ \mathcal{R})$
shows $(s,\ t) \in (\mathit{rstep}\ \mathcal{R})$
using *lin-fresh-rstep-const-replace-closed*[*OF lin fresh*, *THEN const-replace-closed-remove-subst-lhs*] *const step*
by *blast*

lemma *remove-const-subst-steps-lhs*:
assumes *lin*: *linear-sys* $\mathcal{R}$ **and** *fresh*: $(c,\ 0) \notin$ *funas-rel* $\mathcal{R}$
and *const*: $(c,\ 0) \notin$ *funas-term* t
and *steps*: $(s \cdot$ *const-subst* $c,\ t) \in (rstep\ \mathcal{R})^+$
shows $(s,\ t) \in (rstep\ \mathcal{R})^+$
using *lin-fresh-rstep-const-replace-closed*[*THEN const-replace-closed-trancl*,
OF lin fresh, *THEN const-replace-closed-remove-subst-lhs*]
using *const steps*
by *blast*

lemma *remove-const-subst-steps-eq-lhs*:
assumes *lin*: *linear-sys* $\mathcal{R}$ **and** *fresh*: $(c,\ 0) \notin$ *funas-rel* $\mathcal{R}$
and *const*: $(c,\ 0) \notin$ *funas-term* t
and *steps*: $(s \cdot$ *const-subst* $c,\ t) \in (rstep\ \mathcal{R})^*$
shows $(s,\ t) \in (rstep\ \mathcal{R})^*$ **using** *steps const*
by (*cases* $s = t$) (*auto simp*: *rtrancl-eq-or-trancl funas-term-subst ground-subst-apply vars-term-empty-ground*
dest: *remove-const-subst-steps-lhs*[*OF lin fresh const*] *split*: *if-splits*)

lemma *remove-const-subst-steps-rhs*:
assumes *lin*: *linear-sys* $\mathcal{R}$ **and** *fresh*: $(c,\ 0) \notin$ *funas-rel* $\mathcal{R}$
and *const*: $(c,\ 0) \notin$ *funas-term* s
and *steps*: $(s,\ t \cdot$ *const-subst* $c) \in (rstep\ \mathcal{R})^+$
shows $(s,\ t) \in (rstep\ \mathcal{R})^+$
proof −
from *steps* **have** *revs*: $(t \cdot$ *const-subst* $c,\ s) \in (rstep\ (\mathcal{R}^{-1}))^+$
unfolding *rew-converse-outwards* **by** *auto*
have $(t,\ s) \in (rstep\ (\mathcal{R}^{-1}))^+$ **using** *assms*
by (*intro remove-const-subst-steps-lhs*[*OF - - - revs*]) (*auto simp*: *funas-rel-def*)
then show *?thesis* **unfolding** *rew-converse-outwards* **by** *auto*
qed

lemma *remove-const-subst-steps-eq-rhs*:
assumes *lin*: *linear-sys* $\mathcal{R}$ **and** *fresh*: $(c,\ 0) \notin$ *funas-rel* $\mathcal{R}$
and *const*: $(c,\ 0) \notin$ *funas-term* s
and *steps*: $(s,\ t \cdot$ *const-subst* $c) \in (rstep\ \mathcal{R})^*$
shows $(s,\ t) \in (rstep\ \mathcal{R})^*$
using *steps const*
by (*cases* $s = t$) (*auto simp*: *rtrancl-eq-or-trancl funas-term-subst ground-subst-apply vars-term-empty-ground*
dest!: *remove-const-subst-steps-rhs*[*OF lin fresh const*] *split*: *if-splits*)

Main lemmas

lemma *const-subst-eq-ground-eq*:
assumes $s \cdot$ *const-subst* $c = t \cdot$ *const-subst* d $c \neq d$
and $(c,\ 0) \notin$ *funas-term* t $(d,\ 0) \notin$ *funas-term* s
shows $s = t$ **using** *assms*
proof (*induct s arbitrary*: t)

case (*Var x*) **then show** *?case* **by** (*cases t*) *auto*
next
case (*Fun f ts*)
from *Fun*(*2*−) **obtain** *g us* **where** [*simp*]: *t* = *Fun g us* **by** (*cases t*) *auto*
have [*simp*]: *g* = *f* **and** *l*: *length ts* = *length us* **using** *Fun*(*2*)
by (*auto intro*: *map-eq-imp-length-eq*)
have *i* < *length ts* ⟹ *ts* ! *i* = *us* ! *i* **for** *i*
using *Fun*(*1*)[*OF nth-mem*, *of i us* ! *i* **for** *i*] *Fun*(*2*−) *l*
by (*auto simp*: *map-eq-conv*′)
then show *?case* **using** *l*
by (*auto intro*: *nth-equalityI*)
qed

lemma *remove-const-subst-steps*:
assumes *linear-sys* $\mathcal{R}$ **and** (*c*, *0*) ∉ *funas-rel* $\mathcal{R}$ **and** (*d*, *0*) ∉ *funas-rel* $\mathcal{R}$
and *c* ≠ *d* (*c*, *0*) ∉ *funas-term t* (*d*, *0*) ∉ *funas-term s*
and (*s* · *const-subst c*, *t* · *const-subst d*) ∈ (*rstep* $\mathcal{R}$)*
shows (*s*, *t*) ∈ (*rstep* $\mathcal{R}$)*
proof (*cases s* · *const-subst c* = *t* · *const-subst d*)
case *True*
from *const-subst-eq-ground-eq*[*OF this*] *assms*(*4* − *6*) **show** *?thesis* **by** *auto*
next
case *False*
then have *step*: (*s* · *const-subst c*, *t* · *const-subst d*) ∈ (*rstep* $\mathcal{R}$)+ **using** *assms*(*7*)
by (*auto simp*: *rtrancl-eq-or-trancl*)
then have (*s*, *t* · *const-subst d*) ∈ (*rstep* $\mathcal{R}$)+ **using** *assms*
by (*intro remove-const-subst-steps-lhs*[*OF - - - step*]) (*auto simp*: *funas-term-subst*)
from *remove-const-subst-steps-rhs*[*OF - - - this*] **show** *?thesis* **using** *assms*
by *auto*
qed

lemma *remove-const-subst-relcomp-lhs*:
assumes *sys*: *linear-sys* $\mathcal{R}$ *linear-sys* $\mathcal{S}$
and *fr*: (*c*, *0*) ∉ *funas-rel* $\mathcal{R}$ **and** *fs*:(*c*, *0*) ∉ *funas-rel* $\mathcal{S}$
and *funas*: (*c*, *0*) ∉ *funas-term t*
and *seq*: (*s* · *const-subst c*, *t*) ∈ (*rstep* $\mathcal{R}$)* *O* (*rstep* $\mathcal{S}$)*
shows (*s*, *t*) ∈ (*rstep* $\mathcal{R}$)* *O* (*rstep* $\mathcal{S}$)* **using** *seq*
using *lin-fresh-rstep-const-replace-closed*[*OF sys*(*1*) *fr*, *THEN const-replace-closed-rtrancl*]
using *lin-fresh-rstep-const-replace-closed*[*OF sys*(*2*) *fs*, *THEN const-replace-closed-rtrancl*]
using *const-replace-closed-relcomp*
by (*intro const-replace-closed-remove-subst-lhs*[*OF - funas seq*]) *force*

lemma *remove-const-subst-relcomp-rhs*:
assumes *sys*: *linear-sys* $\mathcal{R}$ *linear-sys* $\mathcal{S}$
and *fr*: (*c*, *0*) ∉ *funas-rel* $\mathcal{R}$ **and** *fs*:(*c*, *0*) ∉ *funas-rel* $\mathcal{S}$
and *funas*: (*c*, *0*) ∉ *funas-term s*
and *seq*: (*s*, *t* · *const-subst c*) ∈ (*rstep* $\mathcal{R}$)* *O* (*rstep* $\mathcal{S}$)*
shows (*s*, *t*) ∈ (*rstep* $\mathcal{R}$)* *O* (*rstep* $\mathcal{S}$)*

proof −
from *seq* **have** $(t \cdot \textit{const-subst}\ c,s) \in ((\textit{rstep}\ \mathcal{R})^*\ O\ (\textit{rstep}\ \mathcal{S})^*)^{-1}$
by *auto*
then have $(t \cdot \textit{const-subst}\ c,s) \in ((\textit{rstep}\ \mathcal{S})^*)^{-1}\ O\ ((\textit{rstep}\ \mathcal{R})^*)^{-1}$
using *converse-relcomp* **by** *blast*
note *seq* = *this*[*unfolded rtrancl-converse*[*symmetric*] *rew-converse-inwards*]
from *sys fr fs* **have** *linear-sys* $(\mathcal{S}^{-1})$ *linear-sys* $(\mathcal{R}^{-1})$ $(c,\ 0) \notin \textit{funas-rel}\ (\mathcal{S}^{-1})$ $(c,\ 0) \notin \textit{funas-rel}\ (\mathcal{R}^{-1})$
by (*auto simp*: *funas-rel-def*)
from *remove-const-subst-relcomp-lhs*[*OF this funas seq*]
have $(t,\ s) \in (\textit{rstep}\ (\mathcal{S}^{-1}))^*\ O\ (\textit{rstep}\ (\mathcal{R}^{-1}))^*$ **by** *simp*
then show *?thesis*
unfolding *rew-converse-outwards converse-relcomp*[*symmetric*]
by *simp*
qed

lemma *remove-const-subst-relcomp*:
assumes *sys*: *linear-sys* $\mathcal{R}$ *linear-sys* $\mathcal{S}$
and *fr*: $(c,\ 0) \notin \textit{funas-rel}\ \mathcal{R}$ $(d,\ 0) \notin \textit{funas-rel}\ \mathcal{R}$
and *fs*:$(c,\ 0) \notin \textit{funas-rel}\ \mathcal{S}$ $(d,\ 0) \notin \textit{funas-rel}\ \mathcal{S}$
and *diff*: $c \neq d$ **and** *funas*: $(c,\ 0) \notin \textit{funas-term}\ t$ $(d,\ 0) \notin \textit{funas-term}\ s$
and *seq*: $(s \cdot \textit{const-subst}\ c,\ t \cdot \textit{const-subst}\ d) \in (\textit{rstep}\ \mathcal{R})^*\ O\ (\textit{rstep}\ \mathcal{S})^*$
shows $(s,\ t) \in (\textit{rstep}\ \mathcal{R})^*\ O\ (\textit{rstep}\ \mathcal{S})^*$
proof −
from *diff funas*(*1*) **have** ∗: $(c,\ 0) \notin \textit{funas-term}\ (t \cdot \textit{const-subst}\ d)$
by (*auto simp*: *funas-term-subst*)
show *?thesis* **using** *remove-const-subst-relcomp-rhs*[*OF sys fr*(*2*) *fs*(*2*) *funas*(*2*)
remove-const-subst-relcomp-lhs[*OF sys fr*(*1*) *fs*(*1*) ∗ *seq*]]
by *blast*
qed

end

4 Confluence related rewriting properties

theory *Rewriting-Properties*
imports *Rewriting*
Abstract−Rewriting.Abstract-Rewriting
begin

4.1 Confluence related ARS properties

definition *SCR-on r A* ≡ $(\forall\ a \in A.\ \forall\ b\ c.\ (a,\ b) \in r \wedge (a,\ c) \in r \longrightarrow$
$(\exists\ d.\ (b,\ d) \in r^{=} \wedge\ (c,\ d) \in r^*))$

abbreviation *SCR* :: *'a rel* ⇒ *bool* **where** *SCR r* ≡ *SCR-on r UNIV*

definition *NFP-on* :: *'a rel* ⇒ *'a set* ⇒ *bool* **where**

NFP-on r A $\longleftrightarrow$ ($\forall$ *a*$\in$*A*. $\forall$ *b c*. (*a*, *b*) $\in$ r^* $\wedge$ (*a*, *c*) $\in$ $r^!$ $\longrightarrow$ (*b*, *c*) $\in$ r^*)

abbreviation *NFP* :: *'a rel* $\Rightarrow$ *bool* **where** *NFP r* $\equiv$ *NFP-on r UNIV*

definition *CE-on* :: *'a rel* $\Rightarrow$ *'a rel* $\Rightarrow$ *'a set* $\Rightarrow$ *bool* **where**
CE-on r s A $\longleftrightarrow$ ($\forall$ *a*$\in$*A*. $\forall$ *b*. (*a*, *b*) $\in$ $r^{\leftrightarrow *}$ $\longleftrightarrow$ (*a*, *b*) $\in$ $s^{\leftrightarrow *}$)

abbreviation *CE* :: *'a rel* $\Rightarrow$ *'a rel* $\Rightarrow$ *bool* **where** *CE r s* $\equiv$ *CE-on r s UNIV*

definition *NE-on* :: *'a rel* $\Rightarrow$ *'a rel* $\Rightarrow$ *'a set* $\Rightarrow$ *bool* **where**
NE-on r s A $\longleftrightarrow$ ($\forall$ *a*$\in$*A*. $\forall$ *b*. (*a*, *b*) $\in$ $r^!$ $\longleftrightarrow$ (*a*, *b*) $\in$ $s^!$)

abbreviation *NE* :: *'a rel* $\Rightarrow$ *'a rel* $\Rightarrow$ *bool* **where** *NE r s* $\equiv$ *NE-on r s UNIV*

4.2 Signature closure of relation to model multihole context closure

lemma *all-ctxt-closed-sig-rsteps* [*intro*]:
fixes $\mathcal{R}$ :: (*'f*,*'v*) *term rel*
shows *all-ctxt-closed* $\mathcal{F}$ ((*srstep* $\mathcal{F}$ $\mathcal{R}$)*) (**is** *all-ctxt-closed* - (*?R**))
proof (*rule trans-ctxt-sig-imp-all-ctxt-closed*)
fix *C* :: (*'f*,*'v*) *ctxt* **and** *s t* :: (*'f*,*'v*)*term*
assume *C*: *funas-ctxt C* $\subseteq$ $\mathcal{F}$
and *s*: *funas-term s* $\subseteq$ $\mathcal{F}$
and *t*: *funas-term t* $\subseteq$ $\mathcal{F}$
and *steps*: (*s*,*t*) $\in$ *?R**
from *steps*
show (*C* $\langle$ *s* $\rangle$, *C* $\langle$ *t* $\rangle$) $\in$ *?R**
proof (*induct*)
case (*step t u*)
from *step*(*2*) **have** *tu*: (*t*,*u*) $\in$ *rstep* $\mathcal{R}$ **and** *t*: *funas-term t* $\subseteq$ $\mathcal{F}$ **and** *u*: *funas-term u* $\subseteq$ $\mathcal{F}$
by (*auto dest*: *srstepD*)
have (*C* $\langle$ *t* $\rangle$, *C* $\langle$ *u* $\rangle$) $\in$ *?R* **by** (*rule sig-stepI*[*OF* - - *rstep-ctxtI*[*OF tu*]], *insert C t u*, *auto*)
with *step*(*3*) **show** *?case* **by** *auto*
qed *auto*
qed (*auto intro*: *trans-rtrancl*)

lemma *sigstep-trancl-funas*:
(*s*, *t*) $\in$ (*srstep* $\mathcal{F}$ $\mathcal{S}$)* $\Longrightarrow$ *s* $\neq$ *t* $\Longrightarrow$ *funas-term s* $\subseteq$ $\mathcal{F}$
(*s*, *t*) $\in$ (*srstep* $\mathcal{F}$ $\mathcal{S}$)* $\Longrightarrow$ *s* $\neq$ *t* $\Longrightarrow$ *funas-term t* $\subseteq$ $\mathcal{F}$
by (*auto simp*: *rtrancl-eq-or-trancl dest*: *srstepsD*)

lemma *srrstep-to-srestep*:
(*s*, *t*) $\in$ *srrstep* $\mathcal{F}$ $\mathcal{R}$ $\Longrightarrow$ (*s*, *t*) $\in$ *srstep* $\mathcal{F}$ $\mathcal{R}$
by (*meson in-mono rrstep-rstep-mono sig-step-mono2*)

lemma *srsteps-with-root-step-srstepsD*:

$(s,\ t) \in srsteps\text{-}with\text{-}root\text{-}step\ \mathcal{F}\ \mathcal{R} \Longrightarrow (s,\ t) \in (srstep\ \mathcal{F}\ \mathcal{R})^+$
by (*auto dest*: *srrstep-to-srestep simp*: *srsteps-with-root-step-def*)

lemma *srsteps-with-root-step-sresteps-eqD*:
$(s,\ t) \in srsteps\text{-}with\text{-}root\text{-}step\ \mathcal{F}\ \mathcal{R} \Longrightarrow (s,\ t) \in (srstep\ \mathcal{F}\ \mathcal{R})^*$
by (*auto dest*: *srrstep-to-srestep simp*: *srsteps-with-root-step-def*)

lemma *symcl-srstep-conversion*:
$(s,\ t) \in srstep\ \mathcal{F}\ (\mathcal{R}^{\leftrightarrow}) \Longrightarrow (s,\ t) \in (srstep\ \mathcal{F}\ \mathcal{R})^{\leftrightarrow *}$
by (*simp add*: *conversion-def rstep-converse-dist srstep-symcl-dist*)

lemma *symcl-srsteps-conversion*:
$(s,\ t) \in (srstep\ \mathcal{F}\ (\mathcal{R}^{\leftrightarrow}))^* \Longrightarrow (s,\ t) \in (srstep\ \mathcal{F}\ \mathcal{R})^{\leftrightarrow *}$
by (*simp add*: *conversion-def rstep-converse-dist srstep-symcl-dist*)

lemma *NF-srstep-args*:
assumes $Fun\ f\ ss \in NF\ (srstep\ \mathcal{F}\ \mathcal{R})\ funas\text{-}term\ (Fun\ f\ ss) \subseteq \mathcal{F}\ i < length\ ss$
shows $ss\ !\ i \in NF\ (srstep\ \mathcal{F}\ \mathcal{R})$
proof (*rule ccontr*)
assume $ss\ !\ i \notin NF\ (srstep\ \mathcal{F}\ \mathcal{R})$
then obtain t **where** *step*: $(ss\ !\ i,\ t) \in rstep\ \mathcal{R}\ funas\text{-}term\ t \subseteq \mathcal{F}$
by (*auto simp*: *NF-def sig-step-def*)
from *assms*(*3*) **have** [*simp*]: $Suc\ (length\ ss - Suc\ 0) = length\ ss$ **by** *auto*
from *rstep-ctxtI*[*OF step*(*1*), **where** *?C* = $ctxt\text{-}at\text{-}pos\ (Fun\ f\ ss)[i]$]
have $(Fun\ f\ ss,\ Fun\ f\ (ss[i := t])) \in srstep\ \mathcal{F}\ \mathcal{R}$ **using** *step*(*2*) *assms*(*2*, *3*)
by (*auto simp*: *sig-step-def upd-conv-take-nth-drop min-def UN-subset-iff*
dest: *in-set-takeD in-set-dropD simp flip*: *id-take-nth-drop*)
then show *False* **using** *assms*(*1*)
by (*auto simp*: *NF-def*)
qed

lemma *all-ctxt-closed-srstep-conversions* [*simp*]:
all-ctxt-closed $\mathcal{F}\ ((srstep\ \mathcal{F}\ \mathcal{R})^{\leftrightarrow *})$
by (*simp add*: *all-ctxt-closed-sig-rsteps sig-step-conversion-dist*)

lemma *NFP-stepD*:
$NFP\ r \Longrightarrow (a,\ b) \in r^* \Longrightarrow (a,\ c) \in r^* \Longrightarrow c \in NF\ r \Longrightarrow (b,\ c) \in r^*$
by (*auto simp*: *NFP-on-def*)

lemma *NE-symmetric*: $NE\ r\ s \Longrightarrow NE\ s\ r$
unfolding *NE-on-def* **by** *auto*

lemma *CE-symmetric*: $CE\ r\ s \Longrightarrow CE\ s\ r$
unfolding *CE-on-def* **by** *auto*

Reducing the quantification over rewrite sequences for properties *CR* ...

to rewrite sequences containing at least one root step

lemma *all-ctxt-closed-sig-reflE*:
all-ctxt-closed $\mathcal{F}$ $\mathcal{R}$ $\Longrightarrow$ *funas-term* t $\subseteq$ $\mathcal{F}$ $\Longrightarrow$ $(t, t) \in \mathcal{R}$
proof (*induct* t)
case (*Fun* f *ts*)
from *Fun*(*1*)[*OF nth-mem Fun*(*2*)] *Fun*(*3*)
have $i <$ *length ts* $\Longrightarrow$ *funas-term* (*ts* ! i) $\subseteq$ $\mathcal{F}$ $i <$ *length ts* $\Longrightarrow$ (*ts* ! i, *ts* ! i) $\in \mathcal{R}$ **for** i
by (*auto simp*: *SUP-le-iff*)
then show *?case* **using** *all-ctxt-closedD*[*OF Fun*(*2*)] *Fun*(*3*)
by *simp*
qed (*simp add*: *all-ctxt-closed-def*)

lemma *all-ctxt-closed-relcomp* [*intro*]:
($\bigwedge$ s t. $(s, t) \in \mathcal{R}$ $\Longrightarrow$ $s \neq t$ $\Longrightarrow$ *funas-term* s $\subseteq$ $\mathcal{F}$ $\wedge$ *funas-term* t $\subseteq$ $\mathcal{F}$) $\Longrightarrow$
($\bigwedge$ s t. $(s, t) \in \mathcal{S}$ $\Longrightarrow$ $s \neq t$ $\Longrightarrow$ *funas-term* s $\subseteq$ $\mathcal{F}$ $\wedge$ *funas-term* t $\subseteq$ $\mathcal{F}$) $\Longrightarrow$
all-ctxt-closed $\mathcal{F}$ $\mathcal{R}$ $\Longrightarrow$ *all-ctxt-closed* $\mathcal{F}$ $\mathcal{S}$ $\Longrightarrow$ *all-ctxt-closed* $\mathcal{F}$ ($\mathcal{R}$ O $\mathcal{S}$)
proof −
assume *funas*:($\bigwedge$ s t. $(s, t) \in \mathcal{R}$ $\Longrightarrow$ $s \neq t$ $\Longrightarrow$ *funas-term* s $\subseteq$ $\mathcal{F}$ $\wedge$ *funas-term* t $\subseteq$ $\mathcal{F}$)
($\bigwedge$ s t. $(s, t) \in \mathcal{S}$ $\Longrightarrow$ $s \neq t$ $\Longrightarrow$ *funas-term* s $\subseteq$ $\mathcal{F}$ $\wedge$ *funas-term* t $\subseteq$ $\mathcal{F}$)
and *ctxt-cl*: *all-ctxt-closed* $\mathcal{F}$ $\mathcal{R}$ *all-ctxt-closed* $\mathcal{F}$ $\mathcal{S}$
{fix *f ss ts* **assume** *ass*: $(f,$ *length ss*$) \in \mathcal{F}$ *length ss* = *length ts* $\bigwedge$ i. $i <$ *length ts* $\Longrightarrow$ (*ss* ! i, *ts* ! i) $\in$ ($\mathcal{R}$ O $\mathcal{S}$)
$\bigwedge$ i . $i <$ *length ts* $\Longrightarrow$ *funas-term* (*ts* ! i) $\subseteq$ $\mathcal{F}$ $\bigwedge i$. $i <$ *length ts* $\Longrightarrow$ *funas-term* (*ss* ! i) $\subseteq$ $\mathcal{F}$
from *ass*(*2*, *3*) **obtain** *us* **where** *us*: *length us* = *length ts* $\bigwedge$ i. $i <$ *length ts* $\Longrightarrow$ (*ss* ! i, *us* ! i) $\in \mathcal{R}$
$\bigwedge$ i. $i <$ *length ts* $\Longrightarrow$ (*us* ! i, *ts* ! i) $\in \mathcal{S}$
using *Ex-list-of-length-P*[*of length ts* λ x i. (*ss* ! i, x) $\in \mathcal{R}$ $\wedge$ (x, *ts* ! i) $\in \mathcal{S}$]
by *auto*
from *funas* **have** *fu*: $\bigwedge$ i . $i <$ *length us* $\Longrightarrow$ *funas-term* (*us* ! i) $\subseteq$ $\mathcal{F}$ **using** *us ass*(*4*, *5*)
by (*auto simp*: *funas-rel-def*) (*metis in-mono*)
have (*Fun f ss*, *Fun f us*) $\in \mathcal{R}$ **using** *ass*(*1*, *2*, *5*) *us*(*1*, *2*) *fu*
by (*intro all-ctxt-closedD*[*OF ctxt-cl*(*1*), *of f*]) *auto*
moreover have (*Fun f us*, *Fun f ts*) $\in \mathcal{S}$ **using** *ass*(*1*, *2*, *4*) *us*(*1*, *3*) *fu*
by (*intro all-ctxt-closedD*[*OF ctxt-cl*(*2*), *of f*]) *auto*
ultimately have (*Fun f ss*, *Fun f ts*) $\in$ $\mathcal{R}$ O $\mathcal{S}$ **by** *auto***}**
moreover
{fix x **have** (*Var x*, *Var x*) $\in \mathcal{R}$ (*Var x*, *Var x*) $\in \mathcal{S}$ **using** *ctxt-cl*
by (*auto simp*: *all-ctxt-closed-def*)
then have (*Var x*, *Var x*) $\in$ $\mathcal{R}$ O $\mathcal{S}$ **by** *auto***}**
ultimately show *?thesis* **by** (*auto simp*: *all-ctxt-closed-def*)
qed

abbreviation *prop-to-rel* $P \equiv \{(s, t) |$ s t. P s $t\}$

abbreviation *prop-mctxt-cl* $\mathcal{F}$ *P* ≡ *all-ctxt-closed* $\mathcal{F}$ (*prop-to-rel P*)

lemma *prop-mctxt-cl-Var*:
prop-mctxt-cl $\mathcal{F}$ *P* ⟹ *P* (*Var x*) (*Var x*)
by (*simp add*: *all-ctxt-closed-def*)

lemma *prop-mctxt-cl-refl-on*:
prop-mctxt-cl $\mathcal{F}$ *P* ⟹ *funas-term t* ⊆ $\mathcal{F}$ ⟹ *P t t*
using *all-ctxt-closed-sig-reflE* **by** *blast*

lemma *prop-mctxt-cl-reflcl-on*:
prop-mctxt-cl $\mathcal{F}$ *P* ⟹ *funas-term s* ⊆ $\mathcal{F}$ ⟹ *P s s*
using *all-ctxt-closed-sig-reflE* **by** *blast*

lemma *reduction-relations-to-root-step*:
assumes ⋀ *s t*. (*s*, *t*) ∈ *srsteps-with-root-step* $\mathcal{F}$ $\mathcal{R}$ ⟹ *P s t*
and *cl*: *prop-mctxt-cl* $\mathcal{F}$ *P*
and *well*: *funas-term s* ⊆ $\mathcal{F}$ *funas-term t* ⊆ $\mathcal{F}$
and *steps*: (*s*, *t*) ∈ (*srstep* $\mathcal{F}$ $\mathcal{R}$)*
shows *P s t* **using** *steps well*
proof (*induct s arbitrary*: *t*)
case (*Var x*)
have (*Var x*, *t*) ∈ (*srstep* $\mathcal{F}$ $\mathcal{R}$)+ ⟹ (*Var x*, *t*) ∈ *srsteps-with-root-step* $\mathcal{F}$ $\mathcal{R}$
using *nsrsteps-with-root-step-step-on-args* **by** *blast*
from *assms*(*1*)[*OF this*] **show** *?case* **using** *Var cl*
by (*auto simp*: *rtrancl-eq-or-trancl dest*: *all-ctxt-closed-sig-reflE*)
next
case (*Fun f ss*) **note** *IH* = *this* **show** *?case*
proof (*cases Fun f ss* = *t*)
case *True* **show** *?thesis* **using** *IH*(*2*, *4*) **unfolding** *True*
by (*intro prop-mctxt-cl-reflcl-on*[*OF cl*]) *auto*
next
case *False*
then have *step*: (*Fun f ss*, *t*) ∈ (*srstep* $\mathcal{F}$ $\mathcal{R}$)+ **using** *IH*(*2*)
by (*auto simp*: *refl rtrancl-eq-or-trancl*)
show *?thesis*
proof (*cases* (*Fun f ss*, *t*) ∈ *srsteps-with-root-step* $\mathcal{F}$ $\mathcal{R}$)
case *False*
from *nsrsteps-with-root-step-step-on-args*[*OF step this*] **obtain** *ts*
where ∗[*simp*]: *t* = *Fun f ts* **and** *inv*: *length ss* = *length ts*
∀ *i* < *length ts*. (*ss* ! *i*, *ts* ! *i*) ∈ (*srstep* $\mathcal{F}$ $\mathcal{R}$)*
by *auto*
have *funas*: (*f*, *length ts*) ∈ $\mathcal{F}$ ∀*i*<*length ts*. *funas-term* (*ss* ! *i*) ⊆ $\mathcal{F}$ ∧
funas-term (*ts* ! *i*) ⊆ $\mathcal{F}$
using *IH*(*3*, *4*) *step inv*(*1*) **by** (*auto simp*: *UN-subset-iff*)
then have *t*: ∀ *i* < *length ts*. *P* (*ss* ! *i*) (*ts* ! *i*)
using *prop-mctxt-cl-reflcl-on*[*OF cl*] *IH*(*1*) *inv*
by (*auto simp*: *rtrancl-eq-or-trancl*)

```
    then show ?thesis unfolding ∗ using funas inv(1) all-ctxt-closedD[OF cl]
      by auto
   qed (auto simp add: assms(1))
  qed
qed



abbreviation comp-rrstep-rel 𝓕 ℛ 𝓢 ≡ srsteps-with-root-step 𝓕 ℛ O (srstep 𝓕
𝓢)* ∪
  (srstep 𝓕 ℛ)* O srsteps-with-root-step 𝓕 𝓢

abbreviation comp-rrstep-rel' 𝓕 ℛ 𝓢 ≡ srsteps-with-root-step 𝓕 ℛ O (srstep 𝓕
𝓢)+ ∪
  (srstep 𝓕 ℛ)+ O srsteps-with-root-step 𝓕 𝓢

lemma reduction-join-relations-to-root-step:
  assumes ⋀ s t. (s, t) ∈ comp-rrstep-rel 𝓕 ℛ 𝓢 ⟹ P s t
    and cl: prop-mctxt-cl 𝓕 P
    and well: funas-term s ⊆ 𝓕 funas-term t ⊆ 𝓕
    and steps: (s, t) ∈ (srstep 𝓕 ℛ)* O (srstep 𝓕 𝓢)*
  shows P s t using steps well
proof (induct s arbitrary: t)
  case (Var x)
  have f: (Var x, t) ∈ (srstep 𝓕 ℛ)+ ⟹ (Var x, t) ∈ comp-rrstep-rel 𝓕 ℛ 𝓢
   using nsrsteps-with-root-step-step-on-args[of Var x - 𝓕 ℛ] unfolding srsteps-with-root-step-def
   by (metis (no-types, lifting) Term.term.simps(4) UnI1 relcomp.relcompI rtrancl-eq-or-trancl)
  have s: (Var x, t) ∈ (srstep 𝓕 𝓢)+ ⟹ (Var x, t) ∈ comp-rrstep-rel 𝓕 ℛ 𝓢
   using nsrsteps-with-root-step-step-on-args[of Var x - 𝓕 𝓢] unfolding srsteps-with-root-step-def
   by (metis (no-types, lifting) Term.term.simps(4) UnI2 relcomp.simps rtrancl.simps)
  have t: (Var x, u) ∈ (srstep 𝓕 ℛ)+ ⟹ (u, t) ∈ (srstep 𝓕 𝓢)+ ⟹ (Var x, t)
∈ comp-rrstep-rel 𝓕 ℛ 𝓢 for u
   using nsrsteps-with-root-step-step-on-args[of Var x u 𝓕 ℛ] unfolding srsteps-with-root-step-def
    by auto (meson relcomp.simps trancl-into-rtrancl)
  show ?case using Var f[THEN assms(1)] s[THEN assms(1)] t[THEN assms(1)]
cl
    by (auto simp: rtrancl-eq-or-trancl prop-mctxt-cl-Var)
next
  case (Fun f ss) note IH = this show ?case
  proof (cases Fun f ss = t)
    case True show ?thesis using IH(2, 3, 4) cl
      by (auto simp: True prop-mctxt-cl-refl-on)
  next
    case False
   obtain u where u: (Fun f ss, u) ∈ (srstep 𝓕 ℛ)* (u, t) ∈ (srstep 𝓕 𝓢)* using
IH(2) by auto
    show ?thesis
    proof (cases (Fun f ss, u) ∈ srsteps-with-root-step 𝓕 ℛ)
      case True
```

```
      then have (Fun f ss, t) ∈ comp-rrstep-rel F R S using u
        by (auto simp: srsteps-with-root-step-def)
      from assms(1)[OF this] show ?thesis by simp
    next
      case False note nt-fst = this show ?thesis
      proof (cases (u, t) ∈ srsteps-with-root-step F S)
        case True
          then have (Fun f ss, t) ∈ comp-rrstep-rel F R S using u unfolding
srsteps-with-root-step-def
          by blast
        from assms(1)[OF this] show ?thesis by simp
      next
        case False note no-root = False nt-fst
        show ?thesis
        proof (cases Fun f ss = u ∨ u = t)
          case True
           from assms(1) have f: ⋀ s t. (s, t) ∈ srsteps-with-root-step F R ⟹ P
s t
             and s: ⋀ s t. (s, t) ∈ srsteps-with-root-step F S ⟹ P s t unfolding
srsteps-with-root-step-def
            by blast+
          have u = t ⟹ ?thesis using u cl IH(3, 4)
            by (intro reduction-relations-to-root-step[OF f]) auto
          moreover have Fun f ss = u ⟹ ?thesis using u cl IH(3, 4)
            by (intro reduction-relations-to-root-step[OF s]) auto
          ultimately show ?thesis using True by auto
        next
          case False
           then have steps: (Fun f ss, u) ∈ (srstep F R)+ (u, t) ∈ (srstep F S)+
using u
            by (auto simp: rtrancl-eq-or-trancl)
          obtain ts us
              where [simp]: u = Fun f us and inv-u: length ss = length us ∀ i <
length ts. (ss ! i, us ! i) ∈ (srstep F R)*
              and [simp]: t = Fun f ts and inv-t: length us = length ts ∀ i < length
ts. (us ! i, ts ! i) ∈ (srstep F S)*
            using nsrsteps-with-root-step-step-on-args[OF steps(1) no-root(2)]
            using nsrsteps-with-root-step-step-on-args[OF steps(2) no-root(1)]
            by auto
          from inv-u inv-t cl IH(3, 4) have t: ∀ i < length ts. P (ss ! i) (ts ! i)
           by (auto simp: UN-subset-iff intro!: IH(1)[OF nth-mem, of i ts ! i for i])
          moreover have (f, length ts) ∈ F using IH(4) by auto
          ultimately show ?thesis using IH(3, 4) inv-u inv-t all-ctxt-closedD[OF
cl]
            by (auto simp: UN-subset-iff)
        qed
      qed
    qed
  qed
```

qed

— Reducing search space for *commute* to conversions involving root steps

definition *commute-redp* $\mathcal{F}$ $\mathcal{R}$ $\mathcal{S}$ *s t* $\longleftrightarrow$ $(s,\ t) \in ((srstep\ \mathcal{F}\ \mathcal{S})^*\ O\ ((srstep\ \mathcal{F}\ \mathcal{R})^{-1})^*)$

declare *subsetI*[*rule del*]
lemma *commute-redp-mctxt-cl*:
prop-mctxt-cl $\mathcal{F}$ (*commute-redp* $\mathcal{F}$ $\mathcal{R}$ $\mathcal{S}$)
by (*auto simp*: *commute-redp-def rew-converse-inwards*
dest: *sigstep-trancl-funas intro*!: *all-ctxt-closed-relcomp*)
declare *subsetI*[*intro*!]

lemma *commute-rrstep-intro*:
assumes $\bigwedge$ *s t*. $(s,\ t) \in$ *comp-rrstep-rel'* $\mathcal{F}$ $(\mathcal{R}^{-1})$ $\mathcal{S} \Longrightarrow$ *commute-redp* $\mathcal{F}$ $\mathcal{R}$ $\mathcal{S}$ *s t*
shows *commute* (*srstep* $\mathcal{F}$ $\mathcal{R}$) (*srstep* $\mathcal{F}$ $\mathcal{S}$)
proof −
have [*simp*]: $x \in$ *srsteps-with-root-step* $\mathcal{F}$ $\mathcal{U} \Longrightarrow x \in (srstep\ \mathcal{F}\ \mathcal{U})^*\ O\ \mathcal{L}^*$ **for** *x* $\mathcal{U}$ $\mathcal{L}$
by (*cases x*) (*auto dest*!: *srsteps-with-root-step-sresteps-eqD*)
have [*simp*]: $x \in$ *srsteps-with-root-step* $\mathcal{F}$ $\mathcal{U} \Longrightarrow x \in \mathcal{L}^*\ O\ (srstep\ \mathcal{F}\ \mathcal{U})^*$ **for** *x* $\mathcal{U}$ $\mathcal{L}$
by (*cases x*) (*auto dest*!: *srsteps-with-root-step-sresteps-eqD*)
have *red*: $\bigwedge$ *s t*. $(s,\ t) \in$ *comp-rrstep-rel* $\mathcal{F}$ $(\mathcal{R}^{-1})$ $\mathcal{S} \Longrightarrow$ *commute-redp* $\mathcal{F}$ $\mathcal{R}$ $\mathcal{S}$ *s t* **using** *assms*
unfolding *commute-redp-def srstep-converse-dist*
by (*auto simp*: *rtrancl-eq-or-trancl*) *blast*+
have *comI*: $(\bigwedge$ *s t*. $(s,\ t) \in ((srstep\ \mathcal{F}\ (\mathcal{R}^{-1}))^*)\ O\ (srstep\ \mathcal{F}\ \mathcal{S})^* \Longrightarrow$ *commute-redp* $\mathcal{F}$ $\mathcal{R}$ $\mathcal{S}$ *s t*$) \Longrightarrow$
commute (*srstep* $\mathcal{F}$ $\mathcal{R}$) (*srstep* $\mathcal{F}$ $\mathcal{S}$)
by (*auto simp*: *commute-redp-def commute-def subsetD rew-converse-inwards*)
show *?thesis*
using *reduction-join-relations-to-root-step*[*OF red commute-redp-mctxt-cl*, *of* $\mathcal{R}^{-1}$ $\mathcal{S}$]
by (*intro comI*, *auto*) (*metis* (*no-types*, *lifting*) *commute-redp-def relcompI rew-converse-inwards sigstep-trancl-funas srstep-converse-dist*)
qed

lemma *commute-to-rrstep*:
assumes *commute* (*srstep* $\mathcal{F}$ $\mathcal{R}$) (*srstep* $\mathcal{F}$ $\mathcal{S}$)
shows $\bigwedge$ *s t*. $(s,\ t) \in$ *comp-rrstep-rel* $\mathcal{F}$ $(\mathcal{R}^{-1})$ $\mathcal{S} \Longrightarrow$ *commute-redp* $\mathcal{F}$ $\mathcal{R}$ $\mathcal{S}$ *s t*
using *assms*
unfolding *commute-def commute-redp-def srstep-converse-dist*
by (*auto simp*: *srstep-converse-dist dest*: *srsteps-with-root-step-sresteps-eqD*)

— Reducing search space for *CR* to conversions involving root steps

lemma *CR-Aux*:
assumes $\bigwedge s\ t.\ (s,\ t) \in (srstep\ \mathcal{F}\ (\mathcal{R}^{-1}))^*\ O\ srsteps\text{-}with\text{-}root\text{-}step\ \mathcal{F}\ \mathcal{R} \Longrightarrow commute\text{-}redp\ \mathcal{F}\ \mathcal{R}\ \mathcal{R}\ s\ t$
shows $\bigwedge s\ t.\ (s,\ t) \in comp\text{-}rrstep\text{-}rel\ \mathcal{F}\ (\mathcal{R}^{-1})\ \mathcal{R} \Longrightarrow commute\text{-}redp\ \mathcal{F}\ \mathcal{R}\ \mathcal{R}\ s\ t$
proof −
have *sym*: $commute\text{-}redp\ \mathcal{F}\ \mathcal{R}\ \mathcal{R}\ s\ t \Longrightarrow commute\text{-}redp\ \mathcal{F}\ \mathcal{R}\ \mathcal{R}\ t\ s$ **for** $s\ t$
by (*auto simp*: *commute-redp-def*) (*metis converseI relcomp.relcompI rtrancl-converse rtrancl-converseD*)
{fix $s\ t$ **assume** $(s,\ t) \in (srstep\ \mathcal{F}\ \mathcal{R})^*\ O\ srsteps\text{-}with\text{-}root\text{-}step\ \mathcal{F}\ (\mathcal{R}^{-1})$
then have $commute\text{-}redp\ \mathcal{F}\ \mathcal{R}\ \mathcal{R}\ s\ t$ **unfolding** *commute-redp-def*
by (*auto simp*: *srsteps-with-root-step-def rew-converse-inwards dest*!: *srrstep-to-srestep*)**}**
note $* = this$
{fix $s\ t$ **assume** *ass*: $(s,\ t) \in srsteps\text{-}with\text{-}root\text{-}step\ \mathcal{F}\ (\mathcal{R}^{-1})\ O\ (srstep\ \mathcal{F}\ \mathcal{R})^*$
have [*dest*!]: $(u,\ t) \in (srstep\ \mathcal{F}\ \mathcal{R})^* \Longrightarrow (t,\ u) \in (sig\text{-}step\ \mathcal{F}\ ((rstep\ \mathcal{R})^{-1}))^*$ **for** u
by (*metis rew-converse-outwards rtrancl-converseI srstep-converse-dist*)
from *ass* **have** $(t,\ s) \in (srstep\ \mathcal{F}\ (\mathcal{R}^{-1}))^*\ O\ srsteps\text{-}with\text{-}root\text{-}step\ \mathcal{F}\ \mathcal{R}$
unfolding *srsteps-with-root-step-def rstep-converse-dist*
by (*metis* (*mono-tags*, *lifting*) *O-assoc converse.simps converse-converse converse-inward*(*1*) *converse-relcomp rew-converse-outwards*(*1*, *2*) *sig-step-converse-rstep*)
from *assms*[*OF this*] **have** $commute\text{-}redp\ \mathcal{F}\ \mathcal{R}\ \mathcal{R}\ s\ t$ **using** *sym* **by** *blast***}**
then show $\bigwedge s\ t.\ (s,\ t) \in comp\text{-}rrstep\text{-}rel\ \mathcal{F}\ (\mathcal{R}^{-1})\ \mathcal{R} \Longrightarrow commute\text{-}redp\ \mathcal{F}\ \mathcal{R}\ \mathcal{R}\ s\ t$ **unfolding** *srsteps-with-root-step-def*
by (*metis UnE assms srsteps-with-root-step-def*)
qed

lemma *CR-rrstep-intro*:
assumes $\bigwedge s\ t.\ (s,\ t) \in (srstep\ \mathcal{F}\ (\mathcal{R}^{-1}))^+\ O\ srsteps\text{-}with\text{-}root\text{-}step\ \mathcal{F}\ \mathcal{R} \Longrightarrow commute\text{-}redp\ \mathcal{F}\ \mathcal{R}\ \mathcal{R}\ s\ t$
shows $CR\ (srstep\ \mathcal{F}\ \mathcal{R})$
proof −
{fix $s\ u$ **assume** $(s,\ u) \in (srstep\ \mathcal{F}\ (\mathcal{R}^{-1}))^*\ O\ srsteps\text{-}with\text{-}root\text{-}step\ \mathcal{F}\ \mathcal{R}$
then obtain t **where** *a*: $(s,\ t) \in (srstep\ \mathcal{F}\ (\mathcal{R}^{-1}))^*\ (t,\ u) \in srsteps\text{-}with\text{-}root\text{-}step\ \mathcal{F}\ \mathcal{R}$ **by** *blast*
have $commute\text{-}redp\ \mathcal{F}\ \mathcal{R}\ \mathcal{R}\ s\ u$
proof (*cases* $s = t$)
case [*simp*]: *True*
from *srsteps-with-root-step-srstepsD*[*OF a*(*2*)] **show** *?thesis*
by (*auto simp*: *commute-redp-def*)
next
case *False*
then have $(s,\ t) \in (srstep\ \mathcal{F}\ (\mathcal{R}^{-1}))^+$ **using** *a*(*1*) **unfolding** *rtrancl-eq-or-trancl*
by *simp*
then show *?thesis* **using** *assms a*(*2*) **by** *blast*
qed}
from *commute-rrstep-intro*[*OF CR-Aux*[*OF this*]]
show *?thesis* **unfolding** *CR-iff-self-commute*
by (*metis Un-iff reflcl-trancl relcomp-distrib relcomp-distrib2*)
qed

lemma *CR-to-rrstep*:
assumes *CR* (*srstep* $\mathcal{F}$ $\mathcal{R}$)
shows $\bigwedge$ *s t*. (*s*, *t*) ∈ *comp-rrstep-rel* $\mathcal{F}$ ($\mathcal{R}^{-1}$) $\mathcal{R}$ ⟹ *commute-redp* $\mathcal{F}$ $\mathcal{R}$ $\mathcal{R}$ *s t*
using *assms*
using *commute-to-rrstep*[*OF assms*[*unfolded CR-iff-self-commute*]]
by *simp*

— Reducing search space for *NFP* to conversions involving root steps

definition *NFP-redp* **where**
NFP-redp $\mathcal{F}$ $\mathcal{R}$ *s t* ⟷ *t* ∈ *NF* (*srstep* $\mathcal{F}$ $\mathcal{R}$) ⟶ (*s*, *t*) ∈ (*srstep* $\mathcal{F}$ $\mathcal{R}$)*

lemma *prop-mctxt-cl-NFP-redp*:
prop-mctxt-cl $\mathcal{F}$ (*NFP-redp* $\mathcal{F}$ $\mathcal{R}$)
proof −
{fix *f ts ss* **assume** *sig*: (*f*, *length ss*) ∈ $\mathcal{F}$ *length ts* = *length ss*
and *steps*: ∀ *i* < *length ss*. *ss* ! *i* ∈ *NF* (*srstep* $\mathcal{F}$ $\mathcal{R}$) ⟶ (*ts* ! *i*, *ss* ! *i*) ∈ (*srstep* $\mathcal{F}$ $\mathcal{R}$)*
and *funas*: ∀ *i* < *length ss*. *funas-term* (*ts* ! *i*) ⊆ $\mathcal{F}$ ∧ *funas-term* (*ss* ! *i*) ⊆ $\mathcal{F}$
and *NF*: *Fun f ss* ∈ *NF* (*srstep* $\mathcal{F}$ $\mathcal{R}$)
from *steps* **have** *steps*: *i* < *length ss* ⟹ (*ts* ! *i*, *ss* ! *i*) ∈ (*srstep* $\mathcal{F}$ $\mathcal{R}$)* **for** *i*
using *sig funas NF-srstep-args*[*OF NF*]
by (*auto simp*: *UN-subset-iff*) (*metis in-set-idx*)
then have (*Fun f ts*, *Fun f ss*) ∈ (*srstep* $\mathcal{F}$ $\mathcal{R}$)* **using** *sig*
by (*metis all-ctxt-closed-def all-ctxt-closed-sig-rsteps funas le-sup-iff*)**}**
then show *?thesis*
by (*auto simp*: *NFP-redp-def all-ctxt-closed-def*)
qed

lemma *NFP-rrstep-intro*:
assumes $\bigwedge$ *s t*. (*s*, *t*) ∈ *comp-rrstep-rel′* $\mathcal{F}$ ($\mathcal{R}^{-1}$) $\mathcal{R}$⟹ *NFP-redp* $\mathcal{F}$ $\mathcal{R}$ *s t*
shows *NFP* (*srstep* $\mathcal{F}$ $\mathcal{R}$)
proof −
from *assms* **have** *red*: $\bigwedge$ *t u*. (*t*, *u*) ∈ *comp-rrstep-rel* $\mathcal{F}$ ($\mathcal{R}^{-1}$) $\mathcal{R}$ ⟹ *NFP-redp* $\mathcal{F}$ $\mathcal{R}$ *t u*
apply (*auto simp*: *NFP-redp-def rtrancl-eq-or-trancl*)
apply (*metis NF-no-trancl-step converseD srstep-converse-dist srsteps-with-root-step-srstepsD trancl-converse*)
apply *blast*
apply (*meson NF-no-trancl-step srsteps-with-root-step-srstepsD*)
by *blast*
have $\bigwedge$ *s t*. (*s*, *t*) ∈ (*sig-step* $\mathcal{F}$ ((*rstep* $\mathcal{R}$)$^{-1}$))* *O* (*srstep* $\mathcal{F}$ $\mathcal{R}$)* ⟹ *NFP-redp* $\mathcal{F}$ $\mathcal{R}$ *s t*
using *reduction-join-relations-to-root-step*[*OF red prop-mctxt-cl-NFP-redp*, *of* $\mathcal{R}^{-1}$ $\mathcal{R}$]
by (*auto simp*: *NFP-redp-def*) (*metis* (*no-types*, *lifting*) *relcomp.relcompI rstep-converse-dist rtranclD srstepsD*)

then show *?thesis* **unfolding** *NFP-on-def NFP-redp-def*
by (*auto simp*: *normalizability-def*) (*metis meetI meet-def rstep-converse-dist srstep-converse-dist*)
qed

lemma *NFP-lift-to-conversion*:
assumes *NFP r* $(s, t) \in (r^{\leftrightarrow})^*$ **and** $t \in NF\ r$
shows $(s, t) \in r^*$ **using** *assms*(*2*, *3*)
proof (*induct rule*: *converse-rtrancl-induct*)
case (*step s u*)
then have $(u, t) \in r^!$ **by** *auto*
then show *?case* **using** *assms*(*1*) *step*(*1*) **unfolding** *NFP-on-def*
by *auto*
qed *simp*

lemma *NFP-to-rrstep*:
assumes *NFP* (*srstep* $\mathcal{F}$ $\mathcal{R}$)
shows $\bigwedge s\ t.\ (s, t) \in$ *srsteps-with-root-step* $\mathcal{F}$ $(\mathcal{R}^{\leftrightarrow}) \Longrightarrow$ *NFP-redp* $\mathcal{F}$ $\mathcal{R}$ *s t*
using *assms*
using *NFP-lift-to-conversion*[*OF assms*] **unfolding** *NFP-redp-def srsteps-with-root-step-def*
by *auto* (*metis* (*no-types*, *lifting*) *r-into-rtrancl rstep-converse-dist rtrancl-trans srrstep-to-srestep srstep-symcl-dist*)

— Reducing search space for *UNC* to conversions involving root steps

definition *UN-redp* $\mathcal{F}$ $\mathcal{R}$ *s t* $\longleftrightarrow$ $s \in NF$ (*srstep* $\mathcal{F}$ $\mathcal{R}$) $\wedge$ $t \in NF$ (*srstep* $\mathcal{F}$ $\mathcal{R}$) $\longrightarrow s = t$

lemma *prop-mctxt-cl-UN-redp*:
prop-mctxt-cl $\mathcal{F}$ (*UN-redp* $\mathcal{F}$ $\mathcal{R}$)
proof −
{fix *f ts ss* **assume** *sig*: $(f, length\ ss) \in \mathcal{F}$ *length ts* = *length ss*
and *steps*: $\forall\ i < length\ ss.\ ts\ !\ i \in NF$ (*srstep* $\mathcal{F}$ $\mathcal{R}$) $\wedge$ *ss* ! *i* $\in NF$ (*srstep* $\mathcal{F}$ $\mathcal{R}$) $\longrightarrow$ *ts* ! *i* = *ss* ! *i*
and *funas*: $\forall\ i < length\ ss.\ funas\text{-}term\ (ts\ !\ i) \subseteq \mathcal{F} \wedge funas\text{-}term\ (ss\ !\ i) \subseteq \mathcal{F}$
and *NF*: *Fun f ts* $\in NF$ (*srstep* $\mathcal{F}$ $\mathcal{R}$) *Fun f ss* $\in NF$ (*srstep* $\mathcal{F}$ $\mathcal{R}$)
from *steps* **have** *steps*: $i < length\ ss \Longrightarrow ts\ !\ i = ss\ !\ i$ **for** *i*
using *sig funas NF-srstep-args*[*OF NF*(*1*)] *NF-srstep-args*[*OF NF*(*2*)]
by (*auto simp*: *UN-subset-iff*) (*metis in-set-idx*)
then have *Fun f ts* = *Fun f ss* **using** *sig*(*2*)
by (*simp add*: *nth-equalityI*)**}**
then show *?thesis*
by (*auto simp*: *UN-redp-def all-ctxt-closed-def*)
qed

lemma *UNC-rrstep-intro*:
assumes$\bigwedge s\ t.\ (s, t) \in$ *srsteps-with-root-step* $\mathcal{F}$ $(\mathcal{R}^{\leftrightarrow}) \Longrightarrow$ *UN-redp* $\mathcal{F}$ $\mathcal{R}$ *s t*

shows *UNC* (*srstep* $\mathcal{F}$ $\mathcal{R}$)
proof −
have $\bigwedge$ *s t*. (*s*, *t*) ∈ (*srstep* $\mathcal{F}$ ($\mathcal{R}^{\leftrightarrow}$))* ⟹ *UN-redp* $\mathcal{F}$ $\mathcal{R}$ *s t*
using *reduction-relations-to-root-step*[*OF assms*(*1*) *prop-mctxt-cl-UN-redp*, *of* $\mathcal{R}^{\leftrightarrow}$]
by (*auto simp*: *UN-redp-def*) (*meson rtranclD srstepsD*)
then show *?thesis* **unfolding** *UNC-def UN-redp-def*
by (*auto simp*: *sig-step-conversion-dist*)
qed

lemma *UNC-to-rrstep*:
assumes *UNC* (*srstep* $\mathcal{F}$ $\mathcal{R}$)
shows $\bigwedge$ *s t*. (*s*, *t*) ∈ *srsteps-with-root-step* $\mathcal{F}$ ($\mathcal{R}^{\leftrightarrow}$) ⟹ *UN-redp* $\mathcal{F}$ $\mathcal{R}$ *s t*
using *assms* **unfolding** *UNC-def UN-redp-def srsteps-with-root-step-def*
by (*auto dest*!: *srrstep-to-srestep symcl-srstep-conversion symcl-srsteps-conversion*)
(*metis* (*no-types*, *opaque-lifting*) *conversion-def rtrancl-trans*)

— Reducing search space for *UNF* to conversions involving root steps

lemma *UNF-rrstep-intro*:
assumes $\bigwedge$ *t u*. (*t*, *u*) ∈ *comp-rrstep-rel′* $\mathcal{F}$ ($\mathcal{R}^{-1}$) $\mathcal{R}$ ⟹ *UN-redp* $\mathcal{F}$ $\mathcal{R}$ *t u*
shows *UNF* (*srstep* $\mathcal{F}$ $\mathcal{R}$)
proof −
from *assms* **have** *red*: $\bigwedge$ *t u*. (*t*, *u*) ∈ *comp-rrstep-rel* $\mathcal{F}$ ($\mathcal{R}^{-1}$) $\mathcal{R}$ ⟹ *UN-redp* $\mathcal{F}$ $\mathcal{R}$ *t u*
apply (*auto simp*: *UN-redp-def rtrancl-eq-or-trancl*)
apply (*metis NF-no-trancl-step converseD srstep-converse-dist srsteps-with-root-step-srstepsD trancl-converse*)
apply *blast*
apply (*meson NF-no-trancl-step srsteps-with-root-step-srstepsD*)
by *blast*
have $\bigwedge$ *s t*. (*s*, *t*) ∈ (*sig-step* $\mathcal{F}$ ((*rstep* $\mathcal{R}$)$^{-1}$))* *O* (*srstep* $\mathcal{F}$ $\mathcal{R}$)* ⟹ *UN-redp* $\mathcal{F}$ $\mathcal{R}$ *s t*
using *reduction-join-relations-to-root-step*[*OF red prop-mctxt-cl-UN-redp*, *of* $\mathcal{R}^{-1}$ $\mathcal{R}$]
by (*auto simp*: *UN-redp-def*) (*metis* (*no-types*, *lifting*) *relcomp.relcompI rstep-converse-dist rtranclD srstepsD*)
then show *?thesis* **unfolding** *UNF-on-def UN-redp-def*
by (*auto simp*: *normalizability-def*) (*metis meetI meet-def rstep-converse-dist srstep-converse-dist*)
qed

lemma *UNF-to-rrstep*:
assumes *UNF* (*srstep* $\mathcal{F}$ $\mathcal{R}$)
shows $\bigwedge$ *s t*. (*s*, *t*) ∈ *comp-rrstep-rel* $\mathcal{F}$ ($\mathcal{R}^{-1}$) $\mathcal{R}$ ⟹ *UN-redp* $\mathcal{F}$ $\mathcal{R}$ *s t*
using *assms* **unfolding** *UNF-on-def UN-redp-def normalizability-def srsteps-with-root-step-def*
by (*auto simp flip*: *srstep-converse-dist dest*!: *srrstep-to-srestep*)
(*metis* (*no-types*, *lifting*) *rstep-converse-dist rtrancl.rtrancl-into-rtrancl rtrancl-converseD*

rtrancl-idemp srstep-converse-dist)+

— Reducing search space for *CE* to conversions involving root steps

lemma *CE-rrstep-intro*:
assumes $\bigwedge s\ t.\ (s,\ t) \in \mathit{srsteps\text{-}with\text{-}root\text{-}step}\ \mathcal{F}\ (\mathcal{R}^{\leftrightarrow}) \Longrightarrow (s,\ t) \in (\mathit{srstep}\ \mathcal{F}\ \mathcal{S})^{\leftrightarrow *}$
and $\bigwedge s\ t.\ (s,\ t) \in \mathit{srsteps\text{-}with\text{-}root\text{-}step}\ \mathcal{F}\ (\mathcal{S}^{\leftrightarrow}) \Longrightarrow (s,\ t) \in (\mathit{srstep}\ \mathcal{F}\ \mathcal{R})^{\leftrightarrow *}$
shows *CE* (*srstep* $\mathcal{F}$ $\mathcal{R}$) (*srstep* $\mathcal{F}$ $\mathcal{S}$)
using *reduction-relations-to-root-step*[*OF assms*(*1*), **where** *?s1* = $\lambda\ s\ t.\ s$ **and** *?t1* = $\lambda\ s\ t.\ t$, *of* $\mathcal{F}$ $\mathcal{R}^{\leftrightarrow}$]
using *reduction-relations-to-root-step*[*OF assms*(*2*), **where** *?s1* = $\lambda\ s\ t.\ s$ **and** *?t1* = $\lambda\ s\ t.\ t$, *of* $\mathcal{F}$ $\mathcal{S}^{\leftrightarrow}$]
by (*auto simp*: *CE-on-def*)
(*metis converseI conversion-converse rtrancl-eq-or-trancl sig-step-conversion-dist sigstep-tranclfunas*(*1*, *2*))+

lemma *CE-to-rrstep*:
assumes *CE* (*srstep* $\mathcal{F}$ $\mathcal{R}$) (*srstep* $\mathcal{F}$ $\mathcal{S}$)
shows $\bigwedge s\ t.\ (s,\ t) \in \mathit{srsteps\text{-}with\text{-}root\text{-}step}\ \mathcal{F}\ (\mathcal{R}^{\leftrightarrow}) \Longrightarrow (s,\ t) \in (\mathit{srstep}\ \mathcal{F}\ \mathcal{S})^{\leftrightarrow *}$
$\bigwedge s\ t.\ (s,\ t) \in \mathit{srsteps\text{-}with\text{-}root\text{-}step}\ \mathcal{F}\ (\mathcal{S}^{\leftrightarrow}) \Longrightarrow (s,\ t) \in (\mathit{srstep}\ \mathcal{F}\ \mathcal{R})^{\leftrightarrow *}$
using *assms* **unfolding** *CE-on-def srsteps-with-root-step-def*
by (*auto simp flip*: *srstep-converse-dist dest*!: *srrstep-to-srestep symcl-srsteps-conversion symcl-srstep-conversion*)
(*metis converse-rtrancl-into-rtrancl conversion-rtrancl*)+

— Reducing search space for *NE* to conversions involving root steps

definition *NE-redp* **where**
NE-redp $\mathcal{F}$ $\mathcal{R}$ $\mathcal{S}$ *s t* $\longleftrightarrow$ *t* $\in$ *NF* (*srstep* $\mathcal{F}$ $\mathcal{R}$) $\longrightarrow$ *t* $\in$ *NF* (*srstep* $\mathcal{F}$ $\mathcal{R}$) $\longrightarrow$ (*s*, *t*) $\in$ (*srstep* $\mathcal{F}$ $\mathcal{S}$)*

lemma *prop-mctxt-cl-NE-redp*:
prop-mctxt-cl $\mathcal{F}$ (*NE-redp* $\mathcal{F}$ $\mathcal{R}$ $\mathcal{S}$)
proof −
{fix *f ts ss* **assume** *sig*: (*f*, *length ss*) $\in \mathcal{F}$ *length ts* = *length ss*
and *steps*: $\forall\ i < \mathit{length}\ ss.\ ss\ !\ i \in \mathit{NF}\ (\mathit{srstep}\ \mathcal{F}\ \mathcal{R}) \longrightarrow (ts\ !\ i,\ ss\ !\ i) \in (\mathit{srstep}\ \mathcal{F}\ \mathcal{S})^*$
and *funas*: $\forall\ i < \mathit{length}\ ss.\ \mathit{funas\text{-}term}\ (ts\ !\ i) \subseteq \mathcal{F} \wedge \mathit{funas\text{-}term}\ (ss\ !\ i) \subseteq \mathcal{F}$
and *NF*: *Fun f ss* $\in$ *NF* (*srstep* $\mathcal{F}$ $\mathcal{R}$)
from *steps* **have** *steps*: *i* < *length ss* $\Longrightarrow$ (*ts* ! *i*, *ss* ! *i*) $\in$ (*srstep* $\mathcal{F}$ $\mathcal{S}$)* **for** *i*
using *sig funas NF-srstep-args*[*OF NF*]
by (*auto simp*: *UN-subset-iff*) (*metis in-set-idx*)
then have (*Fun f ts*, *Fun f ss*) $\in$ (*srstep* $\mathcal{F}$ $\mathcal{S}$)* **using** *sig*
by (*metis all-ctxt-closed-def all-ctxt-closed-sig-rsteps funas le-sup-iff*)**}**
then show *?thesis*
by (*auto simp*: *all-ctxt-closed-def NE-redp-def*)

qed

lemma *NE-rrstep-intro*:
assumes $\bigwedge s\ t.\ (s,\ t) \in \mathit{srsteps\text{-}with\text{-}root\text{-}step}\ \mathcal{F}\ \mathcal{R} \Longrightarrow \mathit{NE\text{-}redp}\ \mathcal{F}\ \mathcal{R}\ \mathcal{S}\ s\ t$
and $\bigwedge s\ t.\ (s,\ t) \in \mathit{srsteps\text{-}with\text{-}root\text{-}step}\ \mathcal{F}\ \mathcal{S} \Longrightarrow \mathit{NE\text{-}redp}\ \mathcal{F}\ \mathcal{S}\ \mathcal{R}\ s\ t$
and $\mathit{NF}\ (\mathit{srstep}\ \mathcal{F}\ \mathcal{R}) = \mathit{NF}\ (\mathit{srstep}\ \mathcal{F}\ \mathcal{S})$
shows $\mathit{NE}\ (\mathit{srstep}\ \mathcal{F}\ \mathcal{R})\ (\mathit{srstep}\ \mathcal{F}\ \mathcal{S})$
using *assms*(*3*)
using *reduction-relations-to-root-step*[*OF assms*(*1*) *prop-mctxt-cl-NE-redp*, *of* $\mathcal{R}$]
using *reduction-relations-to-root-step*[*OF assms*(*2*) *prop-mctxt-cl-NE-redp*, *of* $\mathcal{S}$]
by (*auto simp*: *NE-on-def NE-redp-def normalizability-def*)
(*metis rtrancl.rtrancl-refl sigstep-trancl-funas*)+

lemma *NE-to-rrstep*:
assumes $\mathit{NE}\ (\mathit{srstep}\ \mathcal{F}\ \mathcal{R})\ (\mathit{srstep}\ \mathcal{F}\ \mathcal{S})$
shows $\bigwedge s\ t.\ (s,\ t) \in \mathit{srsteps\text{-}with\text{-}root\text{-}step}\ \mathcal{F}\ \mathcal{R} \Longrightarrow \mathit{NE\text{-}redp}\ \mathcal{F}\ \mathcal{R}\ \mathcal{S}\ s\ t$
$\bigwedge s\ t.\ (s,\ t) \in \mathit{srsteps\text{-}with\text{-}root\text{-}step}\ \mathcal{F}\ \mathcal{S} \Longrightarrow \mathit{NE\text{-}redp}\ \mathcal{F}\ \mathcal{S}\ \mathcal{R}\ s\ t$
using *assms* **unfolding** *NE-on-def NE-redp-def srsteps-with-root-step-def*
by (*auto simp*: *normalizability-def simp flip*: *srstep-converse-dist*
dest!: *srrstep-to-srestep*) (*meson converse-rtrancl-into-rtrancl rtrancl-trans*)+

lemma *NE-NF-eq*:
$\mathit{NE}\ \mathcal{R}\ \mathcal{S} \Longrightarrow \mathit{NF}\ \mathcal{R} = \mathit{NF}\ \mathcal{S}$
by (*auto simp*: *NE-on-def NF-def normalizability-def*)

— Reducing search space for *SCR* and *WCR* involving root steps

abbreviation $\mathit{SCRp}\ \mathcal{F}\ \mathcal{R}\ t\ u \equiv \exists v.\ (t,\ v) \in (\mathit{srstep}\ \mathcal{F}\ \mathcal{R})^{=} \wedge (u,\ v) \in (\mathit{srstep}\ \mathcal{F}\ \mathcal{R})^{*}$
lemma *SCR-rrstep-intro*:
assumes $\bigwedge s\ t\ u.\ (s,\ t) \in \mathit{sig\text{-}step}\ \mathcal{F}\ (\mathit{rrstep}\ \mathcal{R}) \Longrightarrow (s,\ u) \in \mathit{srstep}\ \mathcal{F}\ \mathcal{R} \Longrightarrow \mathit{SCRp}\ \mathcal{F}\ \mathcal{R}\ t\ u$
and $\bigwedge s\ t\ u.\ (s,\ t) \in \mathit{srstep}\ \mathcal{F}\ \mathcal{R} \Longrightarrow (s,\ u) \in \mathit{sig\text{-}step}\ \mathcal{F}\ (\mathit{rrstep}\ \mathcal{R}) \Longrightarrow \mathit{SCRp}\ \mathcal{F}\ \mathcal{R}\ t\ u$
shows $\mathit{SCR}\ (\mathit{srstep}\ \mathcal{F}\ \mathcal{R})$
proof −
{fix *s t u* **assume** *step*: $(s,\ t) \in \mathit{srstep}\ \mathcal{F}\ \mathcal{R}\ (s,\ u) \in \mathit{srstep}\ \mathcal{F}\ \mathcal{R}$
from *step*(*1*) **obtain** *p l r* σ **where** *st*: $p \in \mathit{poss}\ s\ (l,\ r) \in \mathcal{R}\ s\ |\text{-}\ p = l \cdot \sigma\ t = s[p \leftarrow r \cdot \sigma]$
using *rstep-to-pos-replace*[*of s t* $\mathcal{R}$] **unfolding** *sig-step-def* **by** *blast*
from *step*(*2*) **obtain** *q l2 r2* $\sigma 2$ **where** *su*: $q \in \mathit{poss}\ s\ (l2,\ r2) \in \mathcal{R}\ s\ |\text{-}\ q = l2 \cdot \sigma 2\ u = s[q \leftarrow r2 \cdot \sigma 2]$
using *rstep-to-pos-replace*[*of s u* $\mathcal{R}$] **unfolding** *sig-step-def* **by** *blast*
from *step st su* **have** *funas*: $\mathit{funas\text{-}term}\ s \subseteq \mathcal{F}\ \mathit{funas\text{-}term}\ t \subseteq \mathcal{F}\ \mathit{funas\text{-}term}\ u \subseteq \mathcal{F}$
by (*auto dest*: *srstepD*)
have *funas2* :$\mathit{funas\text{-}term}\ (r2 \cdot \sigma 2) \subseteq \mathcal{F}$ **using** *funas-term-replace-at-lower*[*OF*

su(*1*)]
using *funas*(*3*) **unfolding** *su*(*4*) **by** *blast*
consider (*a*) $p \leq_p q$ | (*b*) $q \leq_p p$ | (*c*) $p \perp q$
using *position-par-def* **by** *blast*
then have *SCRp* $\mathcal{F}$ $\mathcal{R}$ *t u*
proof *cases*
case *a*
from *a* **have** *up*: $p \in$ *poss u* **using** *st*(*1*) *su*(*1*) **unfolding** *st*(*4*) *su*(*4*)
by (*metis pos-replace-at-pres position-less-eq-def poss-append-poss*)
let *?C* = *ctxt-at-pos s p* **have** *fc*: *funas-ctxt ?C* $\subseteq \mathcal{F}$ **using** *funas*(*1*) *st*(*1*)
by (*metis ctxt-at-pos-subt-at-id funas-ctxt-apply le-sup-iff*)
from *funas* **have** *funas*: *funas-term* (*s* |- *p*) $\subseteq \mathcal{F}$ *funas-term* (*t* |- *p*) $\subseteq \mathcal{F}$ *funas-term* (*u* |- *p*) $\subseteq \mathcal{F}$
using *a st*(*1*) *pos-replace-at-pres*[*OF st*(*1*)] *up* **unfolding** *st*(*4*) *su*(*4*)
by (*intro funas-term-subterm-atI*, *blast*+)+
have (*s* |- *p*, *t* |- *p*) $\in$ *sig-step* $\mathcal{F}$ (*rrstep* $\mathcal{R}$) **unfolding** *st*(*4*) *su*(*4*) **using** *st*(*1* − *3*) *su*(*1* − *3*) *funas*
by (*metis poss-of-termE poss-of-term-replace-term-at rrstep.intros sig-stepI st*(*4*))
moreover have (*s* |- *p*, *u* |- *p*) $\in$ *srstep* $\mathcal{F}$ $\mathcal{R}$ **unfolding** *st*(*4*) *su*(*4*) **using** *st*(*1* − *3*) *su*(*1* − *3*) *funas*
by (*smt* (*verit*, *best*) *a ctxt-at-pos-subt-at-pos ctxt-of-pos-term-apply-replace-at-ident position-less-eq-def*
poss-append-poss replace-subterm-at-itself replace-term-at-subt-at-id rstepI sig-stepI su(*4*))
ultimately obtain *v* **where** (*t* |- *p*, *v*) $\in$ (*srstep* $\mathcal{F}$ $\mathcal{R}$)$^=$ (*u* |- *p*, *v*) $\in$ (*srstep* $\mathcal{F}$ $\mathcal{R}$)*
using *assms*(*1*) **by** *blast*
from *this*(*1*) *srsteps-eq-ctxt-closed*[*OF fc this*(*2*)]
show *?thesis* **using** *a st*(*1*) *su*(*1*) *srsteps-eq-ctxt-closed*[*OF fc*] **unfolding** *st*(*4*) *su*(*4*)
apply (*intro exI*[*of* - *?C*⟨*v*⟩])
apply (*auto simp*: *ctxt-of-pos-term-apply-replace-at-ident less-eq-subt-at-replace*)
apply (*metis ctxt-of-pos-term-apply-replace-at-ident fc srstep-ctxt-closed*)
done
next
case *b*
then have *up*: $q \in$ *poss t* **using** *st*(*1*) *su*(*1*) **unfolding** *st*(*4*) *su*(*4*)
by (*metis pos-replace-at-pres position-less-eq-def poss-append-poss*)
let *?C* = *ctxt-at-pos s q* **have** *fc*: *funas-ctxt ?C* $\subseteq \mathcal{F}$ **using** *funas*(*1*) *su*(*1*)
by (*metis Un-subset-iff ctxt-at-pos-subt-at-id funas-ctxt-apply*)
from *funas* **have** *funas*: *funas-term* (*s* |- *q*) $\subseteq \mathcal{F}$ *funas-term* (*t* |- *q*) $\subseteq \mathcal{F}$ *funas-term* (*u* |- *q*) $\subseteq \mathcal{F}$
using *su*(*1*) *pos-replace-at-pres*[*OF su*(*1*)] *up* **unfolding** *st*(*4*) *su*(*4*)
by (*intro funas-term-subterm-atI*, *blast*+)+
have (*s* |- *q*, *t* |- *q*) $\in$ *srstep* $\mathcal{F}$ $\mathcal{R}$ **unfolding** *st*(*4*) *su*(*4*) **using** *st*(*1* − *3*) *su*(*1* − *3*) *funas*
by (*smt* (*verit*, *del-insts*) *b ctxt-at-pos-subt-at-pos ctxt-of-pos-term-apply-replace-at-ident*
position-less-eq-def poss-append-poss replace-subterm-at-itself replace-term-at-subt-at-id

rstepI sig-stepI st(4))
moreover have (*s* |- *q*, *u* |- *q*) ∈ *sig-step* $\mathcal{F}$ (*rrstep* $\mathcal{R}$) **unfolding** *st(4) su(4)* **using** *st(1 − 3) su(1 − 3) funas*
by (*metis poss-of-termE poss-of-term-replace-term-at rrstep.intros sig-stepI su(4)*)
ultimately obtain *v* **where** (*t* |- *q*, *v*) ∈ (*srstep* $\mathcal{F}$ $\mathcal{R}$)$^=$ (*u* |- *q*, *v*) ∈ (*srstep* $\mathcal{F}$ $\mathcal{R}$)*
using *assms(2)* **by** *blast*
from *this(1) srsteps-eq-ctxt-closed*[*OF fc this(2)*]
show *?thesis* **using** *b st(1) su(1) srsteps-eq-ctxt-closed*[*OF fc*] **unfolding** *st(4) su(4)*
apply (*intro exI*[*of - ?C*⟨*v*⟩])
apply (*auto simp*: *ctxt-of-pos-term-apply-replace-at-ident less-eq-subt-at-replace*)
apply (*smt* (*verit*, *best*) *ctxt-of-pos-term-apply-replace-at-ident fc less-eq-subt-at-replace replace-term-at-above replace-term-at-subt-at-id srstep-ctxt-closed*)
done
next
case *c*
define *v* **where** *v* = *t*[*q* ← *r2* · *σ2*]
have *funasv*: *funas-term v* ⊆ $\mathcal{F}$ **using** *funas su(1)* **unfolding** *v-def su(4)*
using *funas-term-replace-at-upper funas2* **by** *blast*
from *c* **have** ∗: *v* = *u*[*p* ← *r* · *σ*] **unfolding** *v-def st(4) su(4)* **using** *st(1) su(1)*
using *parallel-replace-term-commute* **by** *blast*
from *c* **have** (*t*, *v*) ∈ *rstep* $\mathcal{R}$ **unfolding** *st(4) v-def*
using *su(1 − 3) par-pos-replace-pres*[*OF su(1)*]
by (*metis par-pos-replace-term-at pos-replace-to-rstep position-par-def*)
moreover from *c* **have** (*u*, *v*) ∈ *rstep* $\mathcal{R}$ **unfolding** *su(4)* ∗
using *st(1 − 3) par-pos-replace-pres*[*OF st(1)*]
by (*intro pos-replace-to-rstep*[*of - - l*]) (*auto simp*: *par-pos-replace-term-at*)
ultimately show *?thesis* **using** *funas(2−) funasv*
by *auto*
qed}
then show *?thesis* **unfolding** *SCR-on-def*
by *blast*
qed

lemma *SCE-to-rrstep*:
assumes *SCR* (*srstep* $\mathcal{F}$ $\mathcal{R}$)
shows ⋀ *s t u*. (*s*, *t*) ∈ *sig-step* $\mathcal{F}$ (*rrstep* $\mathcal{R}$) ⟹ (*s*, *u*) ∈ *srstep* $\mathcal{F}$ $\mathcal{R}$ ⟹ *SCRp* $\mathcal{F}$ $\mathcal{R}$ *t u*
⋀ *s t u*. (*s*, *t*) ∈ *srstep* $\mathcal{F}$ $\mathcal{R}$ ⟹ (*s*, *u*) ∈ *sig-step* $\mathcal{F}$ (*rrstep* $\mathcal{R}$) ⟹ *SCRp* $\mathcal{F}$ $\mathcal{R}$ *t u*
using *assms* **unfolding** *SCR-on-def srsteps-with-root-step-def*
by (*auto simp flip*: *srstep-converse-dist dest*!: *srrstep-to-srestep symcl-srsteps-conversion symcl-srstep-conversion*)

lemma *WCR-rrstep-intro*:
assumes ⋀ *s t u*. (*s*, *t*) ∈ *sig-step* $\mathcal{F}$ (*rrstep* $\mathcal{R}$) ⟹ (*s*, *u*) ∈ *srstep* $\mathcal{F}$ $\mathcal{R}$ ⟹ (*t*,

```
u) ∈ (srstep F R)↓
  shows WCR (srstep F R)
proof −
  {fix s t u assume step: (s, t) ∈ srstep F R (s, u) ∈ srstep F R
    from step(1) obtain p l r σ where st: p ∈ poss s (l, r) ∈ R s |- p = l · σ t
= s[p ← r · σ]
      using rstep-to-pos-replace[of s t R] unfolding sig-step-def by blast
    from step(2) obtain q l2 r2 σ2 where su: q ∈ poss s (l2, r2) ∈ R s |- q =
l2 · σ2 u = s[q ← r2 · σ2]
      using rstep-to-pos-replace[of s u R] unfolding sig-step-def by blast
    from step st su have funas: funas-term s ⊆ F funas-term t ⊆ F funas-term u
⊆ F
      by (auto dest: srstepD)
    have funas2 :funas-term (r2 · σ2) ⊆ F using funas-term-replace-at-lower[OF
su(1)]
      using funas(3) unfolding su(4) by blast
    consider (a) p ≤p q | (b) q ≤p p | (c) p ⊥ q
      using position-par-def by blast
    then have (t, u) ∈ (srstep F R)↓
    proof cases
      case a
      then have up: p ∈ poss u using st(1) su(1) unfolding st(4) su(4)
        by (metis pos-replace-at-pres position-less-eq-def poss-append-poss)
      let ?C = ctxt-at-pos s p have fc: funas-ctxt ?C ⊆ F using funas(1) st(1)
        by (metis Un-subset-iff ctxt-at-pos-subt-at-id funas-ctxt-apply)
       from funas have funas: funas-term (s |- p) ⊆ F funas-term (t |- p) ⊆ F
funas-term (u |- p) ⊆ F
        using a st(1) pos-replace-at-pres[OF st(1)] up unfolding st(4) su(4)
        by (intro funas-term-subterm-atI, blast+)+
       have (s |- p, t |- p) ∈ sig-step F (rrstep R) unfolding st(4) su(4) using
st(1 − 3) su(1 − 3) funas
        by (metis poss-of-termE poss-of-term-replace-term-at rrstep.intros sig-stepI
st(4))
      moreover have (s |- p, u |- p) ∈ srstep F R unfolding st(4) su(4) using
st(1 − 3) su(1 − 3) funas
      by (smt (verit, ccfv-threshold) a greater-eq-subt-at-replace less-eq-subt-at-replace
pos-diff-append-itself
         pos-replace-to-rstep position-less-eq-def poss-append-poss replace-term-at-subt-at-id
sig-stepI su(4))
      ultimately have (t |- p, u |- p) ∈ (srstep F R)↓
        using assms(1) by blast
      from sig-steps-join-ctxt-closed[OF fc this(1)]
      show ?thesis using a st(1) su(1) srstep-ctxt-closed[OF fc] unfolding st(4)
su(4)
       by (auto simp: ctxt-of-pos-term-apply-replace-at-ident less-eq-subt-at-replace)
    next
      case b
      then have up: q ∈ poss t using st(1) su(1) unfolding st(4) su(4)
        by (metis pos-les-eq-append-diff pos-replace-at-pres poss-append-poss)
```

```
    let ?C = ctxt-at-pos s q have fc: funas-ctxt ?C ⊆ F using funas(1) su(1)
      by (metis Un-subset-iff ctxt-at-pos-subt-at-id funas-ctxt-apply)
     from funas have funas: funas-term (s |- q) ⊆ F funas-term (t |- q) ⊆ F
funas-term (u |- q) ⊆ F
      using su(1) pos-replace-at-pres[OF su(1)] up unfolding st(4) su(4)
      by (intro funas-term-subterm-atI, blast+)+
     have (s |- q, t |- q) ∈ srstep F R unfolding st(4) su(4) using st(1 − 3)
su(1 − 3) funas
      by (smt (verit, ccfv-SIG) b greater-eq-subt-at-replace less-eq-subt-at-replace
pos-diff-append-itself
       pos-replace-to-rstep position-less-eq-def poss-append-poss replace-term-at-subt-at-id
sig-stepI st(4))
      moreover have (s |- q, u |- q) ∈ sig-step F (rrstep R) unfolding st(4)
su(4) using st(1 − 3) su(1 − 3) funas
      by (metis poss-of-termE poss-of-term-replace-term-at rrstep.intros sig-stepI
su(4))
    ultimately have (t |- q, u |- q) ∈ (srstep F R)↓
      using assms(1) by blast
    from sig-steps-join-ctxt-closed[OF fc this(1)]
    show ?thesis using b st(1) su(1) srstep-ctxt-closed[OF fc] unfolding st(4)
su(4)
     by (auto simp: ctxt-of-pos-term-apply-replace-at-ident less-eq-subt-at-replace)
   next
    case c
    define v where v = t[q ← r2 · σ2]
    have funasv: funas-term v ⊆ F using funas su(1) unfolding v-def su(4)
      using funas-term-replace-at-upper funas2 by blast
    from c have ∗: v = u[p ← r · σ] unfolding v-def st(4) su(4) using st(1)
su(1)
      using parallel-replace-term-commute by blast
    from c have (t, v) ∈ rstep R unfolding st(4) v-def
      using su(1 − 3) par-pos-replace-pres[OF su(1)]
      by (metis par-pos-replace-term-at pos-replace-to-rstep position-par-def)
    moreover from c have (u, v) ∈ rstep R unfolding su(4) ∗
      using st(1 − 3) par-pos-replace-pres[OF st(1)]
      by (metis par-pos-replace-term-at pos-replace-to-rstep)
    ultimately show ?thesis using funas(2−) funasv
      by auto
   qed}
  then show ?thesis unfolding WCR-on-def
   by blast
qed

end
theory Rewriting-LLRG-LV-Mondaic
  imports Rewriting
   Replace-Constant
begin
```

4.3 Specific results about rewriting under a linear variable-separated system

lemma *card-varposs-ground*:
card (*varposs s*) = *0* ⟷ *ground s*
by (*simp add*: *finite-varposs varposs-empty-gound*)

lemma *poss-of-term-subst-apply-varposs*:
assumes *p* ∈ *poss-of-term* (*constT c*) (*s* · *σ*) (*c*, *0*) ∉ *funas-term s*
shows ∃ *q*. *q* ∈ *varposs s* ∧ *q* ≤$_p$ *p* **using** *assms*
proof (*induct p arbitrary*: *s*)
case *Nil*
then show *?case* **by** (*cases s*) (*auto simp*: *poss-of-term-def*)
next
case (*Cons i p*)
show *?case* **using** *Cons*(*1*)[*of args s* ! *i*] *Cons*(*2*−)
apply (*cases s*)
apply (*auto simp*: *poss-of-term-def*)
apply (*metis position-less-eq-Cons*)+
done
qed

lemma *poss-of-term-hole-poss*:
assumes *p* ∈ *poss-of-term t C*⟨*s*⟩ **and** *hole-pos C* ≤$_p$ *p*
shows *p* −$_p$ *hole-pos C* ∈ *poss-of-term t s* **using** *assms*
proof (*induct C arbitrary*: *p*)
case (*More f ss C ts*)
from *More*(*3*) **obtain** *ps* **where** [*simp*]: *p* = *length ss* # *ps* **and** *h*: *hole-pos C*
≤$_p$ *ps*
by (*metis append-Cons hole-pos.simps*(*2*) *less-eq-poss-append-itself pos-les-eq-append-diff*)
show *?case* **using** *More*(*1*)[*OF - h*] *More*(*2*)
by (*auto simp*: *poss-of-term-def*)
qed *auto*

lemma *remove-const-subst-from-match*:
assumes *s* · *const-subst c* = *C*⟨*l* · *σ*⟩ (*c*, *0*) ∉ *funas-term l linear-term l*
shows ∃ *D τ*. *s* = *D*⟨*l* · *τ*⟩ **using** *assms*
proof (*induct card* (*varposs s*) *arbitrary*: *s*)
case (*Suc x*)
from *Suc*(*2*) **obtain** *p ps* **where** *varposs*: *varposs s* = *insert p ps p* ∉ *ps*
by (*metis card-Suc-eq*)
let *?s* = *s*[*p* ← *Fun c* []] **have** *vp*: *p* ∈ *varposs s* **using** *varposs* **by** *auto*
then have ∗: *?s* · *const-subst c* = *s* · *const-subst c*
by (*induct s arbitrary*: *p*) (*auto simp*: *nth-list-update map-update intro*!: *nth-equalityI*)
have *varposs ?s* = *ps* **using** *varposs varposs-ground-replace-at*[*of p s constT c*]
by *auto*
from *Suc*(*1*)[*of ?s*] *Suc*(*2*−) *varposs* **obtain** *D τ* **where** *split*: *s*[*p* ← *constT c*]
= *D*⟨*l* · *τ*⟩
by (*metis* ∗ ‹*varposs s*[*p* ← *constT c*] = *ps*› *card-insert-if diff-Suc-1 finite-varposs*)
have *wit*: *s* = *D*⟨*l* · *τ*⟩[*p* ← *s* |- *p*] **unfolding** *arg-cong*[*OF split*, *of* λ *t*. *t*[*p* ←

s |- *p*], *symmetric*]
using *vp* **by** *simp*
from *vp split* **have** *cases*: $p \perp$ *hole-pos D* $\vee$ *hole-pos D* $\leq_p$ *p*
by *auto* (*metis poss-of-term-const-ctxt-apply poss-of-term-replace-term-at var-poss-imp-poss*)
show *?case*
proof (*cases p* $\perp$ *hole-pos D*)
case *True* **then show** *?thesis* **using** *wit*
by (*auto simp*: *par-hole-pos-replace-term-context-at*)
next
case *False*
then have *hole*: *hole-pos D* $\leq_p$ *p* **using** *cases* **by** *auto*
from *vp split* **have** *p* $\in$ *poss-of-term* (*constT c*) *s*[*p* $\leftarrow$ *constT c*]
using *poss-of-term-replace-term-at varposs-imp-poss* **by** *blast*
from *poss-of-term-hole-poss*[*OF this*[*unfolded split*] *hole*]
have *p* $-_p$ *hole-pos D* $\in$ *poss-of-term* (*constT c*) ($l \cdot \tau$)
by *simp*
from *poss-of-term-subst-apply-varposs*[*OF this Suc*(*4*)] **obtain** *q* **where**
q: *q* $\in$ *varposs l q* $\leq_p$ (*p* $-_p$ (*hole-pos D*)) **by** *blast*
show *?thesis* **using** *wit Suc*(*5*) *hole*
using *linear-term-varposs-subst-replace-term*[*OF Suc*(*5*) *q*, *of* τ *s* |- *p*]
by *auto*
qed
qed (*auto simp*: *card-varposs-ground ground-subst-apply*)

definition *llrg* $\mathcal{R} \longleftrightarrow (\forall\ (l, r) \in \mathcal{R}.$ *linear-term l* $\wedge$ *ground r*)

definition *lv* $\mathcal{R} \longleftrightarrow (\forall\ (l, r) \in \mathcal{R}.$ *linear-term l* $\wedge$ *linear-term r* $\wedge$ *vars-term l* $\cap$ *vars-term r* $= \{\}$)

definition *monadic* $\mathcal{F} \longleftrightarrow (\forall\ (f, n) \in \mathcal{F}.\ n \leq$ *Suc 0*)

— NF of ground terms

lemma *ground-NF-srstep-gsrstep*:
ground s $\Longrightarrow$ *s* $\in$ *NF* (*srstep* $\mathcal{F}$ $\mathcal{R}$) $\Longrightarrow$ *s* $\in$ *NF* (*gsrstep* $\mathcal{F}$ $\mathcal{R}$)
by *blast*

lemma *NF-to-fresh-const-subst-NF*:
assumes *lin*: *linear-sys* $\mathcal{R}$ **and** *fresh-const*: (*c*, *0*) $\notin$ *funas-rel* $\mathcal{R}$ *funas-rel* $\mathcal{R} \subseteq \mathcal{F}$
and *nf-f*: *funas-term s* $\subseteq \mathcal{F}$ *s* $\in$ *NF* (*srstep* $\mathcal{F}$ $\mathcal{R}$)
shows $s \cdot$ *const-subst c* $\in$ *NF* (*gsrstep* $\mathcal{H}$ $\mathcal{R}$)
proof (*rule ccontr*)

assume $s \cdot$ *const-subst* $c \notin$ *NF* (*Restr* (*srstep* $\mathcal{H}$ $\mathcal{R}$) (*Collect ground*))
then obtain C l r σ **where** *step*: $(l, r) \in \mathcal{R}$ $s \cdot$ *const-subst* $c = C\langle l \cdot \sigma \rangle$ **by** *fastforce*
from *step*(*1*) **have** *l*: $(c, 0) \notin$ *funas-term* l *linear-term* l **using** *lin fresh-const*
by (*auto simp*: *funas-rel-def*)
obtain D τ **where** $s = D\langle l \cdot \tau \rangle$ **using** *remove-const-subst-from-match*[*OF step*(*2*) *l*] **by** *blast*
then show *False* **using** *step*(*1*) *nf-f*
by (*meson NF-no-trancl-step fresh-const*(*2*) *r-into-trancl'* *rstepI rstep-trancl-sig-step-r*)
qed

lemma *fresh-const-subst-NF-pres*:
assumes *fresh-const*: $(c, 0) \notin$ *funas-rel* $\mathcal{R}$ *funas-rel* $\mathcal{R} \subseteq \mathcal{F}$
and *nf-f*: *funas-term* $s \subseteq \mathcal{F}$ $\mathcal{F} \subseteq \mathcal{H}$ $(c, 0) \in \mathcal{H}$ $s \cdot$ *const-subst* $c \in$ *NF* (*gsrstep* $\mathcal{H}$ $\mathcal{R}$)
shows $s \in$ *NF* (*srstep* $\mathcal{F}$ $\mathcal{R}$)
proof (*rule ccontr*)
assume $s \notin$ *NF* (*srstep* $\mathcal{F}$ $\mathcal{R}$)
then obtain C l r σ **where** *step*: $(l, r) \in \mathcal{R}$ $s = C\langle l \cdot \sigma \rangle$ **by** *fastforce*
let $?\tau = \lambda\ x.$ *if* $x \in$ *vars-term* l *then* $(\sigma\ x) \cdot$ *const-subst* c *else Fun* c []
define D **where** $D = (C \cdot_c$ *const-subst* $c)$
have *s*: $s \cdot$ *const-subst* $c = D\langle l \cdot ?\tau \rangle$ **unfolding** *D-def step*(*2*)
by (*auto simp*: *subst-compose simp flip*: *subst-subst-compose intro*!: *term-subst-eq*)
have *funas*: *funas-ctxt* $D \subseteq \mathcal{H}$ *funas-term* $(l \cdot ?\tau) \subseteq \mathcal{H}$ *funas-term* $(r \cdot ?\tau) \subseteq \mathcal{H}$
using *step nf-f*(*1* − *3*) *fresh-const*(*2*) **unfolding** *D-def*
by (*auto simp*: *funas-ctxt-subst-apply-ctxt funas-term-subst funas-rel-def split*: *if-splits*)
moreover have *ground-ctxt D ground* $(l \cdot ?\tau)$ *ground* $(r \cdot ?\tau)$ **using** *arg-cong*[*OF s*, *of ground*] **unfolding** *D-def*
by (*auto intro*!: *ground-substI*)
ultimately have $(D\langle l \cdot ?\tau \rangle, D\langle r \cdot ?\tau \rangle) \in$ *gsrstep* $\mathcal{H}$ $\mathcal{R}$ **using** *step*(*1*)
by (*simp add*: *rstepI sig-stepI*)
then show *False* **using** *nf-f*(*4*) **unfolding** *s*[*symmetric*]
by *blast*
qed

lemma *linear-sys-gNF-eq-NF-eq*:
assumes *lin*: *linear-sys* $\mathcal{R}$ *linear-sys* $\mathcal{S}$
and *well*: *funas-rel* $\mathcal{R} \subseteq \mathcal{F}$ *funas-rel* $\mathcal{S} \subseteq \mathcal{F}$
and *fresh*: $(c, 0) \notin$ *funas-rel* $\mathcal{R}$ $(c, 0) \notin$ *funas-rel* $\mathcal{S}$
and *lift*: $\mathcal{F} \subseteq \mathcal{H}$ $(c, 0) \in \mathcal{H}$
and *nf*: *NF* (*gsrstep* $\mathcal{H}$ $\mathcal{R}$) = *NF* (*gsrstep* $\mathcal{H}$ $\mathcal{S}$)
shows *NF* (*srstep* $\mathcal{F}$ $\mathcal{R}$) = *NF* (*srstep* $\mathcal{F}$ $\mathcal{S}$)
proof −
have [*simp*]: ¬ *funas-term* $s \subseteq \mathcal{F} \Longrightarrow s \in$ *NF* (*srstep* $\mathcal{F}$ $\mathcal{U}$) **for** s $\mathcal{U}$ **by** (*meson NF-I sig-stepE*(*1*))
have *d1*: $s \in$ *NF* (*gsrstep* $\mathcal{H}$ $\mathcal{R}$) $\Longrightarrow s \in$ *NF* (*gsrstep* $\mathcal{H}$ $\mathcal{S}$) **for** s **using** *nf* **by** *auto*

have *d2*: $s \in NF\ (gsrstep\ \mathcal{H}\ \mathcal{S}) \Longrightarrow s \in NF\ (gsrstep\ \mathcal{H}\ \mathcal{R})$ **for** s **using** *nf* **by** *auto*
{fix s **assume** n: $s \in NF\ (srstep\ \mathcal{F}\ \mathcal{R})$ **then have** $s \in NF\ (srstep\ \mathcal{F}\ \mathcal{S})$
using *NF-to-fresh-const-subst-NF*[*OF lin*(*1*) *fresh*(*1*) *well*(*1*) - *n*, *THEN d1*]
using *fresh-const-subst-NF-pres*[*OF fresh*(*2*) *well*(*2*) - *lift*, *of s*]
by (*cases funas-term* $s \subseteq \mathcal{F}$) *simp-all***}**
moreover
{fix s **assume** n: $s \in NF\ (srstep\ \mathcal{F}\ \mathcal{S})$ **then have** $s \in NF\ (srstep\ \mathcal{F}\ \mathcal{R})$
using *NF-to-fresh-const-subst-NF*[*OF lin*(*2*) *fresh*(*2*) *well*(*2*) - *n*, *THEN d2*]
using *fresh-const-subst-NF-pres*[*OF fresh*(*1*) *well*(*1*) - *lift*, *of s*]
by (*cases funas-term* $s \subseteq \mathcal{F}$) *simp-all***}**
ultimately show *?thesis* **by** *blast*
qed

— Steps of ground
lemma *gsrsteps-to-srsteps*:
$(s,\ t) \in (gsrstep\ \mathcal{F}\ \mathcal{R})^+ \Longrightarrow (s,\ t) \in (srstep\ \mathcal{F}\ \mathcal{R})^+$
by (*meson inf-le1 trancl-mono*)

lemma *gsrsteps-eq-to-srsteps-eq*:
$(s,\ t) \in (gsrstep\ \mathcal{F}\ \mathcal{R})^* \Longrightarrow (s,\ t) \in (srstep\ \mathcal{F}\ \mathcal{R})^*$
by (*metis gsrsteps-to-srsteps rtrancl-eq-or-trancl*)

lemma *gsrsteps-to-rsteps*:
$(s,\ t) \in (gsrstep\ \mathcal{F}\ \mathcal{R})^+ \Longrightarrow (s,\ t) \in (rstep\ \mathcal{R})^+$
using *gsrsteps-to-srsteps srstepsD* **by** *blast*

lemma *gsrsteps-eq-to-rsteps-eq*:
$(s,\ t) \in (gsrstep\ \mathcal{F}\ \mathcal{R})^* \Longrightarrow (s,\ t) \in (rstep\ \mathcal{R})^*$
by (*metis gsrsteps-eq-to-srsteps-eq rtrancl-eq-or-trancl srstepsD*)

lemma *gsrsteps-eq-relcomp-srsteps-relcompD*:
$(s,\ t) \in (gsrstep\ \mathcal{F}\ \mathcal{R})^*\ O\ (gsrstep\ \mathcal{F}\ \mathcal{S})^* \Longrightarrow (s,\ t) \in (srstep\ \mathcal{F}\ \mathcal{R})^*\ O\ (srstep\ \mathcal{F}\ \mathcal{S})^*$
using *gsrsteps-eq-to-srsteps-eq* **by** *blast*

lemma *gsrsteps-eq-relcomp-to-rsteps-relcomp*:
$(s,\ t) \in (gsrstep\ \mathcal{F}\ \mathcal{R})^*\ O\ (gsrstep\ \mathcal{F}\ \mathcal{S})^* \Longrightarrow (s,\ t) \in (rstep\ \mathcal{R})^*\ O\ (rstep\ \mathcal{S})^*$
using *gsrsteps-eq-relcomp-srsteps-relcompD*
using *gsrsteps-eq-to-rsteps-eq* **by** *blast*

lemma *ground-srsteps-gsrsteps*:
assumes *ground s ground t*
and $(s,\ t) \in (srstep\ \mathcal{F}\ \mathcal{R})^+$
shows $(s,\ t) \in (gsrstep\ \mathcal{F}\ \mathcal{R})^+$

proof −
let *?σ* = λ -. *s*
from *assms*(*3*) **have** *f*: *funas-term s* ⊆ $\mathcal{F}$ **using** *srstepsD* **by** *blast*
have (*s* · *?σ*, *t* · *?σ*) ∈ (*gsrstep* $\mathcal{F}$ $\mathcal{R}$)$^+$ **using** *assms*(*3*, *1*) *f*
proof (*induct*)
case (*base t*)
then have (*s* · *?σ*, *t* · *?σ*) ∈ *gsrstep* $\mathcal{F}$ $\mathcal{R}$
by (*auto intro*: *srstep-subst-closed*)
then show *?case* **by** *auto*
next
case (*step t u*)
from *step*(*2*, *4*, *5*) **have** (*t* · *?σ*, *u* · *?σ*) ∈ *gsrstep* $\mathcal{F}$ $\mathcal{R}$
by (*auto intro*: *srstep-subst-closed*)
then show *?case* **using** *step*(*3* − *5*)
by (*meson Transitive-Closure.trancl-into-trancl*)
qed
then show *?thesis* **using** *assms*(*1*, *2*)
by (*simp add*: *ground-subst-apply*)
qed

lemma *ground-srsteps-eq-gsrsteps-eq*:
assumes *ground s ground t*
and (*s*, *t*) ∈ (*srstep* $\mathcal{F}$ $\mathcal{R}$)*
shows (*s*, *t*) ∈ (*gsrstep* $\mathcal{F}$ $\mathcal{R}$)*
using *ground-srsteps-gsrsteps*
by (*metis assms rtrancl-eq-or-trancl*)

lemma *srsteps-eq-relcomp-gsrsteps-relcomp*:
assumes (*s*, *t*) ∈ (*srstep* $\mathcal{F}$ $\mathcal{R}$)* *O* (*srstep* $\mathcal{F}$ $\mathcal{S}$)*
and *ground s ground t*
shows (*s*, *t*) ∈ (*gsrstep* $\mathcal{F}$ $\mathcal{R}$)* *O* (*gsrstep* $\mathcal{F}$ $\mathcal{S}$)*
proof −
from *assms*(*1*) **obtain** *u* **where** *steps*: (*s*, *u*) ∈ (*srstep* $\mathcal{F}$ $\mathcal{R}$)* (*u*, *t*) ∈ (*srstep* $\mathcal{F}$ $\mathcal{S}$)*
by *blast*
let *?σ* = λ *x*. *s*
have (*s* · *?σ*, *u* · *?σ*) ∈ (*srstep* $\mathcal{F}$ $\mathcal{R}$)* (*u* · *?σ*, *t* · *?σ*) ∈ (*srstep* $\mathcal{F}$ $\mathcal{S}$)* **using** *steps*
using *srsteps-eq-subst-closed*[*OF steps*(*1*), *of ?σ*]
using *srsteps-eq-subst-closed*[*OF steps*(*2*), *of ?σ*]
by (*metis rtrancl-eq-or-trancl srstepsD*)+
then have (*s* · *?σ*, *u* · *?σ*) ∈ (*gsrstep* $\mathcal{F}$ $\mathcal{R}$)* (*u* · *?σ*, *t* · *?σ*) ∈ (*gsrstep* $\mathcal{F}$ $\mathcal{S}$)*
using *assms*(*2*)
by (*auto intro*: *ground-srsteps-eq-gsrsteps-eq*)
then show *?thesis* **using** *assms*(*2*, *3*)
by (*auto simp*: *ground-subst-apply*)
qed

— Steps of llrg systems

lemma *llrg-ground-rhs*:
llrg $\mathcal{R} \Longrightarrow (l,\ r) \in \mathcal{R} \Longrightarrow$ *ground r*
unfolding *llrg-def* **by** *auto*

lemma *llrg-rrsteps-groundness*:
assumes *llrg* $\mathcal{R}$ **and** $(s,\ t) \in$ (*srrstep* $\mathcal{F}$ $\mathcal{R}$)
shows *ground t* **using** *assms*(*2*) *ground-vars-term-empty*
by (*fastforce simp*: *llrg-def sig-step-def dest*!: *llrg-ground-rhs*[*OF assms*(*1*)] *split*: *prod.splits*)

lemma *llrg-rsteps-pres-groundness*:
assumes *llrg* $\mathcal{R}$ *ground s*
and $(s,\ t) \in$ (*srstep* $\mathcal{F}$ $\mathcal{R}$)*
shows *ground t* **using** *assms*(*3*, *2*)
proof (*induct rule*: *rtrancl.induct*)
case (*rtrancl-into-rtrancl s t u*)
then have *ground t* **by** *auto*
then show *?case* **using** *rtrancl-into-rtrancl*(*3*)
by (*auto simp*: *sig-step-def vars-term-ctxt-apply ground-vars-term-empty ground-subst-apply dest*!: *llrg-ground-rhs*[*OF assms*(*1*)] *rstep-imp-C-s-r split*: *prod.splits*)
qed *simp*

lemma *llrg-srsteps-with-root-step-ground*:
assumes *llrg* $\mathcal{R}$ **and** $(s,\ t) \in$ *srsteps-with-root-step* $\mathcal{F}$ $\mathcal{R}$
shows *ground t* **using** *assms llrg-rrsteps-groundness llrg-rsteps-pres-groundness*
unfolding *srsteps-with-root-step-def*
by *blast*

lemma *llrg-srsteps-with-root-step-inv-ground*:
assumes *llrg* $\mathcal{R}$ **and** $(s,\ t) \in$ *srsteps-with-root-step* $\mathcal{F}$ $(\mathcal{R}^{-1})$
shows *ground s* **using** *assms llrg-rrsteps-groundness llrg-rsteps-pres-groundness*
unfolding *srsteps-with-root-step-def*
by (*metis* (*no-types*, *lifting*) *converseD relcomp.cases rtrancl-converseD srrstep-converse-dist srstep-converse-dist*)

lemma *llrg-funas-term-step-pres*:
assumes *llrg* $\mathcal{R}$ **and** $(s,\ t) \in$ (*rstep* $\mathcal{R}$)
shows *funas-term t* $\subseteq$ *funas-rel* $\mathcal{R}$ $\cup$ *funas-term s*
proof −
have [*simp*]: $(l,\ r) \in \mathcal{R} \Longrightarrow r \cdot \sigma = r$ **for** *l r* σ **using** *assms*(*1*) **unfolding** *llrg-def*
by(*auto split*: *prod.splits intro*: *ground-subst-apply*)
show *?thesis* **using** *assms*
by (*auto simp*: *llrg-def funas-rel-def dest*!: *rstep-imp-C-s-r*)
qed

lemma *llrg-funas-term-steps-pres*:
assumes *llrg* $\mathcal{R}$ **and** $(s,\ t) \in$ (*rstep* $\mathcal{R}$)*

shows *funas-term t* $\subseteq$ *funas-rel* $\mathcal{R}$ $\cup$ *funas-term s*
using *assms*(*2*) *llrg-funas-term-step-pres*[*OF assms*(*1*)]
by (*induct*) *auto*

— Steps of monadic llrg systems

lemma *monadic-ground-ctxt-apply*:
monadic $\mathcal{F}$ $\Longrightarrow$ *funas-ctxt C* $\subseteq$ $\mathcal{F}$ $\Longrightarrow$ *ground r* $\Longrightarrow$ *ground* $C\langle r\rangle$
by (*induct C*) (*auto simp*: *monadic-def*)

lemma *llrg-monadic-rstep-pres-groundness*:
assumes *llrg* $\mathcal{R}$ *monadic* $\mathcal{F}$
and (*s*, *t*) $\in$ *srstep* $\mathcal{F}$ $\mathcal{R}$
shows *ground t* **using** *assms*(*3*)
proof −
from *assms*(*3*) **obtain** *C l r* σ **where** *r*: (*l*, *r*) $\in$ $\mathcal{R}$ **and** *t*:*t* = $C\langle r \cdot \sigma\rangle$
using *rstep-imp-C-s-r* **unfolding** *sig-step-def* **by** *blast*
from *assms*(*1*, *3*) **have** *funas*: *funas-term t* $\subseteq$ $\mathcal{F}$ *ground r*
by (*auto simp*: *llrg-ground-rhs*[*OF assms*(*1*) *r*(*1*)] *dest*: *srstepD*)
then have $*$: $r \cdot \sigma = r$ **by** (*simp add*: *ground-subst-apply*)
show *?thesis* **using** *funas assms*(*2*) **unfolding** *t* $*$
by (*intro monadic-ground-ctxt-apply*) *auto*
qed

lemma *llrg-monadic-rsteps-groundness*:
assumes *llrg* $\mathcal{R}$ *monadic* $\mathcal{F}$
and (*s*, *t*) $\in$ (*srstep* $\mathcal{F}$ $\mathcal{R}$)$^+$
shows *ground t* **using** *assms*(*3*)
using *llrg-monadic-rstep-pres-groundness*[*OF assms*(*1* ,*2*)]
by (*induct rule*: *trancl.induct*) *auto*

— Steps in monadic lv system

fun *monadic-term* **where**
monadic-term (*Var x*) = *True*
| *monadic-term* (*Fun f* []) = *True*
| *monadic-term* (*Fun f ts*) = (*length ts* = *Suc 0* $\wedge$ *monadic-term* (*hd ts*))

fun *monadic-get-leave* **where**
monadic-get-leave (*Var x*) = (*Var x*)
| *monadic-get-leave* (*Fun f* []) = *Fun f* []
| *monadic-get-leave* (*Fun f ts*) = *monadic-get-leave* (*hd ts*)

fun *monadic-replace-leave* **where**
monadic-replace-leave t (*Var x*) = *t*
| *monadic-replace-leave t* (*Fun f* []) = *t*
| *monadic-replace-leave t* (*Fun f ts*) = *Fun f* [*monadic-replace-leave t* (*hd ts*)]

lemma *monadic-replace-leave-undo-const-subst*:
assumes *monadic-term s*
shows *monadic-replace-leave* (*monadic-get-leave s*) (*s* · *const-subst c*) = *s* **using** *assms*
proof (*induct s*)
case (*Fun f ts*) **then show** *?case*
by (*cases ts*) *auto*
qed *auto*

lemma *monadic-replace-leave-context*:
assumes *monadic-term* $C\langle s\rangle$
shows *monadic-replace-leave t* $C\langle s\rangle$ = $C\langle$*monadic-replace-leave t s*$\rangle$ **using** *assms*
proof (*induct C*)
case (*More f ss C ts*) **then show** *?case*
by (*cases ss*; *cases ts*) *auto*
qed *simp*

lemma *monadic-replace-leave-subst*:
assumes *monadic-term* (*s* · *σ*) ¬ *ground s*
shows *monadic-replace-leave t* (*s* · *σ*) = *s* · (*λ x*. *monadic-replace-leave t* (*σ x*))
using *assms*
proof (*induct s*)
case (*Fun f ts*) **then show** *?case*
by (*cases ts*) *auto*
qed *auto*

lemma *monadic-sig*:
monadic $\mathcal{F}$ ⟹ (*f*, *length ts*) ∈ $\mathcal{F}$ ⟹ *length ts* ≤ *Suc 0*
by (*auto simp*: *monadic-def*)

lemma *monadic-sig-funas-term-mt*:
monadic $\mathcal{F}$ ⟹ *funas-term s* ⊆ $\mathcal{F}$ ⟹ *monadic-term s*
proof (*induct s*)
case (*Fun f ts*) **then show** *?case* **unfolding** *monadic-def*
by (*cases ts*) *auto*
qed *simp*

lemma *monadic-term-const-pres* [*intro*]:
monadic-term s ⟹ *monadic-term* (*s* · *const-subst c*)
proof (*induct s*)
case (*Fun f ts*) **then show** *?case*
by (*cases ts*) *auto*
qed *simp*

lemma *remove-const-lv-mondaic-step-lhs*:
assumes *lv*: *lv* $\mathcal{R}$ **and** *fresh*: (*c*, *0*) ∉ *funas-rel* $\mathcal{R}$
and *mon*: *monadic* $\mathcal{F}$
and *step*: (*s* · *const-subst c*, *t*) ∈ (*srstep* $\mathcal{F}$ $\mathcal{R}$)

shows $(s,\ t) \in (srstep\ \mathcal{F}\ \mathcal{R})$
proof −
from *step* **obtain** $C\ l\ r\ \sigma$ **where** s: $(l,\ r) \in \mathcal{R}\ s \cdot const\text{-}subst\ c = C\langle l \cdot \sigma\rangle\ t = C\langle r \cdot \sigma\rangle$
by *fastforce*
have lv: $x \in vars\text{-}term\ l \Longrightarrow x \notin vars\text{-}term\ r$ **for** x **using** $s(1)\ lv$
by (*auto simp*: *lv-def*)
from $s(1)$ *fresh* **have** cl: $(c,\ 0) \notin funas\text{-}term\ l$ **by** (*auto simp*: *funas-rel-def*)
have *funas*: *funas-term* $s \subseteq \mathcal{F}\ (c,\ 0) \notin funas\text{-}term\ l\ funas\text{-}term\ t \subseteq \mathcal{F}$ **using** $s(1)$ *fresh*
using *step mon funas-term-subst* **unfolding** *funas-rel-def*
by (*auto dest*!: *srstepD*) *blast*
then have mt: *monadic-term s monadic-term* $(s \cdot const\text{-}subst\ c)$
using *monadic-sig-funas-term-mt*[*OF mon*] **by** *auto*
then have ml: *monadic-term* $(l \cdot \sigma)$ **unfolding** $s(2)$
by (*metis funas-ctxt-apply le-sup-iff step mon monadic-sig-funas-term-mt* $s(2)$ *sig-stepE*(*1*))
show *?thesis*
proof (*cases ground s*)
case *True* **then show** *?thesis* **using** *step*
by (*auto simp*: *ground-subst-apply*)
next
case *False* **note** ng = *this*
then have cs: $(c,\ 0) \in funas\text{-}term\ (s \cdot const\text{-}subst\ c)$
by (*auto simp*: *funas-term-subst vars-term-empty-ground*)
have $ngrl$: ¬ *ground l* **using** $s(2)\ cs\ mt\ ng$
proof (*induct s arbitrary*: C)
case (*Var x*) **then show** *?case* **using** $cl\ cs$
by (*cases C*) (*auto simp*: *funas-rel-def ground-subst-apply*)
next
case (*Fun f ts*)
from *Fun*(*5*−) **obtain** t **where** [*simp*]: $ts = [t]$ **by** (*cases ts*) *auto*
show *?case*
proof (*cases C* = *Hole*)
case *True* **then show** *?thesis* **using** *Fun*(*2*, *3*) cl
by (*auto simp*: *ground-subst-apply*)
next
case *False*
from *this Fun*(*2*, *3*) **obtain** D **where** [*simp*]: $C = More\ f\ []\ D\ []$
by (*cases C*) (*auto simp*: *Cons-eq-append-conv*)
show *?thesis* **using** *Fun*(*1*)[*of t D*] *Fun*(*2*−)
by *simp*
qed
qed
let $?\tau = \lambda\ x.\ if\ x \in vars\text{-}term\ l\ then\ monadic\text{-}replace\text{-}leave\ (monadic\text{-}get\text{-}leave\ s)\ (\sigma\ x)\ else\ (\sigma\ x)$
have $C\langle l \cdot (\lambda\ x.\ monadic\text{-}replace\text{-}leave\ (monadic\text{-}get\text{-}leave\ s)\ (\sigma\ x))\rangle = C\langle l \cdot\ ?\tau\rangle$
by (*auto intro*: *term-subst-eq*)

```
    then have s = C⟨l · ?τ⟩ using arg-cong[OF s(2), of monadic-replace-leave
(monadic-get-leave s),
      unfolded monadic-replace-leave-undo-const-subst[OF mt(1), of c],
      unfolded monadic-replace-leave-context[OF mt(2)[unfolded s(2)]],
      unfolded monadic-replace-leave-subst[OF ml ngrl]]
      by presburger
    moreover have t = C⟨r · ?τ⟩ using lv unfolding s(3)
      by (auto intro!: term-subst-eq)
    ultimately show ?thesis using s(1) funas(1, 3)
      by blast
  qed
qed

lemma remove-const-lv-mondaic-step-rhs:
  assumes lv: lv R and fresh: (c, 0) ∉ funas-rel R
    and mon: monadic F
    and step: (s, t · const-subst c) ∈ (srstep F R)
  shows (s, t) ∈ (srstep F R)
proof −
  have inv-v: lv (R^-1)(c, 0) ∉ funas-rel (R^-1) using fresh lv
    by (auto simp: funas-rel-def lv-def)
  have (t · const-subst c, s) ∈ (srstep F (R^-1)) using step
    by (auto simp: rew-converse-outwards)
  from remove-const-lv-mondaic-step-lhs[OF inv-v mon this]
  have (t, s) ∈ (srstep F (R^-1)) by simp
  then show ?thesis by (auto simp: rew-converse-outwards)
qed

lemma remove-const-lv-mondaic-steps-lhs:
  assumes lv: lv R and fresh: (c, 0) ∉ funas-rel R
    and mon: monadic F
    and steps: (s · const-subst c, t) ∈ (srstep F R)^+
  shows (s, t) ∈ (srstep F R)^+
  using remove-const-lv-mondaic-step-lhs[OF lv fresh mon] steps
  by (meson converse-tranclE r-into-trancl trancl-into-trancl2)

lemma remove-const-lv-mondaic-steps-rhs:
  assumes lv: lv R and fresh: (c, 0) ∉ funas-rel R
    and mon: monadic F
    and steps: (s, t · const-subst c) ∈ (srstep F R)^+
  shows (s, t) ∈ (srstep F R)^+
  using remove-const-lv-mondaic-step-rhs[OF lv fresh mon] steps
  by (meson trancl.simps)


lemma remove-const-lv-mondaic-steps:
  assumes lv: lv R and fresh: (c, 0) ∉ funas-rel R
    and mon: monadic F
    and steps: (s · const-subst c, t · const-subst c) ∈ (srstep F R)^+
```

shows $(s,\ t) \in (srstep\ \mathcal{F}\ \mathcal{R})^+$
using *remove-const-lv-mondaic-steps-rhs*[*OF lv fresh mon remove-const-lv-mondaic-steps-lhs*[*OF assms*]]
by *simp*

— Steps on lv trs

lemma *lv-root-step-idep-subst*:
assumes *lv* $\mathcal{R}$
and $(s,\ t) \in srrstep\ \mathcal{F}\ \mathcal{R}$
and *well*: $\bigwedge x.\ funas\text{-}term\ (\sigma\ x) \subseteq \mathcal{F} \bigwedge x.\ funas\text{-}term\ (\tau\ x) \subseteq \mathcal{F}$
shows $(s \cdot \sigma,\ t \cdot \tau) \in\ srrstep\ \mathcal{F}\ \mathcal{R}$
proof −
from *assms*(*2*) **obtain** *l r* γ **where** *mid*: $s = l \cdot \gamma\ t = r \cdot \gamma\ (l,\ r) \in \mathcal{R}$
by (*auto simp*: *sig-step-def*)
from *mid*(*3*) *assms*(*1*) **have** *vs*: $x \in vars\text{-}term\ l \Longrightarrow x \notin vars\text{-}term\ r$ **for** x
by (*auto simp*: *lv-def*)
let $?\sigma = \lambda\ x.\ if\ x \in vars\text{-}term\ l\ then\ (\gamma\ x) \cdot \sigma\ else\ (\gamma\ x) \cdot \tau$
have *subst*: $s \cdot \sigma = l \cdot ?\sigma\ t \cdot \tau = r \cdot ?\sigma$
unfolding *mid subst-subst-compose*[*symmetric*]
unfolding *term-subst-eq-conv*
by (*auto simp*: *subst-compose-def vs*)
then show *?thesis* **unfolding** *subst*
using *assms*(*2*) *mid*(*3*) *well* **unfolding** *mid*(*1*, *2*)
by (*auto simp*: *sig-step-def funas-term-subst*)
qed

lemma *lv-srsteps-with-root-step-idep-subst*:
assumes *lv* $\mathcal{R}$
and $(s,\ t) \in srsteps\text{-}with\text{-}root\text{-}step\ \mathcal{F}\ \mathcal{R}$
and *well*: $\bigwedge x.\ funas\text{-}term\ (\sigma\ x) \subseteq \mathcal{F} \bigwedge x.\ funas\text{-}term\ (\tau\ x) \subseteq \mathcal{F}$
shows $(s \cdot \sigma,\ t \cdot \tau) \in srsteps\text{-}with\text{-}root\text{-}step\ \mathcal{F}\ \mathcal{R}$ **using** *assms*(*2*)
using *lv-root-step-idep-subst*[*OF assms*(*1*) - *well*, **where** *?x1* = *id* **and** *?x2* = *id*]
using *srsteps-eq-subst-closed*[*OF* - *well*(*1*), **where** *?x1* = *id* **and** $?\mathcal{R} = \mathcal{R}$]
using *srsteps-eq-subst-closed*[*OF* - *well*(*2*), **where** *?x1* = *id* **and** $?\mathcal{R} = \mathcal{R}$]
by (*auto simp*: *srsteps-with-root-step-def*) (*metis* (*full-types*) *relcomp3-I*)

end
theory *Rewriting-GTRS*
imports *Rewriting*
Replace-Constant
begin

4.4 Specific results about rewriting under a ground system

abbreviation *ground-sys* $\mathcal{R} \equiv (\forall\ (s,\ t) \in \mathcal{R}.\ ground\ s \wedge ground\ t)$

lemma *srrstep-ground*:
assumes *ground-sys* $\mathcal{R}$
and $(s, t) \in$ *srrstep* $\mathcal{F}$ $\mathcal{R}$
shows *ground s ground t* **using** *assms*
by (*auto simp*: *sig-step-def ground-subst-apply vars-term-subst elim*!: *rrstep-subst*)

lemma *srstep-pres-ground-l*:
assumes *ground-sys* $\mathcal{R}$ *ground s*
and $(s, t) \in$ *srstep* $\mathcal{F}$ $\mathcal{R}$
shows *ground t* **using** *assms*
by (*auto simp*: *sig-step-def ground-subst-apply dest*!: *rstep-imp-C-s-r*)

lemma *srstep-pres-ground-r*:
assumes *ground-sys* $\mathcal{R}$ *ground t*
and $(s, t) \in$ *srstep* $\mathcal{F}$ $\mathcal{R}$
shows *ground s* **using** *assms*
by (*auto simp*: *ground-vars-term-empty vars-term-subst sig-step-def vars-term-ctxt-apply ground-subst-apply dest*!: *rstep-imp-C-s-r*)

lemma *srsteps-pres-ground-l*:
assumes *ground-sys* $\mathcal{R}$ *ground s*
and $(s, t) \in (srstep\ \mathcal{F}\ \mathcal{R})^+$
shows *ground t* **using** *assms*(*3*, *2*) *srstep-pres-ground-l*[*OF assms*(*1*)]
by (*induct rule*: *converse-trancl-induct*) *auto*

lemma *srsteps-pres-ground-r*:
assumes *ground-sys* $\mathcal{R}$ *ground t*
and $(s, t) \in (srstep\ \mathcal{F}\ \mathcal{R})^+$
shows *ground s* **using** *assms*(*3*, *2*) *srstep-pres-ground-r*[*OF assms*(*1*)]
by (*induct rule*: *converse-trancl-induct*) *auto*

lemma *srsteps-eq-pres-ground-l*:
assumes *ground-sys* $\mathcal{R}$ *ground s*
and $(s, t) \in (srstep\ \mathcal{F}\ \mathcal{R})^*$
shows *ground t* **using** *srsteps-pres-ground-l*[*OF assms*(*1*, *2*)] *assms*(*2*, *3*)
by (*auto simp*: *rtrancl-eq-or-trancl*)

lemma *srsteps-eq-pres-ground-r*:
assumes *ground-sys* $\mathcal{R}$ *ground t*
and $(s, t) \in (srstep\ \mathcal{F}\ \mathcal{R})^*$
shows *ground s* **using** *srsteps-pres-ground-r*[*OF assms*(*1*, *2*)] *assms*(*2*, *3*)
by (*auto simp*: *rtrancl-eq-or-trancl*)

lemma *srsteps-with-root-step-ground*:
assumes *ground-sys* $\mathcal{R}$
and $(s, t) \in$ *srsteps-with-root-step* $\mathcal{F}$ $\mathcal{R}$
shows *ground s ground t* **using** *srrstep-ground*[*OF assms*(*1*)]
using *srsteps-eq-pres-ground-l*[*OF assms*(*1*)]

using *srsteps-eq-pres-ground-r*[*OF assms*(*1*)]
using *assms*(*2*) **unfolding** *srsteps-with-root-step-def*
by (*meson relcomp.cases*)+

4.5 funas

lemma *srrstep-funas*:
assumes *ground-sys* $\mathcal{R}$
and (*s*, *t*) ∈ *srrstep* $\mathcal{F}$ $\mathcal{R}$
shows *funas-term s* ⊆ *funas-rel* $\mathcal{R}$ *funas-term t* ⊆ *funas-rel* $\mathcal{R}$ **using** *assms*
by (*auto simp*: *sig-step-def funas-term-subst ground-vars-term-empty funas-rel-def split*: *prod.splits elim*!: *rrstep-subst*)

lemma *srstep-funas-l*:
assumes *ground-sys* $\mathcal{R}$
and (*s*, *t*) ∈ *srstep* $\mathcal{F}$ $\mathcal{R}$
shows *funas-term t* ⊆ *funas-term s* ∪ *funas-rel* $\mathcal{R}$ **using** *assms*
by (*auto simp*: *ground-vars-term-empty vars-term-subst sig-step-def vars-term-ctxt-apply funas-term-subst funas-rel-def split*: *prod.splits dest*!: *rstep-imp-C-s-r*)

lemma *srstep-funas-r*:
assumes *ground-sys* $\mathcal{R}$
and (*s*, *t*) ∈ *srstep* $\mathcal{F}$ $\mathcal{R}$
shows *funas-term s* ⊆ *funas-term t* ∪ *funas-rel* $\mathcal{R}$ **using** *assms*
by (*auto simp*: *ground-vars-term-empty vars-term-subst sig-step-def vars-term-ctxt-apply funas-term-subst funas-rel-def split*: *prod.splits dest*!: *rstep-imp-C-s-r*)

lemma *srsteps-funas-l*:
assumes *ground-sys* $\mathcal{R}$
and (*s*, *t*) ∈ (*srstep* $\mathcal{F}$ $\mathcal{R}$)$^+$
shows *funas-term t* ⊆ *funas-term s* ∪ *funas-rel* $\mathcal{R}$ **using** *assms*(*2*)
by (*induct rule*: *converse-trancl-induct*) (*auto dest*: *srstep-funas-l*[*OF assms*(*1*)])

lemma *srsteps-funas-r*:
assumes *ground-sys* $\mathcal{R}$
and (*s*, *t*) ∈ (*srstep* $\mathcal{F}$ $\mathcal{R}$)$^+$
shows *funas-term s* ⊆ *funas-term t* ∪ *funas-rel* $\mathcal{R}$ **using** *assms*(*2*)
by (*induct rule*: *converse-trancl-induct*) (*auto dest*: *srstep-funas-r*[*OF assms*(*1*)])

lemma *srsteps-eq-funas-l*:
assumes *ground-sys* $\mathcal{R}$
and (*s*, *t*) ∈ (*srstep* $\mathcal{F}$ $\mathcal{R}$)*
shows *funas-term t* ⊆ *funas-term s* ∪ *funas-rel* $\mathcal{R}$ **using** *srsteps-funas-l*[*OF assms*(*1*)] *assms*(*2*)
by (*auto simp*: *rtrancl-eq-or-trancl*)

lemma *srsteps-eq-funas-r*:
assumes *ground-sys* $\mathcal{R}$
and (*s*, *t*) ∈ (*srstep* $\mathcal{F}$ $\mathcal{R}$)*

shows *funas-term* $s \subseteq$ *funas-term* $t \cup$ *funas-rel* $\mathcal{R}$ **using** *srsteps-funas-r*[*OF assms*(*1*)] *assms*(*2*)
by (*auto simp*: *rtrancl-eq-or-trancl*)

lemma *srsteps-with-root-step-funas*:
assumes *ground-sys* $\mathcal{R}$
and (s, t) $\in$ *srsteps-with-root-step* $\mathcal{F}$ $\mathcal{R}$
shows *funas-term* $s \subseteq$ *funas-rel* $\mathcal{R}$ *funas-term* $t \subseteq$ *funas-rel* $\mathcal{R}$
using *srrstep-funas*[*OF assms*(*1*)]
using *srsteps-eq-funas-l*[*OF assms*(*1*)]
using *srsteps-eq-funas-r*[*OF assms*(*1*)]
using *assms*(*2*) **unfolding** *srsteps-with-root-step-def*
by (*metis relcompEpair sup-absorb2*)+

end

5 Reducing Rewrite Properties to Properties on Ground Terms over Left-Linear Right-Ground Systems

theory *Ground-Reduction-on-LLRG*
imports
Rewriting-Properties
Rewriting-LLRG-LV-Mondaic
begin

lemma *llrg-linear-sys*:
llrg $\mathcal{R} \Longrightarrow$ *linear-sys* $\mathcal{R}$
by (*auto simp*: *llrg-def*)

6 LLRG results

lemma *llrg-commute*:
assumes *sig*: *funas-rel* $\mathcal{R} \subseteq \mathcal{F}$ *funas-rel* $\mathcal{S} \subseteq \mathcal{F}$
and *fresh*: (c, 0) $\notin \mathcal{F}$
and *llrg*: *llrg* $\mathcal{R}$ *llrg* $\mathcal{S}$
and *com*: *commute* (*gsrstep* (*insert* (c, 0) $\mathcal{F}$) $\mathcal{R}$) (*gsrstep* (*insert* (c, 0) $\mathcal{F}$) $\mathcal{S}$)
shows *commute* (*srstep* $\mathcal{F}$ $\mathcal{R}$) (*srstep* $\mathcal{F}$ $\mathcal{S}$)
proof −
let *?*$\sigma = \lambda$ -. *Fun* c [] **let** *?*$\mathcal{H}$ = *insert* (c, 0) $\mathcal{F}$
from *fresh* **have** *fresh-sys*: (c, 0) $\notin$ *funas-rel* $\mathcal{S}$ (c, 0) $\notin$ *funas-rel* ($\mathcal{R}^{-1}$) **using** *sig*
by (*auto simp*: *funas-rel-def*)
have *linear*: *linear-sys* $\mathcal{S}$ *linear-sys* ($\mathcal{R}^{-1}$) **using** *llrg* **unfolding** *llrg-def* **by** *auto*
have *sig′*: *funas-rel* $\mathcal{S} \subseteq \mathcal{F}$ *funas-rel* ($\mathcal{R}^{-1}$) $\subseteq \mathcal{F}$ **using** *sig* **by** (*auto simp*: *funas-rel-def*)
have *mono*: $\mathcal{F} \subseteq$ *?*$\mathcal{H}$ **by** *auto*

```
  {fix s t assume ass: (s, t) ∈ comp-rrstep-rel' F (R^-1) S
   from ass have funas: funas-term s ⊆ F funas-term t ⊆ F
     by (metis Un-iff relcomp.cases srstepsD srsteps-with-root-step-srstepsD)+
   from ass have m: (s, t) ∈ (srstep ?H (R^-1))* O (srstep ?H S)*
     using rtrancl-mono[OF sig-step-mono[OF mono]]
     by (auto dest!: srsteps-with-root-step-sresteps-eqD trancl-into-rtrancl)
   have gr: ground s ∨ ground t using ass llrg
     unfolding srstep-converse-dist trancl-converse
    by (auto simp: llrg-srsteps-with-root-step-inv-ground llrg-srsteps-with-root-step-ground)
   have gstep: (s · ?σ, t · ?σ) ∈ (gsrstep ?H S)* O (gsrstep ?H (R^-1))*
    using srsteps-eq-subst-relcomp-closed[OF m, THEN srsteps-eq-relcomp-gsrsteps-relcomp,
of ?σ,
     THEN subsetD[OF com[unfolded commute-def], unfolded rew-converse-inwards]]
     by (auto simp: ground-substI)
  have [simp]: ground s ⟹ (c, 0) ∉ funas-term s ground t ⟹ (c, 0) ∉ funas-term
t
     using funas fresh by auto
   have (s, t) ∈ (rstep S)* O (rstep (R^-1))*
     using remove-const-subst-relcomp-lhs[OF linear fresh-sys, of t s]
     using gsrsteps-eq-relcomp-to-rsteps-relcomp[OF gstep] gr
     using remove-const-subst-relcomp-lhs[OF linear fresh-sys, of t s]
     using remove-const-subst-relcomp-rhs[OF linear fresh-sys, of s t]
       by (cases ground s) (simp-all add: ground-subst-apply)
   from rsteps-eq-relcomp-srsteps-eq-relcompI[OF sig' funas this]
   have commute-redp F R S s t unfolding commute-redp-def
     by (simp add: rew-converse-inwards)}
  then show ?thesis by (intro commute-rrstep-intro) simp
qed

lemma llrg-CR:
  assumes sig: funas-rel R ⊆ F
   and fresh: (c, 0) ∉ F
   and llrg: llrg R
   and com: CR (gsrstep (insert (c, 0) F) R)
  shows CR (srstep F R)
  using assms llrg-commute unfolding CR-iff-self-commute
  by metis


lemma llrg-SCR:
  assumes sig: funas-rel R ⊆ F and fresh: (c, 0) ∉ F
   and llrg: llrg R
   and scr: SCR (gsrstep (insert (c, 0) F) R)
  shows SCR (srstep F R)
proof −
  let ?σ = const-subst c let ?H = insert (c, 0) F
  from fresh have fresh-sys: (c, 0) ∉ funas-rel R using sig
    by (auto simp: funas-rel-def)
   have mono: F ⊆ ?H by auto note sig-trans = subsetD[OF srstep-monp[OF
```

mono]]
{fix *s t u* **assume** *ass*: $(s, t) \in$ *srstep* $\mathcal{F}$ $\mathcal{R}$ $(s, u) \in$ *srstep* $\mathcal{F}$ $\mathcal{R}$
and *root*: $(s, t) \in$ *sig-step* $\mathcal{F}$ (*rrstep* $\mathcal{R}$) $\vee$ $(s, u) \in$ *sig-step* $\mathcal{F}$ (*rrstep* $\mathcal{R}$)
from *ass* **have** *funas*: *funas-term* $s \subseteq \mathcal{F}$ *funas-term* $t \subseteq \mathcal{F}$ *funas-term* $u \subseteq \mathcal{F}$
by (*force dest*: *sig-stepE*)+
from *root* **have** *gr*: *ground* $t \vee$ *ground* u **using** *ass llrg llrg-rrsteps-groundness*
unfolding *srstep-converse-dist trancl-converse* **by** *blast*
have $*$: *ground* $(s \cdot ?\sigma)$ *ground* $(t \cdot ?\sigma)$ *ground* $(u \cdot ?\sigma)$ **by** *auto*
from *this scr* **obtain** *v* **where** *v*: $(t \cdot ?\sigma, v) \in$ (*gsrstep* $?\mathcal{H}$ $\mathcal{R}$)$^=$ $\wedge$ $(u \cdot ?\sigma, v)$ $\in$ (*gsrstep* $?\mathcal{H}$ $\mathcal{R}$)*
using *srstep-subst-closed*[*OF sig-trans*[*OF ass*(*1*)], *of* $?\sigma$]
using *srstep-subst-closed*[*OF sig-trans*[*OF ass*(*2*)], *of* $?\sigma$]
using *ass* **unfolding** *SCR-on-def*
by *auto* (*metis* (*no-types*, *lifting*) $*$)
then have *fv*: *funas-term* $v \subseteq \mathcal{F}$ **using** *gr llrg-funas-term-step-pres*[*OF llrg*, *of* - *v*]
using *llrg-funas-term-steps-pres*[*OF llrg*, *of* - *v*] *funas sig*
by (*auto simp*: *ground-subst-apply elim*!: *sig-stepE dest*!: *gsrsteps-eq-to-rsteps-eq*) *blast*+
then have *c-free*: $(c, 0) \notin$ *funas-term* *v* **using** *fresh* **by** *blast*
then have $v = t \cdot$ *const-subst* $c \Longrightarrow$ *ground* *t* **using** *arg-cong*[*of v t* $\cdot$ $?\sigma$ *funas-term*]
by (*auto simp*: *funas-term-subst vars-term-empty-ground split*: *if-splits*)
from *this v* **have** $(t, v) \in$ (*srstep* $\mathcal{F}$ $\mathcal{R}$)$^=$ $(u, v) \in$ (*srstep* $\mathcal{F}$ $\mathcal{R}$)*
using *remove-const-subst-steps-eq-lhs*[*OF llrg-linear-sys*[*OF llrg*] *fresh-sys c-free*, *of u*,
THEN rsteps-eq-srsteps-eqI[*OF sig funas*(*3*) *fv*]]
using *remove-const-subst-step-lhs*[*OF llrg-linear-sys*[*OF llrg*] *fresh-sys c-free*, *of t*,
THEN sig-stepI[*OF funas*(*2*) *fv*]]
by (*auto simp*: *ground-subst-apply dest*: *srstepD gsrsteps-eq-to-rsteps-eq*)
then have *SCRp* $\mathcal{F}$ $\mathcal{R}$ *t u* **by** *blast***}**
then show *?thesis* **by** (*intro SCR-rrstep-intro*) (*metis srrstep-to-srestep*)+
qed

lemma *llrg-WCR*:
assumes *sig*: *funas-rel* $\mathcal{R} \subseteq \mathcal{F}$ **and** *fresh*: $(c, 0) \notin \mathcal{F}$
and *llrg*: *llrg* $\mathcal{R}$
and *wcr*: *WCR* (*gsrstep* (*insert* $(c, 0)$ $\mathcal{F}$) $\mathcal{R}$)
shows *WCR* (*srstep* $\mathcal{F}$ $\mathcal{R}$)
proof −
let $?\sigma$ = *const-subst c* **let** $?\mathcal{H}$ = *insert* $(c, 0)$ $\mathcal{F}$
from *fresh* **have** *fresh-sys*: $(c, 0) \notin$ *funas-rel* $\mathcal{R}$ $(c, 0) \notin$ *funas-rel* $(\mathcal{R}^{-1})$ **using** *sig*
by (*auto simp*: *funas-rel-def*)
have *lin*: *linear-sys* $\mathcal{R}$ *linear-sys* $(\mathcal{R}^{-1})$ **using** *llrg* **unfolding** *llrg-def* **by** *auto*
have *mono*: $\mathcal{F} \subseteq ?\mathcal{H}$ **by** *auto* **note** *sig-trans* = *subsetD*[*OF srstep-monp*[*OF mono*]]
{fix *s t u* **assume** *ass*: $(s, t) \in$ *sig-step* $\mathcal{F}$ (*rrstep* $\mathcal{R}$) $(s, u) \in$ *srstep* $\mathcal{F}$ $\mathcal{R}$

from *ass* **have** *funas*: *funas-term* $s \subseteq \mathcal{F}$ *funas-term* $t \subseteq \mathcal{F}$ *funas-term* $u \subseteq \mathcal{F}$
by *blast*+
then have *c-free*: $(c,\ 0) \notin$ *funas-term* t **using** *fresh* **by** *blast*
from *ass* **have** *gr*: *ground* t **using** *ass llrg llrg-rrsteps-groundness* **by** *blast*
have $*$: *ground* $(s \cdot ?\sigma)$ *ground* $(u \cdot ?\sigma)$ **by** *auto*
from *this wcr* **have** *w*: $(t,\ u \cdot ?\sigma) \in (gsrstep\ ?\mathcal{H}\ \mathcal{R})^{\downarrow}$
using *srstep-subst-closed*[*OF sig-trans*[*OF srrstep-to-srestep*[*OF ass*(*1*)]], *of* *?σ*]
using *srstep-subst-closed*[*OF sig-trans*[*OF ass*(*2*)], *of* *?σ*]
using *ass* **unfolding** *WCR-on-def*
by *auto* (*metis* $*$ *gr ground-subst-apply*)
have $(t,\ u) \in (srstep\ \mathcal{F}\ \mathcal{R})^{\downarrow}$ **unfolding** *join-def*
using *remove-const-subst-relcomp-rhs*[*OF lin fresh-sys c-free*
gsrsteps-eq-relcomp-to-rsteps-relcomp[*OF w*[*unfolded join-def rew-converse-inwards*]],
THEN rsteps-eq-relcomp-srsteps-eq-relcompI[*OF sig funas-rel-converse*[*OF* *sig*] *funas*(*2*−)]]
by (*metis* (*no-types*, *lifting*) *srstep-converse-dist*)**}**
then show *?thesis* **by** (*intro WCR-rrstep-intro*) *simp*
qed

lemma *llrg-UNF*:
assumes *sig*: *funas-rel* $\mathcal{R} \subseteq \mathcal{F}$ **and** *fresh*: $(c,\ 0) \notin \mathcal{F}$
and *llrg*: *llrg* $\mathcal{R}$
and *unf*: *UNF* (*gsrstep* (*insert* $(c,\ 0)\ \mathcal{F}$) $\mathcal{R}$)
shows *UNF* (*srstep* $\mathcal{F}\ \mathcal{R}$)
proof −
let *?σ* = *const-subst c* **let** *?H* = *insert* $(c,\ 0)\ \mathcal{F}$
from *fresh* **have** *fresh-sys*: $(c,\ 0) \notin$ *funas-rel* $\mathcal{R}$ **using** *sig*
by (*auto simp*: *funas-rel-def*)
have *mono*: $\mathcal{F} \subseteq ?\mathcal{H}$ **by** *auto*
{fix *s t* **assume** *ass*: $(s,\ t) \in$ *comp-rrstep-rel'* $\mathcal{F}$ $(\mathcal{R}^{-1})$ $\mathcal{R}$
from *ass* **have** *funas*: *funas-term* $s \subseteq \mathcal{F}$ *funas-term* $t \subseteq \mathcal{F}$
by (*metis Un-iff relcomp.cases srstepsD srsteps-with-root-step-srstepsD*)+
from *ass* **have** *m*: $(s,\ t) \in (srstep\ ?\mathcal{H}\ (\mathcal{R}^{-1}))^*\ O\ (srstep\ ?\mathcal{H}\ \mathcal{R})^*$
using *rtrancl-mono*[*OF sig-step-mono*[*OF mono*]]
by (*auto dest*!: *srsteps-with-root-step-sresteps-eqD trancl-into-rtrancl*)
have *gr*: *ground* $s \vee$ *ground* t **using** *ass llrg*
unfolding *srstep-converse-dist trancl-converse*
by (*auto simp*: *llrg-srsteps-with-root-step-inv-ground llrg-srsteps-with-root-step-ground*)
have *wit*: $s \cdot ?\sigma \in NF\ (gsrstep\ ?\mathcal{H}\ \mathcal{R}) \Longrightarrow t \cdot ?\sigma \in NF\ (gsrstep\ ?\mathcal{H}\ \mathcal{R}) \Longrightarrow s \cdot ?\sigma = t \cdot ?\sigma$ **using** *unf*
using *srsteps-eq-subst-relcomp-closed*[*OF m*, *THEN srsteps-eq-relcomp-gsrsteps-relcomp*, *of* *?σ*]
by (*auto simp*: *UNF-on-def rew-converse-outwards normalizability-def*)
from *funas NF-to-fresh-const-subst-NF*[*OF llrg-linear-sys*[*OF llrg*] *fresh-sys* *sig*(*1*)]
have $s \in NF\ (srstep\ \mathcal{F}\ \mathcal{R}) \Longrightarrow s \cdot ?\sigma \in NF\ (gsrstep\ ?\mathcal{H}\ \mathcal{R})\ t \in NF\ (srstep\ \mathcal{F}\ \mathcal{R}) \Longrightarrow t \cdot ?\sigma \in NF\ (gsrstep\ ?\mathcal{H}\ \mathcal{R})$

```
    by auto
   moreover have s · ?σ = t · ?σ ⟹ s = t using gr funas fresh
   by (cases ground s) (auto simp: ground-subst-apply funas-term-subst vars-term-empty-ground
split: if-splits)
   ultimately have UN-redp F R s t using wit unfolding UN-redp-def
    by auto}
 then show ?thesis by (intro UNF-rrstep-intro) simp
qed


lemma llrg-NFP:
 assumes sig: funas-rel R ⊆ F and fresh: (c, 0) ∉ F
  and llrg: llrg R
  and nfp: NFP (gsrstep (insert (c, 0) F) R)
 shows NFP (srstep F R)
proof −
 let ?σ = const-subst c let ?H = insert (c, 0) F
 from fresh have fresh-sys: (c, 0) ∉ funas-rel R
  using sig by (auto simp: funas-rel-def)
 have mono: F ⊆ ?H by auto
 {fix s t assume ass: (s, t) ∈ comp-rrstep-rel' F (R^-1) R
  from ass have funas: funas-term s ⊆ F funas-term t ⊆ F
   by (metis Un-iff relcomp.cases srstepsD srsteps-with-root-step-srstepsD)+
  from ass have m: (s, t) ∈ (srstep ?H (R^-1))* O (srstep ?H R)*
   using rtrancl-mono[OF sig-step-mono[OF mono]]
   by (auto dest!: srsteps-with-root-step-sresteps-eqD trancl-into-rtrancl)
  have gr: ground s ∨ ground t using ass llrg
   unfolding srstep-converse-dist trancl-converse
  by (auto simp: llrg-srsteps-with-root-step-inv-ground llrg-srsteps-with-root-step-ground)
  have wit: t · ?σ ∈ NF (gsrstep ?H R) ⟹ (s · ?σ, t · ?σ) ∈ (gsrstep (insert
(c, 0) F) R)*
   using NFP-stepD[OF nfp]
  using srsteps-eq-subst-relcomp-closed[OF m, THEN srsteps-eq-relcomp-gsrsteps-relcomp,
of ?σ]
   by (auto simp: rew-converse-outwards)
  from funas NF-to-fresh-const-subst-NF[OF llrg-linear-sys[OF llrg] fresh-sys(1)
sig(1)]
  have t ∈ NF (srstep F R) ⟹ t · ?σ ∈ NF (gsrstep ?H R) by auto
  moreover have (s · ?σ, t · ?σ) ∈ (rstep R)* ⟹ (s, t) ∈ (srstep F R)* using
gr funas fresh
    using remove-const-subst-steps-eq-lhs[OF llrg-linear-sys[OF llrg] fresh-sys,
THEN rsteps-eq-srsteps-eqI[OF sig funas]]
    using remove-const-subst-steps-eq-rhs[OF llrg-linear-sys[OF llrg] fresh-sys,
THEN rsteps-eq-srsteps-eqI[OF sig funas]]
   by (cases ground s) (auto simp: ground-subst-apply)
  ultimately have NFP-redp F R s t using wit unfolding NFP-redp-def
   by (auto dest: gsrsteps-eq-to-rsteps-eq)}
 then show ?thesis by (intro NFP-rrstep-intro) simp
qed
```

lemma *llrg-NE-aux*:
assumes $(s,\ t) \in$ *srsteps-with-root-step* $\mathcal{F}\ \mathcal{R}$
and *sig*: *funas-rel* $\mathcal{R} \subseteq \mathcal{F}$ *funas-rel* $\mathcal{S} \subseteq \mathcal{F}$ **and** *fresh*: $(c,\ 0) \notin \mathcal{F}$
and *llrg*: *llrg* $\mathcal{R}$ *llrg* $\mathcal{S}$
and *ne*: *NE* (*gsrstep* (*insert* $(c,\ 0)\ \mathcal{F}$) $\mathcal{R}$) (*gsrstep* (*insert* $(c,\ 0)\ \mathcal{F}$) $\mathcal{S}$)
shows *NE-redp* $\mathcal{F}\ \mathcal{R}\ \mathcal{S}\ s\ t$
proof −
from *fresh* **have** *fresh-sys*: $(c,\ 0) \notin$ *funas-rel* $\mathcal{R}$ $(c,\ 0) \notin$ *funas-rel* $\mathcal{S}$
using *sig* **by** (*auto simp*: *funas-rel-def*)
let $?\sigma$ = *const-subst* c **let** $?\mathcal{H}$ = *insert* $(c,\ 0)\ \mathcal{F}$
let $?\mathcal{H}$ = *insert* $(c,\ 0)\ \mathcal{F}$ **have** *mono*: $\mathcal{F} \subseteq ?\mathcal{H}$ **by** *auto*
from *assms*(*1*) **have** *gr*: *ground* t **and** *funas*: *funas-term* $s \subseteq \mathcal{F}$ *funas-term* $t \subseteq \mathcal{F}$
using *llrg-srsteps-with-root-step-ground*[*OF llrg*(*1*)]
by (*meson srstepsD srsteps-with-root-step-srstepsD*)+
from *funas* **have** *fresh-t*: $(c,\ 0) \notin$ *funas-term* t **using** *fresh* **by** *auto*
from *srsteps-subst-closed*[*OF srsteps-monp*[*OF mono*, *THEN subsetD*, *OF srsteps-with-root-step-srstepsD*[*OF assms*(*1*)]], *of* $?\sigma$]
have $(s \cdot ?\sigma,\ t) \in$ (*gsrstep*$?\mathcal{H}\ \mathcal{R})^+$ **using** *gr*
by (*auto simp*: *ground-subst-apply intro*!: *ground-srsteps-gsrsteps*)
then have $t \in$ *NF* (*srstep* $\mathcal{F}\ \mathcal{R}) \Longrightarrow (s \cdot ?\sigma,\ t) \in$ (*gsrstep* (*insert* $(c,\ 0)\ \mathcal{F}$) $\mathcal{S})^*$
using *NF-to-fresh-const-subst-NF*[*OF llrg-linear-sys*[*OF llrg*(*1*)] *fresh-sys*(*1*) *sig*(*1*) *funas*(*2*), *of* $?\mathcal{H}$]
using *gr NE-NF-eq*[*OF ne*, *symmetric*] *ne* **unfolding** *NE-on-def*
by (*auto simp*: *normalizability-def ground-subst-apply dest*!: *trancl-into-rtrancl*)
then show *NE-redp* $\mathcal{F}\ \mathcal{R}\ \mathcal{S}\ s\ t$ **unfolding** *NE-redp-def*
using *remove-const-subst-steps-eq-lhs*[*OF llrg-linear-sys*[*OF llrg*(*2*)] *fresh-sys*(*2*) *fresh-t*,
THEN rsteps-eq-srsteps-eqI[*OF sig*(*2*) *funas*]]
by (*auto dest*: *gsrsteps-eq-to-rsteps-eq*)
qed

lemma *llrg-NE*:
assumes *sig*: *funas-rel* $\mathcal{R} \subseteq \mathcal{F}$ *funas-rel* $\mathcal{S} \subseteq \mathcal{F}$ **and** *fresh*: $(c,\ 0) \notin \mathcal{F}$
and *llrg*: *llrg* $\mathcal{R}$ *llrg* $\mathcal{S}$
and *ne*: *NE* (*gsrstep* (*insert* $(c,\ 0)\ \mathcal{F}$) $\mathcal{R}$) (*gsrstep* (*insert* $(c,\ 0)\ \mathcal{F}$) $\mathcal{S}$)
shows *NE* (*srstep* $\mathcal{F}\ \mathcal{R}$) (*srstep* $\mathcal{F}\ \mathcal{S}$)
proof −
have *f*: $(c,\ 0) \notin$ *funas-rel* $\mathcal{R}$ $(c,\ 0) \notin$ *funas-rel* $\mathcal{S}$ **using** *fresh sig*
by (*auto simp*: *funas-rel-def*)
{fix $s\ t$ **assume** $(s,\ t) \in$ *srsteps-with-root-step* $\mathcal{F}\ \mathcal{R}$
from *llrg-NE-aux*[*OF this sig fresh llrg ne*] **have** *NE-redp* $\mathcal{F}\ \mathcal{R}\ \mathcal{S}\ s\ t$
by *simp***}**
moreover
{fix $s\ t$ **assume** $(s,\ t) \in$ *srsteps-with-root-step* $\mathcal{F}\ \mathcal{S}$
from *llrg-NE-aux*[*OF this sig*(*2*, *1*) *fresh llrg*(*2*, *1*) *NE-symmetric*[*OF ne*]]

have *NE-redp* $\mathcal{F}$ $\mathcal{S}$ $\mathcal{R}$ *s t* **by** *simp***}**
ultimately show *?thesis* **using** *linear-sys-gNF-eq-NF-eq*[*OF llrg-linear-sys*[*OF llrg*(*1*)]
llrg-linear-sys[*OF llrg*(*2*)] *sig f - - NE-NF-eq*[*OF ne*]]
by (*intro NE-rrstep-intro*) *auto*
qed

6.1 Specialized for monadic signature

lemma *monadic-commute*:
assumes *llrg* $\mathcal{R}$ *llrg* $\mathcal{S}$ *monadic* $\mathcal{F}$
and *com*: *commute* (*gsrstep* $\mathcal{F}$ $\mathcal{R}$) (*gsrstep* $\mathcal{F}$ $\mathcal{S}$)
shows *commute* (*srstep* $\mathcal{F}$ $\mathcal{R}$) (*srstep* $\mathcal{F}$ $\mathcal{S}$)
proof −
{fix *s t* **assume** *ass*: $(s, t) \in$ *comp-rrstep-rel'* $\mathcal{F}$ $(\mathcal{R}^{-1})$ $\mathcal{S}$
then obtain *u* **where** *steps*: $(s, u) \in (srstep\ \mathcal{F}\ (\mathcal{R}^{-1}))^+$ $(u, t) \in (srstep\ \mathcal{F}\ \mathcal{S})^+$
by (*auto simp*: *sig-step-converse-rstep dest*: *srsteps-with-root-step-srstepsD*)
have *gr*: *ground s ground t* **using** *steps assms*(*1* − *3*)
unfolding *rew-converse-outwards*
by (*auto simp*: *llrg-srsteps-with-root-step-ground llrg-monadic-rsteps-groundness*)
from *steps*(*1*) **have** *f*: *funas-term s* $\subseteq \mathcal{F}$ **using** *srstepsD* **by** *blast*
let $?\sigma = \lambda$ *-. s*
from *steps gr* **have** $(s, u \cdot ?\sigma) \in (gsrstep\ \mathcal{F}\ (\mathcal{R}^{-1}))^+$ $(u \cdot ?\sigma, t) \in (gsrstep\ \mathcal{F}\ \mathcal{S})^+$
unfolding *rew-converse-outwards*
using *srsteps-subst-closed*[**where** $?\sigma = ?\sigma$ **and** $?s = u$, *of - F*] *f*
by (*force simp*: *ground-subst-apply intro*: *ground-srsteps-gsrsteps*)+
then have $(s, t) \in (gsrstep\ \mathcal{F}\ \mathcal{S})^*$ *O* $(gsrstep\ \mathcal{F}\ (\mathcal{R}^{-1}))^*$ **using** *com*
unfolding *commute-def rew-converse-inwards*
by (*meson relcomp.relcompI subsetD trancl-into-rtrancl*)
from *gsrsteps-eq-relcomp-srsteps-relcompD*[*OF this*]
have *commute-redp* $\mathcal{F}$ $\mathcal{R}$ $\mathcal{S}$ *s t* **unfolding** *commute-redp-def*
by (*simp add*: *rew-converse-inwards*)**}**
then show *?thesis* **by** (*intro commute-rrstep-intro*) *simp*
qed

lemma *monadic-CR*:
assumes *llrg* $\mathcal{R}$ *monadic* $\mathcal{F}$
and *CR* (*gsrstep* $\mathcal{F}$ $\mathcal{R}$)
shows *CR* (*srstep* $\mathcal{F}$ $\mathcal{R}$) **using** *monadic-commute assms*
unfolding *CR-iff-self-commute*
by *blast*

lemma *monadic-SCR*:
assumes *sig*: *funas-rel* $\mathcal{R} \subseteq \mathcal{F}$ *monadic* $\mathcal{F}$
and *llrg*: *llrg* $\mathcal{R}$
and *scr*: *SCR* (*gsrstep* $\mathcal{F}$ $\mathcal{R}$)

shows *SCR* (*srstep* $\mathcal{F}$ $\mathcal{R}$)
proof −
{fix *s t u* **assume** *ass*: (*s*, *t*) ∈ *srstep* $\mathcal{F}$ $\mathcal{R}$ (*s*, *u*) ∈ *srstep* $\mathcal{F}$ $\mathcal{R}$
and *root*: (*s*, *t*) ∈ *sig-step* $\mathcal{F}$ (*rrstep* $\mathcal{R}$) ∨ (*s*, *u*) ∈ *sig-step* $\mathcal{F}$ (*rrstep* $\mathcal{R}$)
from *ass* **have** *funas*: *funas-term s* ⊆ $\mathcal{F}$ *funas-term t* ⊆ $\mathcal{F}$ *funas-term u* ⊆ $\mathcal{F}$
by *blast*+
from *root* **have** *gr*: *ground t ground u* **using** *ass sig*(*2*) *llrg*
by (*metis llrg-monadic-rstep-pres-groundness*)+
let *?σ* = λ -. *t* **have** *grs*: *ground* (*s* · *?σ*) **using** *gr* **by** *auto*
from *this scr* **obtain** *v* **where** *v*: (*t*, *v*) ∈ (*gsrstep* $\mathcal{F}$ $\mathcal{R}$)$^=$ ∧ (*u*, *v*) ∈ (*gsrstep* $\mathcal{F}$ $\mathcal{R}$)*
using *srstep-subst-closed*[*OF ass*(*1*), *of ?σ*]
using *srstep-subst-closed*[*OF ass*(*2*), *of ?σ*]
using *gr* **unfolding** *SCR-on-def*
by (*metis Int-iff UNIV-I funas*(*2*) *ground-subst-apply mem-Collect-eq mem-Sigma-iff*)
then have *SCRp* $\mathcal{F}$ $\mathcal{R}$ *t u*
by (*metis Int-iff Un-iff gsrsteps-eq-to-srsteps-eq*)**}**
then show *?thesis* **by** (*intro SCR-rrstep-intro*) (*metis srrstep-to-srestep*)+
qed

lemma *monadic-WCR*:
assumes *sig*: *funas-rel* $\mathcal{R}$ ⊆ $\mathcal{F}$ *monadic* $\mathcal{F}$
and *llrg*: *llrg* $\mathcal{R}$
and *wcr*: *WCR* (*gsrstep* $\mathcal{F}$ $\mathcal{R}$)
shows *WCR* (*srstep* $\mathcal{F}$ $\mathcal{R}$)
proof −
{fix *s t u* **assume** *ass*: (*s*, *t*) ∈ *sig-step* $\mathcal{F}$ (*rrstep* $\mathcal{R}$) (*s*, *u*) ∈ *srstep* $\mathcal{F}$ $\mathcal{R}$
from *ass* **have** *funas*: *funas-term s* ⊆ $\mathcal{F}$ *funas-term t* ⊆ $\mathcal{F}$ *funas-term u* ⊆ $\mathcal{F}$
by *blast*+
from *srrstep-to-srestep ass* **have** *gr*: *ground t ground u***using** *ass sig*(*2*) *llrg*
by (*metis llrg-monadic-rstep-pres-groundness*)+
let *?σ* = λ -. *t* **have** *grs*: *ground* (*s* · *?σ*) **using** *gr* **by** *auto*
from *this wcr* **have** *w*: (*t*, *u*) ∈ (*gsrstep* $\mathcal{F}$ $\mathcal{R}$)$^\downarrow$ **using** *gr ass*(*2*) *funas*
using *srstep-subst-closed*[*OF srrstep-to-srestep*[*OF ass*(*1*)], *of ?σ*]
using *srstep-subst-closed*[*OF ass*(*2*), *of ?σ*]
unfolding *WCR-on-def*
by (*metis IntI SigmaI UNIV-I ground-subst-apply mem-Collect-eq*)
then have (*t*, *u*) ∈ (*srstep* $\mathcal{F}$ $\mathcal{R}$)$^\downarrow$ **unfolding** *join-def*
by (*metis gsrsteps-eq-to-srsteps-eq joinD joinI join-def*)**}**
then show *?thesis* **by** (*intro WCR-rrstep-intro*) *simp*
qed

lemma *monadic-UNF*:
assumes *llrg* $\mathcal{R}$ *monadic* $\mathcal{F}$
and *unf*: *UNF* (*gsrstep* $\mathcal{F}$ $\mathcal{R}$)
shows *UNF* (*srstep* $\mathcal{F}$ $\mathcal{R}$)
proof −
{fix *s t* **assume** *ass*: (*s*, *t*) ∈ *comp-rrstep-rel′* $\mathcal{F}$ ($\mathcal{R}^{-1}$) $\mathcal{R}$
then obtain *u* **where** *steps*: (*s*, *u*) ∈ (*srstep* $\mathcal{F}$ ($\mathcal{R}^{-1}$))$^+$ (*u*, *t*) ∈ (*srstep* $\mathcal{F}$

$\mathcal{R})^+$
by (*auto simp*: *sig-step-converse-rstep dest*: *srsteps-with-root-step-srstepsD*)
have *gr*: *ground s ground t* **using** *steps assms*(*1*, *2*)
unfolding *rew-converse-outwards*
by (*auto simp*: *llrg-srsteps-with-root-step-ground llrg-monadic-rsteps-groundness*)
from *steps*(*1*) **have** *f*: *funas-term s* $\subseteq \mathcal{F}$ **using** *srstepsD* **by** *blast*
let *?σ* = λ -. *s*
from *steps gr* **have** (*s*, *u* · *?σ*) $\in$ (*gsrstep* $\mathcal{F}$ ($\mathcal{R}^{-1}$))$^+$ (*u* · *?σ*, *t*) $\in$ (*gsrstep* $\mathcal{F}$ $\mathcal{R})^+$
unfolding *rew-converse-outwards*
using *srsteps-subst-closed*[**where** *?σ* = *?σ* **and** *?s* = *u*, *of* - $\mathcal{F}$] *f*
by (*force simp*: *ground-subst-apply intro*: *ground-srsteps-gsrsteps*)+
then have *UN-redp* $\mathcal{F}$ $\mathcal{R}$ *s t* **using** *unf ground-NF-srstep-gsrstep*[*OF gr*(*1*), *of* $\mathcal{F}$ $\mathcal{R}$]
using *ground-NF-srstep-gsrstep*[*OF gr*(*2*), *of* $\mathcal{F}$ $\mathcal{R}$]
by (*auto simp*: *UNF-on-def UN-redp-def normalizability-def rew-converse-outwards*)
(*meson trancl-into-rtrancl*)**}**
then show *?thesis* **by** (*intro UNF-rrstep-intro*) *simp*
qed

lemma *monadic-UNC*:
assumes *llrg* $\mathcal{R}$ *monadic* $\mathcal{F}$
and *well*: *funas-rel* $\mathcal{R}$ $\subseteq \mathcal{F}$
and *unc*: *UNC* (*gsrstep* $\mathcal{F}$ $\mathcal{R}$)
shows *UNC* (*srstep* $\mathcal{F}$ $\mathcal{R}$)
proof −
{fix *s t* **assume** *ass*: (*s*, *t*) $\in$ (*srstep* $\mathcal{F}$ $\mathcal{R}$)$^{\leftrightarrow *}$ *s* $\in$ *NF* (*srstep* $\mathcal{F}$ $\mathcal{R}$) *t* $\in$ *NF* (*srstep* $\mathcal{F}$ $\mathcal{R}$)
then have *s* = *t*
proof (*cases s* = *t*)
case *False*
then have $\exists$ *s′ t′*. (*s*, *t*) $\in$ (*srstep* $\mathcal{F}$ $\mathcal{R}$)$^{\leftrightarrow *}$ $\wedge$ (*s′*, *s*) $\in$ (*srstep* $\mathcal{F}$ $\mathcal{R}$) $\wedge$ (*t′*, *t*) $\in$ (*srstep* $\mathcal{F}$ $\mathcal{R}$)
using *ass* **by** (*auto simp*: *conversion-def rtrancl-eq-or-trancl dest*: *tranclD tranclD2*)
then obtain *s′ t′* **where** *split*: (*s*, *t*) $\in$ (*srstep* $\mathcal{F}$ $\mathcal{R}$)$^{\leftrightarrow *}$ (*s′*, *s*) $\in$ (*srstep* $\mathcal{F}$ $\mathcal{R}$) (*t′*, *t*) $\in$ (*srstep* $\mathcal{F}$ $\mathcal{R}$) **by** *blast*
from *split*(*2*, *3*) **have** *gr*: *ground s ground t* **using** *llrg-monadic-rstep-pres-groundness*[*OF assms*(*1*, *2*)]
by *blast*+
from *ground-srsteps-eq-gsrsteps-eq*[*OF this ass*(*1*)[*unfolded sig-step-conversion-dist*]]
have (*s*, *t*) $\in$ (*gsrstep* $\mathcal{F}$ $\mathcal{R}$)$^{\leftrightarrow *}$
unfolding *conversion-def Restr-smycl-dist*
by (*simp add*: *rstep-converse-dist srstep-symcl-dist*)
then show *?thesis* **using** *ass*(*2*−) *unc gr*
by (*auto simp*: *UNC-def ground-NF-srstep-gsrstep*)
qed *auto***}**
then show *?thesis* **by** (*auto simp*: *UNC-def UN-redp-def*)

qed

lemma *monadic-NFP*:
assumes *llrg* $\mathcal{R}$ *monadic* $\mathcal{F}$
and *nfp*: *NFP* (*gsrstep* $\mathcal{F}$ $\mathcal{R}$)
shows *NFP* (*srstep* $\mathcal{F}$ $\mathcal{R}$)
proof −
{fix *s t u* **assume** *ass*: $(s, t) \in (srstep\ \mathcal{F}\ \mathcal{R})^*$ $(s, u) \in (srstep\ \mathcal{F}\ \mathcal{R})^*$ $u \in NF$ (*srstep* $\mathcal{F}$ $\mathcal{R}$)
have $(t, u) \in (srstep\ \mathcal{F}\ \mathcal{R})^*$
proof (*cases* $s = u \vee s = t$)
case [*simp*]: *True* **show** *?thesis* **using** *ass*
by (*metis NF-not-suc True rtrancl.rtrancl-refl*)
next
case *False*
then have *steps*:$(s, t) \in (srstep\ \mathcal{F}\ \mathcal{R})^+$ $(s, u) \in (srstep\ \mathcal{F}\ \mathcal{R})^+$ **using** *ass*(*1*, *2*)
by (*auto simp add*: *rtrancl-eq-or-trancl*)
then have *gr*: *ground t ground u* **using** *assms*(*1*, *2*) *llrg-monadic-rsteps-groundness*
by *blast*+
let $?\sigma = \lambda$ -. *t* **from** *steps gr* **have** $(s \cdot ?\sigma, t) \in (gsrstep\ \mathcal{F}\ \mathcal{R})^+$ $(s \cdot ?\sigma, u) \in (gsrstep\ \mathcal{F}\ \mathcal{R})^+$
by (*auto intro*!: *ground-srsteps-gsrsteps*)
(*metis ground-subst-apply srstepsD srsteps-subst-closed*)+
then have $(t, u) \in (gsrstep\ \mathcal{F}\ \mathcal{R})^*$ **using** *nfp ass*(*3*) *gr*
by (*auto simp*: *NFP-on-def*) (*metis ground-NF-srstep-gsrstep normalizability-I trancl-into-rtrancl*)
then show *?thesis* **by** (*auto dest*: *gsrsteps-eq-to-srsteps-eq*)
qed}
then show *?thesis* **unfolding** *NFP-on-def*
by *auto*
qed

end

7 Reducing Rewrite Properties to Properties on Ground Terms over Ground Systems

theory *Ground-Reduction-on-GTRS*
imports
Rewriting-Properties
Rewriting-GTRS
Rewriting-LLRG-LV-Mondaic
begin

lemma *ground-sys-nf-eq-lift*:

```
  fixes ℛ :: ('f, 'v) term rel
  assumes gtrs: ground-sys ℛ ground-sys 𝒮
    and nf: NF (gsrstep ℱ ℛ) = NF (gsrstep ℱ 𝒮)
  shows NF (srstep ℱ ℛ) = NF (srstep ℱ 𝒮)
proof −
  {fix s 𝒰 𝒱 assume ass: ground-sys (𝒰 :: ('f, 'v) term rel) ground-sys 𝒱
    NF (gsrstep ℱ 𝒰) = NF (gsrstep ℱ 𝒱) s ∈ NF (srstep ℱ 𝒰)
    have s ∈ NF (srstep ℱ 𝒱)
    proof (rule ccontr)
      assume s ∉ NF (srstep ℱ 𝒱)
      then obtain C l r σ where step: (l, r) ∈ 𝒱 and rep: s = C⟨l · σ⟩
        and funas: funas-ctxt C ⊆ ℱ funas-term l ⊆ ℱ funas-term r ⊆ ℱ using
ass(2)
        by (auto simp: funas-term-subst NF-def sig-step-def dest!: rstep-imp-C-s-r)
blast
      from step ass(2) rep have rep: s = C⟨l⟩ ground l ground r
        by (auto intro: ground-subst-apply)
      from step rep(2−) funas have l ∉ NF (gsrstep ℱ 𝒱)
        by (auto simp: NF-def sig-step-def Image-def)
      from this ass(3) have l ∉ NF (srstep ℱ 𝒰) by auto
      then obtain t where (l, t) ∈ srstep ℱ 𝒰 by auto
      from srstep-ctxt-closed[OF funas(1) this, unfolded rep(1)[symmetric]]
      show False using ass(4)
        by auto
    qed}
    then show ?thesis using assms
      by (smt (verit, best) equalityI subsetI)
qed

lemma ground-sys-inv:
 ground-sys ℛ ⟹ ground-sys (ℛ^-1) by auto

lemma ground-sys-symcl:
 ground-sys ℛ ⟹ ground-sys (ℛ^↔) by auto

lemma ground-sys-comp-rrstep-rel'-ground:
  assumes ground-sys ℛ ground-sys 𝒮
   and (s, t) ∈ comp-rrstep-rel' ℱ ℛ 𝒮
 shows ground s ground t
proof −
  from assms(3) consider (a) (s, t) ∈ srsteps-with-root-step ℱ ℛ O (srstep ℱ
𝒮)^+ |
    (b) (s, t) ∈ (srstep ℱ ℛ)^+ O srsteps-with-root-step ℱ 𝒮
    by auto
  then have ground s ∧ ground t
  proof cases
    case a
    then show ?thesis using srsteps-with-root-step-ground(1)[OF assms(1)]
    using srsteps-with-root-step-ground(2)[OF assms(1), THEN srsteps-pres-ground-l[OF
```

assms(*2*)]]
by *blast*
next
case *b*
then show *?thesis* **using** *srsteps-with-root-step-ground*(*2*)[*OF assms*(*2*)]
using *srsteps-with-root-step-ground*(*1*)[*OF assms*(*2*), *THEN srsteps-pres-ground-r*[*OF assms*(*1*)]]
by *blast*
qed
then show *ground s ground t* **by** *simp-all*
qed

lemma *GTRS-commute*:
assumes *ground-sys* $\mathcal{R}$ *ground-sys* $\mathcal{S}$
and *com*: *commute* (*gsrstep* $\mathcal{F}$ $\mathcal{R}$) (*gsrstep* $\mathcal{F}$ $\mathcal{S}$)
shows *commute* (*srstep* $\mathcal{F}$ $\mathcal{R}$) (*srstep* $\mathcal{F}$ $\mathcal{S}$)
proof −
{fix *s t* **assume** *ass*: (*s*, *t*) ∈ *comp-rrstep-rel′* $\mathcal{F}$ ($\mathcal{R}^{-1}$) $\mathcal{S}$
then obtain *u* **where** *steps*: (*s*, *u*) ∈ (*srstep* $\mathcal{F}$ ($\mathcal{R}^{-1}$))$^+$ (*u*, *t*) ∈ (*srstep* $\mathcal{F}$ $\mathcal{S}$)$^+$
by (*auto simp*: *sig-step-converse-rstep dest*: *srsteps-with-root-step-srstepsD*)
have *gr*: *ground s ground t ground u*
using *ground-sys-comp-rrstep-rel′-ground*[*OF ground-sys-inv*[*OF assms*(*1*)] *assms*(*2*) *ass*]
using *srsteps-pres-ground-r*[*OF assms*(*2*) - *steps*(*2*)] **by** *auto*
then have (*s*, *u*) ∈ (*gsrstep* $\mathcal{F}$ ($\mathcal{R}^{-1}$))* (*u*, *t*) ∈ (*gsrstep* $\mathcal{F}$ $\mathcal{S}$)* **using** *steps*
by (*auto dest*!: *trancl-into-rtrancl intro*: *ground-srsteps-eq-gsrsteps-eq*)
then have (*s*, *t*) ∈ (*gsrstep* $\mathcal{F}$ $\mathcal{S}$)* *O* (*gsrstep* $\mathcal{F}$ ($\mathcal{R}^{-1}$))* **using** *com steps*
by (*auto simp*: *commute-def rew-converse-inwards*)
from *gsrsteps-eq-relcomp-srsteps-relcompD*[*OF this*]
have *commute-redp* $\mathcal{F}$ $\mathcal{R}$ $\mathcal{S}$ *s t* **unfolding** *commute-redp-def*
by (*simp add*: *rew-converse-inwards*)**}**
then show *?thesis* **by** (*intro commute-rrstep-intro*) *simp*
qed

lemma *GTRS-CR*:
assumes *ground-sys* $\mathcal{R}$
and *CR* (*gsrstep* $\mathcal{F}$ $\mathcal{R}$)
shows *CR* (*srstep* $\mathcal{F}$ $\mathcal{R}$) **using** *GTRS-commute assms*
unfolding *CR-iff-self-commute*
by *blast*

lemma *GTRS-SCR*:
assumes *gtrs*: *ground-sys* $\mathcal{R}$
and *scr*: *SCR* (*gsrstep* $\mathcal{F}$ $\mathcal{R}$)
shows *SCR* (*srstep* $\mathcal{F}$ $\mathcal{R}$)
proof −
{fix *s t u* **assume** *ass*: (*s*, *t*) ∈ *srstep* $\mathcal{F}$ $\mathcal{R}$ (*s*, *u*) ∈ *srstep* $\mathcal{F}$ $\mathcal{R}$

```
    and root: (s, t) ∈ sig-step F (rrstep R) ∨ (s, u) ∈ sig-step F (rrstep R)
    from ass have funas: funas-term s ⊆ F funas-term t ⊆ F funas-term u ⊆ F
by blast+
    from root have gr: ground s ground t ground u using ass gtrs
      using srrstep-ground srstep-pres-ground-l srstep-pres-ground-r
      by metis+
    from scr obtain v where v: (t, v) ∈ (gsrstep F R)^= ∧ (u, v) ∈ (gsrstep F
R)^*
      using gr unfolding SCR-on-def
      by (metis Int-iff UNIV-I ass mem-Collect-eq mem-Sigma-iff)
    then have SCRp F R t u
    by (metis (full-types) Int-iff Un-iff gsrsteps-eq-to-srsteps-eq)}
  then show ?thesis by (intro SCR-rrstep-intro) (metis srrstep-to-srestep)+
qed

lemma GTRS-WCR:
  assumes gtrs: ground-sys R
   and wcr: WCR (gsrstep F R)
  shows WCR (srstep F R)
proof −
  {fix s t u assume ass: (s, t) ∈ sig-step F (rrstep R) (s, u) ∈ srstep F R
    from ass have funas: funas-term s ⊆ F funas-term t ⊆ F funas-term u ⊆ F
by blast+
    from srrstep-ground[OF gtrs ass(1)] have gr: ground s ground t ground u
      using srstep-pres-ground-l[OF gtrs - ass(2)]
      by simp-all
    from this wcr have w: (t, u) ∈ (gsrstep F R)^↓ using ass funas
      unfolding WCR-on-def
      by (metis IntI SigmaI UNIV-I mem-Collect-eq srrstep-to-srestep)
    then have (t, u) ∈ (srstep F R)^↓ unfolding join-def
      by (metis (full-types) gsrsteps-eq-to-srsteps-eq joinD joinI join-def)}
  then show ?thesis by (intro WCR-rrstep-intro) simp
qed

lemma GTRS-UNF:
  assumes gtrs: ground-sys R
   and unf: UNF (gsrstep F R)
  shows UNF (srstep F R)
proof −
  {fix s t assume ass: (s, t) ∈ comp-rrstep-rel' F (R^-1) R
    then obtain u where steps: (s, u) ∈ (srstep F (R^-1))^+ (u, t) ∈ (srstep F
R)^+
      by (auto simp: sig-step-converse-rstep dest: srsteps-with-root-step-srstepsD)
    have gr: ground s ground t ground u
     using ground-sys-comp-rrstep-rel'-ground[OF ground-sys-inv[OF gtrs] gtrs ass]
      using srsteps-pres-ground-r[OF gtrs  - steps(2)] by auto
    from steps(1) have f: funas-term s ⊆ F by (simp add: srstepsD)
    let ?σ = λ -. s
    from steps gr have (s, u · ?σ) ∈ (gsrstep F (R^-1))^+ (u · ?σ, t) ∈ (gsrstep F
```

$\mathcal{R})^+$
unfolding *srstep-converse-dist Restr-converse trancl-converse*
using *srsteps-subst-closed*[**where** *?σ = ?σ* **and** *?s = u*, *of - F*] *f*
by (*force simp*: *ground-subst-apply intro*: *ground-srsteps-gsrsteps*)+
then have *UN-redp F R s t* **using** *unf ground-NF-srstep-gsrstep*[*OF gr*(*1*), *of F R*]
using *ground-NF-srstep-gsrstep*[*OF gr*(*2*), *of F R*]
by (*auto simp*: *UNF-on-def UN-redp-def normalizability-def rew-converse-outwards*)
(*meson trancl-into-rtrancl*)**}**
then show *?thesis* **by** (*intro UNF-rrstep-intro*) *simp*
qed

lemma *GTRS-UNC*:
assumes *gtrs*: *ground-sys R*
and *unc*: *UNC* (*gsrstep F R*)
shows *UNC* (*srstep F R*)
proof −
{fix *s t* **assume** *ass*: (*s*, *t*) ∈ *srsteps-with-root-step F* ($\mathcal{R}^{\leftrightarrow}$)
from *ass* **have** *funas*: *funas-term s* ⊆ *F funas-term t* ⊆ *F*
by (*meson srstepsD srsteps-with-root-step-srstepsD*)+
from *ass* **have** *ground s ground t* **using** *srsteps-with-root-step-ground*[*OF ground-sys-symcl*[*OF gtrs*]]
by *auto*
then have *UN-redp F R s t* **unfolding** *UN-redp-def* **using** *ass unc* **unfolding** *UNC-def*
by (*simp add*: *ground-NF-srstep-gsrstep ground-srsteps-eq-gsrsteps-eq gsrstep-conversion-dist srsteps-with-root-step-sresteps-eqD*)**}**
then show *?thesis* **by** (*intro UNC-rrstep-intro*) *simp*
qed

lemma *GTRS-NFP*:
assumes *ground-sys R*
and *nfp*: *NFP* (*gsrstep F R*)
shows *NFP* (*srstep F R*)
proof −
{fix *s t* **assume** *ass*: (*s*, *t*) ∈ *comp-rrstep-rel′ F* ($\mathcal{R}^{-1}$) *R*
from *ass* **have** *funas*: *funas-term s* ⊆ *F funas-term t* ⊆ *F*
by (*meson Un-iff relcompEpair srstepsD srsteps-with-root-step-srstepsD*)+
from *ass* **have** *ground s ground t*
by (*metis assms*(*1*) *ground-sys-comp-rrstep-rel′-ground ground-sys-inv*)+
from *this ass* **have** (*s*, *t*) ∈ (*gsrstep F* ($\mathcal{R}^{-1}$))* *O* (*gsrstep F R*)*
by (*intro srsteps-eq-relcomp-gsrsteps-relcomp*) (*auto dest*!: *srsteps-with-root-step-sresteps-eqD*)
then have *t* ∈ *NF* (*gsrstep F R*) ⟹ (*s*, *t*) ∈ (*gsrstep F R*)* **using** *NFP-stepD*[*OF nfp*]
by (*auto simp*: *rew-converse-outwards*)
then have *NFP-redp F R s t* **unfolding** *NFP-redp-def*
by (*simp add*: ‹*ground t*› *ground-NF-srstep-gsrstep gsrsteps-eq-to-srsteps-eq*)**}**

then show *?thesis* **by** (*intro NFP-rrstep-intro*) *simp*
qed

lemma *GTRS-NE-aux*:
assumes $(s, t) \in$ *srsteps-with-root-step* $\mathcal{F}$ $\mathcal{R}$
and *gtrs*: *ground-sys* $\mathcal{R}$ *ground-sys* $\mathcal{S}$
and *ne*: *NE* (*gsrstep* $\mathcal{F}$ $\mathcal{R}$) (*gsrstep* $\mathcal{F}$ $\mathcal{S}$)
shows *NE-redp* $\mathcal{F}$ $\mathcal{R}$ $\mathcal{S}$ *s t*
proof −
from *assms*(*1*) **have** *gr*: *ground s ground t*
using *srsteps-with-root-step-ground*[*OF gtrs*(*1*)] **by** *simp-all*
have $(s, t) \in$ (*gsrstep* $\mathcal{F}$ $\mathcal{R}$)$^+$ **using** *gr assms*(*1*)
by (*auto dest*: *srsteps-with-root-step-srstepsD intro*: *ground-srsteps-gsrsteps*)
then have *t* ∈ *NF* (*srstep* $\mathcal{F}$ $\mathcal{R}$) $\Longrightarrow$ $(s, t) \in$ (*gsrstep* $\mathcal{F}$ $\mathcal{S}$)*
using *gr ne* **unfolding** *NE-on-def*
by (*auto simp*: *normalizability-def ground-subst-apply dest*!: *trancl-into-rtrancl*)
blast
then show *NE-redp* $\mathcal{F}$ $\mathcal{R}$ $\mathcal{S}$ *s t* **unfolding** *NE-redp-def*
by (*simp add*: *gsrsteps-eq-to-srsteps-eq*)
qed

lemma *GTRS-NE*:
assumes *gtrs*: *ground-sys* $\mathcal{R}$ *ground-sys* $\mathcal{S}$
and *ne*: *NE* (*gsrstep* $\mathcal{F}$ $\mathcal{R}$) (*gsrstep* $\mathcal{F}$ $\mathcal{S}$)
shows *NE* (*srstep* $\mathcal{F}$ $\mathcal{R}$) (*srstep* $\mathcal{F}$ $\mathcal{S}$)
proof −
{fix *s t* **assume** $(s, t) \in$ *srsteps-with-root-step* $\mathcal{F}$ $\mathcal{R}$
from *GTRS-NE-aux*[*OF this gtrs ne*] **have** *NE-redp* $\mathcal{F}$ $\mathcal{R}$ $\mathcal{S}$ *s t*
by *simp***}**
moreover
{fix *s t* **assume** $(s, t) \in$ *srsteps-with-root-step* $\mathcal{F}$ $\mathcal{S}$
from *GTRS-NE-aux*[*OF this gtrs*(*2*, *1*) *NE-symmetric*[*OF ne*]]
have *NE-redp* $\mathcal{F}$ $\mathcal{S}$ $\mathcal{R}$ *s t* **by** *simp***}**
ultimately show *?thesis*
using *ground-sys-nf-eq-lift*[*OF gtrs NE-NF-eq*[*OF ne*]]
by (*intro NE-rrstep-intro*) *auto*
qed

lemma *gtrs-CE-aux*:
assumes *step*:$(s, t) \in$ *srsteps-with-root-step* $\mathcal{F}$ $(\mathcal{R}^{\leftrightarrow})$
and *gtrs*: *ground-sys* $\mathcal{R}$ *ground-sys* $\mathcal{S}$
and *ce*: *CE* (*gsrstep* $\mathcal{F}$ $\mathcal{R}$) (*gsrstep* $\mathcal{F}$ $\mathcal{S}$)
shows $(s, t) \in$ (*srstep* $\mathcal{F}$ $\mathcal{S}$)$^{\leftrightarrow *}$
proof −
from *step gtrs*(*1*) **have** *ground s ground t*
by (*metis ground-sys-symcl srsteps-with-root-step-ground*)+
then have $(s, t) \in$ (*gsrstep* $\mathcal{F}$ $\mathcal{R}$)$^{\leftrightarrow *}$ **using** *step*

by (*simp add*: *ground-srsteps-eq-gsrsteps-eq gsrstep-conversion-dist srsteps-with-root-step-sresteps-eqD*)
then have $(s, t) \in (gsrstep\ \mathcal{F}\ \mathcal{S})^{\leftrightarrow *}$ **using** *ce* **unfolding** *CE-on-def*
by *blast*
then show $(s, t) \in (srstep\ \mathcal{F}\ \mathcal{S})^{\leftrightarrow *}$
by (*simp add*: *gsrstep-conversion-dist gsrsteps-eq-to-srsteps-eq symcl-srsteps-conversion*)
qed

lemma *gtrs-CE*:
assumes *gtrs*: *ground-sys* $\mathcal{R}$ *ground-sys* $\mathcal{S}$
and *ce*: *CE* (*gsrstep* $\mathcal{F}$ $\mathcal{R}$) (*gsrstep* $\mathcal{F}$ $\mathcal{S}$)
shows *CE* (*srstep* $\mathcal{F}$ $\mathcal{R}$) (*srstep* $\mathcal{F}$ $\mathcal{S}$)
proof −
{fix *s t* **assume** $(s, t) \in srsteps\text{-}with\text{-}root\text{-}step\ \mathcal{F}\ (\mathcal{R}^{\leftrightarrow})$
from *gtrs-CE-aux*[*OF this gtrs ce*] **have** $(s, t) \in (srstep\ \mathcal{F}\ \mathcal{S})^{\leftrightarrow *}$ **by** *simp***}**
moreover
{fix *s t* **assume** $(s, t) \in srsteps\text{-}with\text{-}root\text{-}step\ \mathcal{F}\ (\mathcal{S}^{\leftrightarrow})$
from *gtrs-CE-aux*[*OF this gtrs*(*2*, *1*) *CE-symmetric*[*OF ce*]]
have $(s, t) \in (srstep\ \mathcal{F}\ \mathcal{R})^{\leftrightarrow *}$ **by** *simp***}**
ultimately show *?thesis*
by (*intro CE-rrstep-intro*) *auto*
qed

end

8 Reducing Rewrite Properties to Properties on Ground Terms over Linear Variable-Separated Systems

theory *Ground-Reduction-on-LV*
imports
Rewriting-Properties
Rewriting-LLRG-LV-Mondaic
begin

lemma *lv-linear-sys*: $lv\ \mathcal{R} \Longrightarrow linear\text{-}sys\ \mathcal{R}$
by (*auto simp*: *lv-def*)

lemma *comp-rrstep-rel′-sig-mono*:
$\mathcal{F} \subseteq \mathcal{G} \Longrightarrow comp\text{-}rrstep\text{-}rel'\ \mathcal{F}\ \mathcal{R}\ \mathcal{S} \subseteq comp\text{-}rrstep\text{-}rel'\ \mathcal{G}\ \mathcal{R}\ \mathcal{S}$
by (*meson Un-mono relcomp-mono srsteps-monp srsteps-with-root-step-sig-mono*)

lemma *srsteps-eqD*: $(s, t) \in (srstep\ \mathcal{F}\ \mathcal{R})^* \Longrightarrow (s, t) \in (rstep\ \mathcal{R})^*$
by (*metis rtrancl-eq-or-trancl srstepsD*)

9 Linear-variable separated results

locale *open-terms-two-const-lv* =
  fixes $\mathcal{R}$ :: ('*f*, '*v*) *term rel* **and** $\mathcal{F}$ *c d*
  assumes *lv*: *lv* $\mathcal{R}$ **and** *sig*: *funas-rel* $\mathcal{R} \subseteq \mathcal{F}$
    and *fresh*: $(c,\ 0) \notin \mathcal{F}$ $(d,\ 0) \notin \mathcal{F}$
    and *diff*: $c \neq d$
begin

abbreviation $\mathcal{H} \equiv$ *insert* $(c,\ 0)$ (*insert* $(d,0)$ $\mathcal{F}$)
abbreviation $\sigma_c \equiv$ *const-subst c*
abbreviation $\sigma_d \equiv$ *const-subst d*

lemma *sig-mono*: $\mathcal{F} \subseteq \mathcal{H}$ **by** *auto*
lemma *fresh-sym-c*: $(c,\ 0) \notin$ *funas-rel* $\mathcal{R}$ **using** *sig fresh*
  by (*auto simp*: *funas-rel-def*)
lemma *fresh-sym-d*: $(d,\ 0) \notin$ *funas-rel* $\mathcal{R}$ **using** *sig fresh*
  by (*auto simp*: *funas-rel-def*)
lemma *fresh-sym-c-inv*: $(c,\ 0) \notin$ *funas-rel* $(\mathcal{R}^{-1})$ **using** *sig fresh*
  by (*auto simp*: *funas-rel-def*)
lemma *fresh-sym-d-inv*: $(d,\ 0) \notin$ *funas-rel* $(\mathcal{R}^{-1})$ **using** *sig fresh*
  by (*auto simp*: *funas-rel-def*)

lemmas *all-fresh* = *fresh-sym-c fresh-sym-d fresh-sym-c-inv fresh-sym-d-inv*

lemma *sig-inv*: *funas-rel* $(\mathcal{R}^{-1}) \subseteq \mathcal{F}$ **using** *sig* **unfolding** *funas-rel-def* **by** *auto*
lemma *lv-inv*: *lv* $(\mathcal{R}^{-1})$ **using** *lv* **unfolding** *lv-def* **by** *auto*
lemma *well-subst*:
  $\bigwedge x.$ *funas-term* ((*const-subst c*) *x*) $\subseteq \mathcal{H}$
  $\bigwedge x.$ *funas-term* ((*const-subst d*) *x*) $\subseteq \mathcal{H}$
  by *auto*

lemma *srsteps-with-root-step-to-grsteps*:
  assumes $(s,\ t) \in$ *srsteps-with-root-step* $\mathcal{F}$ $\mathcal{R}$
  shows $(s \cdot \sigma_c,\ t \cdot \sigma_d) \in (\mathit{gsrstep}\ \mathcal{H}\ \mathcal{R})^*$
proof −
  from *assms* **have** *lift*: $(s,\ t) \in$ *srsteps-with-root-step* $\mathcal{H}$ $\mathcal{R}$
    using *srsteps-with-root-step-sig-mono*[*OF sig-mono*]
    by *blast*
 note [*dest*!] = *lv-srsteps-with-root-step-idep-subst*[*OF lv - well-subst*, *THEN srsteps-with-root-step-sresteps-eq*1
  have $(s \cdot \sigma_c,\ t \cdot \sigma_d) \in (\mathit{srstep}\ \mathcal{H}\ \mathcal{R})^*$ **using** *lift*
    using *srsteps-eq-subst-closed*[*OF - well-subst*(*1*)]
    using *srsteps-eq-subst-closed*[*OF - well-subst*(*2*)]
    by (*auto dest*: *trancl-into-rtrancl*)
  then show *?thesis*
    by (*intro ground-srsteps-eq-gsrsteps-eq*) *auto*
qed

lemma *comp-rrstep-rel′-to-grsteps*:
assumes $(s,\ t) \in$ *comp-rrstep-rel′* $\mathcal{F}$ $(\mathcal{R}^{-1})$ $\mathcal{R}$
shows $(s \cdot \sigma_c,\ t \cdot \sigma_d) \in$ $(gsrstep\ \mathcal{H}\ (\mathcal{R}^{-1}))^*$ O $(gsrstep\ \mathcal{H}\ \mathcal{R})^*$
proof −
from *assms* **have** *lift*: $(s,\ t) \in$ *comp-rrstep-rel′* $\mathcal{H}$ $(\mathcal{R}^{-1})$ $\mathcal{R}$ **using** *sig-mono*
by (*meson in-mono relcomp-mono srsteps-monp srsteps-with-root-step-sig-mono sup-mono*)
note [*dest*!] = *lv-srsteps-with-root-step-idep-subst*[*OF lv - well-subst*, *THEN srsteps-with-root-step-sresteps-eq1*
note [*dest*!] = *lv-srsteps-with-root-step-idep-subst*[*OF lv-inv - well-subst*, *THEN srsteps-with-root-step-sresteps-eqD*]
have $(s \cdot \sigma_c,\ t \cdot \sigma_d) \in$ $(srstep\ \mathcal{H}\ (\mathcal{R}^{-1}))^*$ O $(srstep\ \mathcal{H}\ \mathcal{R})^*$ **using** *lift*
using *srsteps-eq-subst-closed*[*OF - well-subst*(*1*)]
using *srsteps-eq-subst-closed*[*OF - well-subst*(*2*)]
by (*auto dest*!: *trancl-into-rtrancl*) *blast*
then show *?thesis*
by (*intro srsteps-eq-relcomp-gsrsteps-relcomp*) *auto*
qed

lemma *gsrsteps-eq-to-srsteps*:
assumes $(s \cdot \sigma_c,\ t \cdot \sigma_d) \in$ $(gsrstep\ \mathcal{H}\ \mathcal{R})^*$
and *funas*: *funas-term* $s \subseteq \mathcal{F}$ *funas-term* $t \subseteq \mathcal{F}$
shows $(s,\ t) \in$ $(srstep\ \mathcal{F}\ \mathcal{R})^*$
proof −
from *funas* **have** *d*: $(d,\ 0) \notin$ *funas-term* s **and** *c*: $(c,\ 0) \notin$ *funas-term* $(t \cdot \sigma_d)$
using *fresh diff* **by** (*auto simp*: *funas-term-subst*)
have $(s,\ t) \in$ $(rstep\ \mathcal{R})^*$ **using** *c gsrsteps-eq-to-rsteps-eq*[*OF assms*(*1*)]
using *remove-const-subst-steps-eq-lhs*[*OF lv-linear-sys*[*OF lv*] *fresh-sym-c*,
THEN remove-const-subst-steps-eq-rhs[*OF lv-linear-sys*[*OF lv*] *fresh-sym-d d*]]
by *auto*
then show *?thesis* **using** *funas sig* **by** *blast*
qed

lemma *convert-NF-to-GNF*:
funas-term $t \subseteq \mathcal{F} \Longrightarrow t \in NF\ (srstep\ \mathcal{F}\ \mathcal{R}) \Longrightarrow t \cdot \sigma_c \in NF\ (gsrstep\ \mathcal{H}\ \mathcal{R})$
funas-term $t \subseteq \mathcal{F} \Longrightarrow t \in NF\ (srstep\ \mathcal{F}\ \mathcal{R}) \Longrightarrow t \cdot \sigma_d \in NF\ (gsrstep\ \mathcal{H}\ \mathcal{R})$
using *NF-to-fresh-const-subst-NF*[*OF lv-linear-sys*[*OF lv*] *fresh-sym-c sig*]
using *NF-to-fresh-const-subst-NF*[*OF lv-linear-sys*[*OF lv*] *fresh-sym-d sig*]
by *blast+*

lemma *convert-GNF-to-NF*:
funas-term $t \subseteq \mathcal{F} \Longrightarrow t \cdot \sigma_c \in NF\ (gsrstep\ \mathcal{H}\ \mathcal{R}) \Longrightarrow t \in NF\ (srstep\ \mathcal{F}\ \mathcal{R})$
funas-term $t \subseteq \mathcal{F} \Longrightarrow t \cdot \sigma_d \in NF\ (gsrstep\ \mathcal{H}\ \mathcal{R}) \Longrightarrow t \in NF\ (srstep\ \mathcal{F}\ \mathcal{R})$
using *fresh-const-subst-NF-pres*[*OF fresh-sym-c sig*]
using *fresh-const-subst-NF-pres*[*OF fresh-sym-d sig*]
using *sig-mono* **by** *blast+*

lemma *lv-CR*:
assumes *cr*: *CR* (*gsrstep* $\mathcal{H}$ $\mathcal{R}$)
shows *CR* (*srstep* $\mathcal{F}$ $\mathcal{R}$)
proof −
{fix *s t* **assume** *ass*: (*s*, *t*) ∈ (*srstep* $\mathcal{F}$ ($\mathcal{R}^{-1}$))$^+$ *O srsteps-with-root-step* $\mathcal{F}$ $\mathcal{R}$
from *ass* **have** *funas*: *funas-term s* ⊆ $\mathcal{F}$ *funas-term t* ⊆ $\mathcal{F}$
by (*metis Pair-inject ass relcompE srstepsD srsteps-with-root-step-srstepsD*)+
have (*s* · σ_c, *t* · σ_d) ∈ (*gsrstep* $\mathcal{H}$ ($\mathcal{R}^{-1}$))* *O* (*gsrstep* $\mathcal{H}$ $\mathcal{R}$)*
using *ass comp-rrstep-rel′-to-grsteps* **by** *auto*
then have *s*: (*s* · σ_c, *t* · σ_d) ∈ (*gsrstep* $\mathcal{H}$ $\mathcal{R}$)* *O* (*gsrstep* $\mathcal{H}$ ($\mathcal{R}^{-1}$))*
using *cr* **unfolding** *CR-on-def*
by (*auto simp*: *join-def rew-converse-outwards*)
from *gsrsteps-eq-relcomp-to-rsteps-relcomp*[*OF this*]
have *commute-redp* $\mathcal{F}$ $\mathcal{R}$ $\mathcal{R}$ *s t* **unfolding** *commute-redp-def* **using** *funas fresh*
using *remove-const-subst-relcomp*[*OF lv-linear-sys*[*OF lv*] *lv-linear-sys*[*OF lv-inv*]
all-fresh diff, *THEN rsteps-eq-relcomp-srsteps-eq-relcompI*[*OF sig sig-inv funas*]]
by (*metis srstep-converse-dist subsetD*)**}**
then show *?thesis* **by** (*intro CR-rrstep-intro*) *simp*
qed

lemma *lv-WCR*:
assumes *wcr*: *WCR* (*gsrstep* $\mathcal{H}$ $\mathcal{R}$)
shows *WCR* (*srstep* $\mathcal{F}$ $\mathcal{R}$)
proof −
note *sig-trans-root* = *subsetD*[*OF srrstep-monp*[*OF sig-mono*]]
note *sig-trans* = *subsetD*[*OF srstep-monp*[*OF sig-mono*]]
{fix *s t u* **assume** *ass*: (*s*, *t*) ∈ *sig-step* $\mathcal{F}$ (*rrstep* $\mathcal{R}$) (*s*, *u*) ∈ *srstep* $\mathcal{F}$ $\mathcal{R}$
from *ass* **have** *funas*: *funas-term s* ⊆ $\mathcal{F}$ *funas-term t* ⊆ $\mathcal{F}$ *funas-term u* ⊆ $\mathcal{F}$ **by** *blast*+
then have *free*: (*d*, *0*) ∉ *funas-term u* (*c*, *0*) ∉ *funas-term t* **using** *fresh* **by** *blast*+
have ∗: *ground* (*s* · σ_c) *ground* (*t* · σ_d) *ground* (*u* · σ_c) **by** *auto*
moreover have (*s* · σ_c, *t* · σ_d) ∈ *srstep* $\mathcal{H}$ $\mathcal{R}$ **using** *lv sig-trans-root*[*OF ass*(*1*)]
by (*intro lv-root-step-idep-subst*[*THEN srrstep-to-srestep*]) *auto*
moreover have (*s* · σ_c, *u* · σ_c) ∈ *srstep* $\mathcal{H}$ $\mathcal{R}$
using *srstep-subst-closed*[*OF sig-trans*[*OF ass*(*2*)], *of* σ_c]
by *auto*
ultimately have *w*: (*t* · σ_d, *u* · σ_c) ∈ (*gsrstep* $\mathcal{H}$ $\mathcal{R}$)$^\downarrow$ **using** *wcr* **unfolding** *WCR-on-def*
by *auto* (*metis* (*no-types*, *lifting*) ∗)
note *join-unfolded* = *w*[*unfolded join-def rew-converse-inwards*]
have (*t*, *u*) ∈ (*srstep* $\mathcal{F}$ $\mathcal{R}$)$^\downarrow$ **unfolding** *join-def*
using *remove-const-subst-relcomp*[*OF lv-linear-sys*[*OF lv*] *lv-linear-sys*[*OF lv-inv*]
fresh-sym-d fresh-sym-c fresh-sym-d-inv fresh-sym-c-inv diff[*symmetric*] *free*
gsrsteps-eq-relcomp-to-rsteps-relcomp[*OF join-unfolded*],

THEN rsteps-eq-relcomp-srsteps-eq-relcompI[*OF sig sig-inv funas*(*2*−)]]
by (*metis* (*no-types*, *lifting*) *srstep-converse-dist*)**}**
then show *?thesis* **by** (*intro WCR-rrstep-intro*) *simp*
qed

lemma *lv-NFP*:
assumes *nfp*: *NFP* (*gsrstep* $\mathcal{H}$ $\mathcal{R}$)
shows *NFP* (*srstep* $\mathcal{F}$ $\mathcal{R}$)
proof −
{fix *s t* **assume** *ass*: (*s*, *t*) ∈ *comp-rrstep-rel′* $\mathcal{F}$ ($\mathcal{R}^{-1}$) $\mathcal{R}$
from *ass* **have** *funas*: *funas-term s* ⊆ $\mathcal{F}$ *funas-term t* ⊆ $\mathcal{F}$
by (*metis Un-iff ass relcompEpair srstepsD srsteps-with-root-step-srstepsD*)+
from *comp-rrstep-rel′-to-grsteps*[*OF ass*]
have (*s* · σ_c, *t* · σ_d) ∈ (*gsrstep* $\mathcal{H}$ ($\mathcal{R}^{-1}$))* *O* (*gsrstep* $\mathcal{H}$ $\mathcal{R}$)* **by** *simp*
then have *t* · σ_d ∈ *NF* (*gsrstep* $\mathcal{H}$ $\mathcal{R}$) ⟹ (*s* · σ_c, *t* · σ_d) ∈ (*gsrstep* $\mathcal{H}$ $\mathcal{R}$)*
using *NFP-stepD*[*OF nfp*]
by (*auto simp*: *rew-converse-outwards*)
from *gsrsteps-eq-to-srsteps*[*OF this funas*]
have *NFP-redp* $\mathcal{F}$ $\mathcal{R}$ *s t* **unfolding** *NFP-redp-def*
using *convert-NF-to-GNF*(*2*)[*OF funas*(*2*)]
by *simp***}**
then show *?thesis* **by** (*intro NFP-rrstep-intro*) *simp*
qed

lemma *lv-UNF*:
assumes *unf*: *UNF* (*gsrstep* $\mathcal{H}$ $\mathcal{R}$)
shows *UNF* (*srstep* $\mathcal{F}$ $\mathcal{R}$)
proof −
{fix *s t* **assume** *ass*: (*s*, *t*) ∈ *comp-rrstep-rel′* $\mathcal{F}$ ($\mathcal{R}^{-1}$) $\mathcal{R}$
from *ass* **have** *funas*: *funas-term s* ⊆ $\mathcal{F}$ *funas-term t* ⊆ $\mathcal{F}$
by (*metis Un-iff ass relcompEpair srstepsD srsteps-with-root-step-srstepsD*)+
from *comp-rrstep-rel′-to-grsteps*[*OF ass*]
have (*s* · σ_c, *t* · σ_d) ∈ (*gsrstep* $\mathcal{H}$ ($\mathcal{R}^{-1}$))* *O* (*gsrstep* $\mathcal{H}$ $\mathcal{R}$)* **by** *simp*
then have *s* · σ_c ∈ *NF* (*gsrstep* $\mathcal{H}$ $\mathcal{R}$) ⟹ *t* · σ_d ∈ *NF* (*gsrstep* $\mathcal{H}$ $\mathcal{R}$) ⟹ *s* · σ_c = *t* · σ_d
using *unf* **unfolding** *UNF-on-def*
by (*auto simp*: *normalizability-def rew-converse-outwards*)
then have *UN-redp* $\mathcal{F}$ $\mathcal{R}$ *s t* **unfolding** *UN-redp-def*
using *convert-NF-to-GNF*(*1*)[*OF funas*(*1*)]
using *convert-NF-to-GNF*(*2*)[*OF funas*(*2*)]
by (*metis NF-not-suc funas gsrsteps-eq-to-srsteps rtrancl-eq-or-trancl*)**}**
then show *?thesis* **by** (*intro UNF-rrstep-intro*) *simp*
qed

lemma *lv-UNC*:
assumes *unc*: *UNC* (*gsrstep* $\mathcal{H}$ $\mathcal{R}$)
shows *UNC* (*srstep* $\mathcal{F}$ $\mathcal{R}$)
proof −
have *lv-conv*: *lv* ($\mathcal{R}^{\leftrightarrow}$) **using** *lv* **by** (*auto simp*: *lv-def*)

{fix s t **assume** *ass*: $(s,\ t) \in$ *srsteps-with-root-step* $\mathcal{F}$ $(\mathcal{R}^{\leftrightarrow})$
from *ass* **have** *funas*: *funas-term* $s \subseteq \mathcal{F}$ *funas-term* $t \subseteq \mathcal{F}$
by (*metis Un-iff ass relcompEpair srstepsD srsteps-with-root-step-srstepsD*)+
have $(s \cdot \sigma_c,\ t \cdot \sigma_d) \in (\mathit{gsrstep}\ \mathcal{H}\ \mathcal{R})^{\leftrightarrow *}$
using *lv-srsteps-with-root-step-idep-subst*[*OF lv-conv srsteps-with-root-step-sig-mono*[*OF sig-mono*, *THEN subsetD*, *OF ass*]
well-subst, *THEN srsteps-with-root-step-sresteps-eqD*]
unfolding *conversion-def Restr-smycl-dist srstep-symcl-dist*
by (*intro ground-srsteps-eq-gsrsteps-eq*) *auto*
then have $s \cdot \sigma_c \in$ *NF* $(\mathit{gsrstep}\ \mathcal{H}\ \mathcal{R}) \Longrightarrow t \cdot \sigma_d \in$ *NF* $(\mathit{gsrstep}\ \mathcal{H}\ \mathcal{R}) \Longrightarrow s \cdot \sigma_c = t \cdot \sigma_d$
using *unc* **unfolding** *UNC-def*
by (*auto simp*: *normalizability-def sig-step-converse-rstep rtrancl-converse Restr-converse*)
then have *UN-redp* $\mathcal{F}$ $\mathcal{R}$ s t **unfolding** *UN-redp-def*
using *convert-NF-to-GNF*(*1*)[*OF funas*(*1*)]
using *convert-NF-to-GNF*(*2*)[*OF funas*(*2*)]
by (*metis NF-not-suc funas gsrsteps-eq-to-srsteps rtrancl-eq-or-trancl*)**}**
then show *?thesis* **by** (*intro UNC-rrstep-intro*) *simp*
qed

lemma *lv-SCR*:
assumes *scr*: *SCR* $(\mathit{gsrstep}\ \mathcal{H}\ \mathcal{R})$
shows *SCR* $(\mathit{srstep}\ \mathcal{F}\ \mathcal{R})$
proof −
note *sig-trans-root* = *subsetD*[*OF srrstep-monp*[*OF sig-mono*]]
note *sig-trans* = *subsetD*[*OF srstep-monp*[*OF sig-mono*]]
note *cl-on-c* = *lin-fresh-rstep-const-replace-closed*[*OF lv-linear-sys*[*OF lv*] *fresh-sym-c*]
lin-fresh-rstep-const-replace-closed[*OF lv-linear-sys*[*OF lv-inv*] *fresh-sym-c-inv*]
note *cl-on-d* = *lin-fresh-rstep-const-replace-closed*[*OF lv-linear-sys*[*OF lv*] *fresh-sym-d*]
lin-fresh-rstep-const-replace-closed[*OF lv-linear-sys*[*OF lv-inv*] *fresh-sym-d-inv*]
{fix s t u **assume** *ass*: $(s,\ t) \in \mathit{srstep}\ \mathcal{F}\ \mathcal{R}\ (s,\ u) \in \mathit{srstep}\ \mathcal{F}\ \mathcal{R}$
and *root*: $(s,\ t) \in \mathit{sig\text{-}step}\ \mathcal{F}\ (\mathit{rrstep}\ \mathcal{R}) \vee (s,\ u) \in \mathit{sig\text{-}step}\ \mathcal{F}\ (\mathit{rrstep}\ \mathcal{R})$
from *ass* **have** *funas*: *funas-term* $s \subseteq \mathcal{F}$ *funas-term* $t \subseteq \mathcal{F}$ *funas-term* $u \subseteq \mathcal{F}$
by (*force dest*: *sig-stepE*)+
then have *fresh-c-t*:$(c,\ 0) \notin$ *funas-term* $(t \cdot \sigma_d)$ **and** *fresh-d-u*:$(d,\ 0) \notin$ *funas-term* u **using** *fresh diff*
by (*auto simp*: *funas-term-subst*)
have $*$: $(s,\ t) \in \mathit{sig\text{-}step}\ \mathcal{F}\ (\mathit{rrstep}\ \mathcal{R}) \Longrightarrow (s \cdot \sigma_c,\ t \cdot \sigma_d) \in \mathit{gsrstep}\ \mathcal{H}\ \mathcal{R} \wedge (s \cdot \sigma_c,\ u \cdot \sigma_c) \in \mathit{gsrstep}\ \mathcal{H}\ \mathcal{R}$
$(s,\ u) \in \mathit{sig\text{-}step}\ \mathcal{F}\ (\mathit{rrstep}\ \mathcal{R}) \Longrightarrow (s \cdot \sigma_d,\ t \cdot \sigma_d) \in \mathit{gsrstep}\ \mathcal{H}\ \mathcal{R} \wedge (s \cdot \sigma_d,\ u \cdot \sigma_c) \in \mathit{gsrstep}\ \mathcal{H}\ \mathcal{R}$
using *srstep-subst-closed*[*OF sig-trans*[*OF ass*(*2*)] *well-subst*(*1*)]
using *srstep-subst-closed*[*OF sig-trans*[*OF ass*(*2*)] *well-subst*(*2*)]
using *lv-root-step-idep-subst*[*OF lv*] *sig-trans-root*[*of* $(s,\ t)$ $\mathcal{R}$] *sig-trans-root*[*of* $(s,\ u)$ $\mathcal{R}$]
by (*simp-all add*: *ass*(*1*) *sig-trans srstep-subst-closed srrstep-to-srestep*)
from *this scr* **have** $(u \cdot \sigma_c,\ t \cdot \sigma_d) \in (\mathit{gsrstep}\ \mathcal{H}\ \mathcal{R})^*\ O\ ((\mathit{gsrstep}\ \mathcal{H}\ \mathcal{R})^{=})^{-1}$

using *root* **unfolding** *SCR-on-def* **by** (*meson UNIV-I converse-iff rel-comp.simps*)
then have *v*: $(u \cdot \sigma_c,\ t \cdot \sigma_d) \in (srstep\ \mathcal{H}\ \mathcal{R})^*\ O\ ((srstep\ \mathcal{H}\ \mathcal{R})^=)^{-1}$
by (*auto dest*: *gsrsteps-eq-to-srsteps-eq*)
have [*simp*]: *funas-term v* $\subseteq \mathcal{F} \Longrightarrow$ *term-to-sig* $\mathcal{F}$ *x v* = *v* **for** *x v*
by (*simp add*: *subset-insertI2*)
have $(u,\ t) \in (srstep\ \mathcal{F}\ \mathcal{R})^*\ O\ ((srstep\ \mathcal{F}\ \mathcal{R})^=)^{-1}$
proof −
from *v* **have** $(u,\ t \cdot \sigma_d) \in (rstep\ \mathcal{R})^*\ O\ (rstep\ (\mathcal{R}^{-1}))^=$
using *const-replace-closed-relcomp*[*THEN const-replace-closed-remove-subst-lhs*, *OF*
const-replace-closed-rtrancl[*OF cl-on-c*(*1*)]
const-replace-closed-symcl[*OF cl-on-c*(*2*)] *fresh-c-t*, *of u*]
by (*auto simp*: *relcomp.relcompI srstepD simp flip*: *rstep-converse-dist dest*: *srsteps-eqD*)
then have $(u,\ t) \in (rstep\ \mathcal{R})^*\ O\ (rstep\ (\mathcal{R}^{-1}))^=$
using *const-replace-closed-relcomp*[*THEN const-replace-closed-remove-subst-lhs*, *OF*
const-replace-closed-symcl[*OF cl-on-d*(*1*)]
const-replace-closed-rtrancl[*OF cl-on-d*(*2*)] *fresh-d-u*, *of t*]
using *converse-relcomp*[**where** *?s* = $(rstep\ (\mathcal{R}^{-1}))^=$ **and** *?r* = $(rstep\ \mathcal{R})^*$]
by (*metis* (*no-types*, *lifting*) *converseD converse-Id converse-Un converse-converse rstep-converse-dist rtrancl-converse*)
then obtain *v* **where** $(u,\ v) \in (rstep\ \mathcal{R})^*$ $(v,\ t) \in (rstep\ (\mathcal{R}^{-1}))^=$ **by** *auto*
then have $(u,\ term\text{-}to\text{-}sig\ \mathcal{F}\ x\ v) \in (srstep\ \mathcal{F}\ \mathcal{R})^*$ $(term\text{-}to\text{-}sig\ \mathcal{F}\ x\ v,\ t) \in (srstep\ \mathcal{F}\ (\mathcal{R}^{-1}))^=$
using *funas*(*2*, *3*) *sig fresh*
by (*auto simp*: *rtrancl-eq-or-trancl rstep-trancl-sig-step-r subset-insertI2 rew-converse-outwards dest*: *fuans-term-term-to-sig*[*THEN subsetD*] *intro*!: *rstep-term-to-sig-r rstep-srstepI rsteps-eq-srsteps-eqI*)
then show *?thesis*
by (*metis converse-Id converse-Un relcomp.relcompI srstep-converse-dist*)
qed
then have *SCRp* $\mathcal{F}$ $\mathcal{R}$ *t u*
by *auto*}
then show *?thesis* **by** (*intro SCR-rrstep-intro*) (*metis srrstep-to-srestep*)+
qed

end

locale *open-terms-two-const-lv-two-sys* =
open-terms-two-const-lv $\mathcal{R}$
for $\mathcal{R}$ :: ('*f*, '*v*) *term rel* +
fixes $\mathcal{S}$ :: ('*f*, '*v*) *term rel*
assumes *lv-S*: *lv* $\mathcal{S}$ **and** *sig-S*: *funas-rel* $\mathcal{S} \subseteq \mathcal{F}$
begin

lemma *fresh-sym-c-S*: $(c,\ 0) \notin$ *funas-rel* $\mathcal{S}$ **using** *sig-S fresh*
by (*auto simp*: *funas-rel-def*)
lemma *fresh-sym-d-S*: $(d,\ 0) \notin$ *funas-rel* $\mathcal{S}$ **using** *sig-S fresh*
by (*auto simp*: *funas-rel-def*)

lemma *lv-commute*:
assumes *com*: *commute* (*gsrstep* $\mathcal{H}$ $\mathcal{R}$) (*gsrstep* $\mathcal{H}$ $\mathcal{S}$)
shows *commute* (*srstep* $\mathcal{F}$ $\mathcal{R}$) (*srstep* $\mathcal{F}$ $\mathcal{S}$)
proof −
have *linear*: *linear-sys* $\mathcal{S}$ *linear-sys* $(\mathcal{R}^{-1})$ **using** *lv lv-S* **unfolding** *lv-def* **by** *auto*
{fix *s t* **assume** *ass*: $(s,\ t) \in$ *comp-rrstep-rel'* $\mathcal{F}$ $(\mathcal{R}^{-1})$ $\mathcal{S}$
from *ass* **have** *funas*: *funas-term* $s \subseteq \mathcal{F}$ *funas-term* $t \subseteq \mathcal{F}$
by (*metis Un-iff ass relcompEpair srstepsD srsteps-with-root-step-srstepsD*)+
have $(s \cdot \sigma_c,\ t \cdot \sigma_d) \in$ (*gsrstep* $\mathcal{H}$ $(\mathcal{R}^{-1})$)* *O* (*gsrstep* $\mathcal{H}$ $\mathcal{S}$)*
using *comp-rrstep-rel'-sig-mono*[*OF sig-mono*, *THEN subsetD*, *OF ass*]
using *srsteps-eq-subst-closed*[*OF - well-subst*(*1*)] *srsteps-eq-subst-closed*[*OF - well-subst*(*2*)]
by (*auto simp*: *rew-converse-inwards dest*!: *trancl-into-rtrancl*
lv-srsteps-with-root-step-idep-subst[*OF lv-inv - well-subst*, *THEN srsteps-with-root-step-sresteps-eqD*]
lv-srsteps-with-root-step-idep-subst[*OF lv-S - well-subst*, *THEN srsteps-with-root-step-sresteps-eqD*]
intro!: *srsteps-eq-relcomp-gsrsteps-relcomp*) *blast*
then have *w*: $(s \cdot \sigma_c,\ t \cdot \sigma_d) \in$ (*gsrstep* $\mathcal{H}$ $\mathcal{S}$)* *O* (*gsrstep* $\mathcal{H}$ $(\mathcal{R}^{-1})$)* **using** *com*
unfolding *commute-def Restr-converse srstep-converse-dist*
by *blast*
have $(s,\ t) \in$ (*srstep* $\mathcal{F}$ $\mathcal{S}$)* *O* (*srstep* $\mathcal{F}$ $(\mathcal{R}^{-1})$)* **using** *funas sig-S sig-inv fresh*
using *remove-const-subst-relcomp*[*OF linear fresh-sym-c-S fresh-sym-d-S fresh-sym-c-inv fresh-sym-d-inv diff - -*
gsrsteps-eq-relcomp-to-rsteps-relcomp[*OF w*]]
by (*intro rsteps-eq-relcomp-srsteps-eq-relcompI*) *auto*
then have *commute-redp* $\mathcal{F}$ $\mathcal{R}$ $\mathcal{S}$ *s t* **unfolding** *commute-redp-def*
by (*simp add*: *rew-converse-inwards*)**}**
then show *?thesis* **by** (*intro commute-rrstep-intro*) *simp*
qed

lemma *lv-NE*:
assumes *ne*: *NE* (*gsrstep* $\mathcal{H}$ $\mathcal{R}$) (*gsrstep* $\mathcal{H}$ $\mathcal{S}$)
shows *NE* (*srstep* $\mathcal{F}$ $\mathcal{R}$) (*srstep* $\mathcal{F}$ $\mathcal{S}$)
proof −
from *NE-NF-eq*[*OF ne*] **have** *ne-eq*: *NF* (*srstep* $\mathcal{F}$ $\mathcal{R}$) = *NF* (*srstep* $\mathcal{F}$ $\mathcal{S}$)
using *lv lv-S sig sig-S fresh-sym-c fresh-sym-c-S*
by (*intro linear-sys-gNF-eq-NF-eq*) (*auto dest*: *lv-linear-sys*)
{fix *s t* **assume** *step*: $(s,\ t) \in$ *srsteps-with-root-step* $\mathcal{F}$ $\mathcal{R}$
then have *funas*: *funas-term* $s \subseteq \mathcal{F}$ *funas-term* $t \subseteq \mathcal{F}$
by (*metis Un-iff step relcompEpair srstepsD srsteps-with-root-step-srstepsD*)+

then have *fresh*: $(c,\ 0) \notin$ *funas-term* t $(d,\ 0) \notin$ *funas-term* s **using** *fresh* **by** *auto*
from *srsteps-with-root-step-sig-mono*[*OF sig-mono*, *THEN subsetD*, *OF step*]
have $(s \cdot \sigma_c,\ t \cdot \sigma_d) \in (gsrstep\ \mathcal{H}\ \mathcal{R})^*$
using *lv-srsteps-with-root-step-idep-subst*[*OF lv - well-subst*, *THEN srsteps-with-root-step-sresteps-eqD*]
by (*intro ground-srsteps-eq-gsrsteps-eq*) *auto*
then have $t \cdot \sigma_d \in NF\ (gsrstep\ \mathcal{H}\ \mathcal{S}) \Longrightarrow (s \cdot \sigma_c,\ t \cdot \sigma_d) \in (gsrstep\ \mathcal{H}\ \mathcal{S})^*$
using *ne NE-NF-eq*[*OF ne*] **unfolding** *NE-on-def*
by (*auto simp*: *normalizability-def*)
from *this*[*THEN gsrsteps-eq-to-rsteps-eq*, *THEN remove-const-subst-steps*[*OF lv-linear-sys*[*OF lv-S*] *fresh-sym-c-S fresh-sym-d-S diff fresh*]]
have *NE-redp* $\mathcal{F}\ \mathcal{R}\ \mathcal{S}\ s\ t$ **using** *NF-to-fresh-const-subst-NF*[*OF lv-linear-sys*[*OF lv-S*] *fresh-sym-d-S sig-S funas*(*2*)]
using *funas sig-S* **unfolding** *NE-redp-def ne-eq*
by (*auto intro*: *rsteps-eq-srsteps-eqI*)**}**
moreover
{fix $s\ t$ **assume** *step*: $(s,\ t) \in$ *srsteps-with-root-step* $\mathcal{F}\ \mathcal{S}$
then have *funas*: *funas-term* $s \subseteq \mathcal{F}$ *funas-term* $t \subseteq \mathcal{F}$
by (*metis sig-S sig-stepE sig-step-rsteps-dist srsteps-with-root-step-srstepsD*)+
then have *fresh*: $(c,\ 0) \notin$ *funas-term* t $(d,\ 0) \notin$ *funas-term* s **using** *fresh* **by** *auto*
from *srsteps-with-root-step-sig-mono*[*OF sig-mono*, *THEN subsetD*, *OF step*]
have $(s \cdot \sigma_c,\ t \cdot \sigma_d) \in (gsrstep\ \mathcal{H}\ \mathcal{S})^*$
using *lv-srsteps-with-root-step-idep-subst*[*OF lv-S - well-subst*, *THEN srsteps-with-root-step-sresteps-eqD*]
by (*intro ground-srsteps-eq-gsrsteps-eq*) *auto*
then have $t \cdot \sigma_d \in NF\ (gsrstep\ \mathcal{H}\ \mathcal{R}) \Longrightarrow (s \cdot \sigma_c,\ t \cdot \sigma_d) \in (gsrstep\ \mathcal{H}\ \mathcal{R})^*$
using *ne NE-NF-eq*[*OF ne*] **unfolding** *NE-on-def*
by (*auto simp*: *normalizability-def*)
from *this*[*THEN gsrsteps-eq-to-rsteps-eq*, *THEN remove-const-subst-steps*[*OF lv-linear-sys*[*OF lv*] *fresh-sym-c fresh-sym-d diff fresh*]]
have *NE-redp* $\mathcal{F}\ \mathcal{S}\ \mathcal{R}\ s\ t$ **using** *NF-to-fresh-const-subst-NF*[*OF lv-linear-sys*[*OF lv*] *fresh-sym-d sig funas*(*2*), *of* $\mathcal{H}$]
using *funas sig* **unfolding** *NE-redp-def ne-eq*
by (*auto intro*: *rsteps-eq-srsteps-eqI*)**}**
ultimately show *?thesis* **using** *ne-eq*
by (*intro NE-rrstep-intro*) *auto*
qed

end

— CE is special as it only needs one additional constant therefore not included in the locale
lemma *lv-CE-aux*:
assumes $(s,\ t) \in$ *srsteps-with-root-step* $\mathcal{F}$ $(\mathcal{R}^{\leftrightarrow})$
and *sig*: *funas-rel* $\mathcal{R} \subseteq \mathcal{F}$ *funas-rel* $\mathcal{S} \subseteq \mathcal{F}$
and *fresh*: $(c,\ 0) \notin \mathcal{F}$ **and** *const*: $(a,\ 0) \in \mathcal{F}$
and *lv*: *lv* $\mathcal{R}$ *lv* $\mathcal{S}$
and *ce*: *CE* (*gsrstep* (*insert* $(c,\ 0)$ $\mathcal{F}$) $\mathcal{R}$) (*gsrstep* (*insert* $(c,\ 0)$ $\mathcal{F}$) $\mathcal{S}$)
shows $(s,\ t) \in (srstep\ \mathcal{F}\ \mathcal{S})^{\leftrightarrow *}$

proof −
let *?*$\mathcal{H}$ = *insert* (*c*, *0*) $\mathcal{F}$ **have** *mono*: $\mathcal{F} \subseteq$ *?*$\mathcal{H}$ **by** *auto*
have *lv-conv*: *lv* ($\mathcal{R}^{\leftrightarrow}$) *lv* ($\mathcal{S}^{\leftrightarrow}$) **using** *lv* **by** (*auto simp*: *lv-def*)
have *sig-conv*: *funas-rel* ($\mathcal{S}^{\leftrightarrow}$) $\subseteq \mathcal{F}$ **using** *sig*(*2*) **by** (*auto simp*: *funas-rel-def*)
from *fresh* **have** *fresh-sys*: (*c*, *0*) $\notin$ *funas-rel* $\mathcal{R}$ (*c*, *0*) $\notin$ *funas-rel* $\mathcal{S}$ (*c*, *0*) $\notin$ *funas-rel* ($\mathcal{S}^{\leftrightarrow}$)
using *sig* **by** (*auto simp*: *funas-rel-def*)
from *assms*(*1*) **have** *lift*: (*s*, *t*) $\in$ *srsteps-with-root-step* *?*$\mathcal{H}$ ($\mathcal{R}^{\leftrightarrow}$)
unfolding *srsteps-with-root-step-def*
by (*meson in-mono mono relcomp-mono rtrancl-mono srrstep-monp srstep-monp*)
from *assms*(*1*) **have** *funas*: *funas-term s* $\subseteq \mathcal{F}$ *funas-term t* $\subseteq \mathcal{F}$
by (*meson srstepsD srsteps-with-root-step-srstepsD*)+
from *srsteps-with-root-step-sresteps-eqD*[*OF assms*(*1*), *THEN subsetD*[*OF srsteps-eq-monp*[*OF mono*]]]
have (*s* · *const-subst c*, *t* · *const-subst c*) $\in$ (*srstep* *?*$\mathcal{H}$ $\mathcal{R}$)$^{\leftrightarrow *}$
by (*auto simp*: *sig-step-conversion-dist intro*: *srsteps-eq-subst-closed*)
moreover have (*s* · *const-subst c*, *t* · *const-subst a*) $\in$ (*srstep* *?*$\mathcal{H}$ $\mathcal{R}$)$^{\leftrightarrow *}$
unfolding *sig-step-conversion-dist* **using** *const*
by (*intro lv-srsteps-with-root-step-idep-subst*[*OF lv-conv*(*1*) *lift*, *THEN srsteps-with-root-step-sresteps-eqD*]) *auto*
moreover have *ground* (*s* · *const-subst c*) *ground* (*t* · *const-subst a*) *ground* (*t* · *const-subst a*)
by *auto*
ultimately have *toS*:(*s* · *const-subst c*, *t* · *const-subst c*) $\in$ (*gsrstep* *?*$\mathcal{H}$ $\mathcal{S}$)$^{\leftrightarrow *}$
(*s* · *const-subst c*, *t* · *const-subst a*) $\in$ (*gsrstep* *?*$\mathcal{H}$ $\mathcal{S}$)$^{\leftrightarrow *}$
using *ground-srsteps-eq-gsrsteps-eq*[**where** *?*$\mathcal{F}$ = *?*$\mathcal{H}$ **and** *?*$\mathcal{R}$ = $\mathcal{R}^{\leftrightarrow}$]
using *ce* **unfolding** *CE-on-def*
by (*auto simp*: *Restr-smycl-dist conversion-def srstep-symcl-dist*)
then have ∗: (*t* · *const-subst a*, *t* · *const-subst c*) $\in$ (*gsrstep* *?*$\mathcal{H}$ $\mathcal{S}$)$^{\leftrightarrow *}$
by (*metis* (*no-types*, *lifting*) *conversion-inv conversion-rtrancl rtrancl.rtrancl-into-rtrancl*)
have (*t* · *const-subst a*, *t*) $\in$ (*srstep* $\mathcal{F}$ $\mathcal{S}$)$^{\leftrightarrow *}$ **using** *const*
using *funas*(*2*) *fresh*
using *remove-const-subst-steps-eq-rhs*[*OF lv-linear-sys*[*OF lv-conv*(*2*)] *fresh-sys*(*3*)
-
gsrsteps-eq-to-rsteps-eq[*OF* ∗[*unfolded gsrstep-conversion-dist*]]]
by (*cases vars-term t* = {})
(*auto simp*: *funas-term-subst sig-step-conversion-dist split*: *if-splits intro*!: *rsteps-eq-srsteps-eqI*[*OF sig-conv*])
moreover have (*s*, *t* · *const-subst a*) $\in$ (*srstep* $\mathcal{F}$ $\mathcal{S}$)$^{\leftrightarrow *}$ **using** *const*
using *funas fresh*
using *remove-const-subst-steps-eq-lhs*[*OF lv-linear-sys*[*OF lv-conv*(*2*)] *fresh-sys*(*3*)
-
gsrsteps-eq-to-rsteps-eq[*OF toS*(*2*)[*unfolded gsrstep-conversion-dist*]]]
by (*cases vars-term t* = {})
(*auto simp*: *sig-step-conversion-dist funas-term-subst split*: *if-splits intro*!: *rsteps-eq-srsteps-eqI*[*OF sig-conv*])
ultimately show (*s*, *t*) $\in$ (*srstep* $\mathcal{F}$ $\mathcal{S}$)$^{\leftrightarrow *}$
by (*meson conversionE conversionI rtrancl-trans*)
qed

lemma *lv-CE*:
assumes *sig*: *funas-rel* $\mathcal{R} \subseteq \mathcal{F}$ *funas-rel* $\mathcal{S} \subseteq \mathcal{F}$
and *fresh*: $(c, 0) \notin \mathcal{F}$ **and** *const*: $(a, 0) \in \mathcal{F}$
and *lv*: *lv* $\mathcal{R}$ *lv* $\mathcal{S}$
and *ce*: *CE* (*gsrstep* (*insert* $(c, 0)$ $\mathcal{F}$) $\mathcal{R}$) (*gsrstep* (*insert* $(c, 0)$ $\mathcal{F}$) $\mathcal{S}$)
shows *CE* (*srstep* $\mathcal{F}$ $\mathcal{R}$) (*srstep* $\mathcal{F}$ $\mathcal{S}$)
proof −
{fix *s t* **assume** $(s, t) \in$ *srsteps-with-root-step* $\mathcal{F}$ $(\mathcal{R}^{\leftrightarrow})$
from *lv-CE-aux*[*OF this assms*] **have** $(s, t) \in (srstep\ \mathcal{F}\ \mathcal{S})^{\leftrightarrow *}$ **by** *simp***}**
moreover
{fix *s t* **assume** $(s, t) \in$ *srsteps-with-root-step* $\mathcal{F}$ $(\mathcal{S}^{\leftrightarrow})$
from *lv-CE-aux*[*OF this sig*(2, 1) *fresh const lv*(2, 1) *CE-symmetric*[*OF ce*]]
have $(s, t) \in (srstep\ \mathcal{F}\ \mathcal{R})^{\leftrightarrow *}$ **by** *simp***}**
ultimately show *?thesis*
by (*intro CE-rrstep-intro*) *auto*
qed

9.1 Specialized for monadic signature

lemma *lv-NE-aux*:
assumes $(s, t) \in$ *srsteps-with-root-step* $\mathcal{F}$ $\mathcal{R}$ **and** *fresh*: $(c, 0) \notin \mathcal{F}$
and *sig*: *funas-rel* $\mathcal{R} \subseteq \mathcal{F}$ *funas-rel* $\mathcal{S} \subseteq \mathcal{F}$
and *lv*: *lv* $\mathcal{R}$ *lv* $\mathcal{S}$
and *mon*: *monadic* $\mathcal{F}$
and *ne*: *NE* (*gsrstep* (*insert* $(c, 0)$ $\mathcal{F}$) $\mathcal{R}$) (*gsrstep* (*insert* $(c, 0)$ $\mathcal{F}$) $\mathcal{S}$)
shows *NE-redp* $\mathcal{F}$ $\mathcal{R}$ $\mathcal{S}$ *s t*
proof −
let *?σ* = *const-subst c* **let** *?H* = *insert* $(c, 0)$ $\mathcal{F}$
have *mono*: $\mathcal{F} \subseteq$ *?H* **by** *auto*
from *mon* **have** *mh*: *monadic ?H* **by** (*auto simp*: *monadic-def*)
from *fresh* **have** *fresh-sys*: $(c, 0) \notin$ *funas-rel* $\mathcal{R}$ $(c, 0) \notin$ *funas-rel* $\mathcal{S}$ **using** *sig*
by (*auto simp*: *funas-rel-def*)
from *assms* **have** *funas*: *funas-term s* $\subseteq \mathcal{F}$ *funas-term t* $\subseteq \mathcal{F}$
by (*meson srstepsD srsteps-with-root-step-srstepsD*)+
from *funas* **have** *fresh-t*: $(c, 0) \notin$ *funas-term t* **using** *fresh* **by** *auto*
from *srsteps-subst-closed*[*OF srsteps-monp*[*OF mono*, *THEN subsetD*, *OF srsteps-with-root-step-srstepsD*[*OF assms*(1)]], *of ?σ*]
have *gstep*: $(s \cdot ?\sigma,\ t \cdot ?\sigma) \in (gsrstep\ (insert\ (c, 0)\ \mathcal{F})\ \mathcal{R})^+$
by (*auto simp*: *ground-subst-apply intro*!: *ground-srsteps-gsrsteps*)
then have *neq*: $t \in NF\ (srstep\ \mathcal{F}\ \mathcal{R}) \Longrightarrow s \cdot ?\sigma \neq t \cdot ?\sigma$
using *NF-to-fresh-const-subst-NF*[*OF lv-linear-sys*[*OF lv*(1)] *fresh-sys*(1) *sig*(1) *funas*(2), *of ?H*]
by (*metis NF-no-trancl-step*)
then have $t \in NF\ (srstep\ \mathcal{F}\ \mathcal{R}) \Longrightarrow (s \cdot ?\sigma,\ t \cdot ?\sigma) \in (gsrstep\ (insert\ (c, 0)\ \mathcal{F})\ \mathcal{S})^+$ **using** *gstep*
using *NF-to-fresh-const-subst-NF*[*OF lv-linear-sys*[*OF lv*(1)] *fresh-sys*(1) *sig*(1) *funas*(2), *of ?H*]

using *NE-NF-eq*[*OF ne*, *symmetric*] *ne* **unfolding** *NE-on-def*
by (*auto simp*: *normalizability-def ground-subst-apply*) (*meson rtrancl-eq-or-trancl*)
then show *?thesis* **unfolding** *NE-redp-def*
using *remove-const-lv-mondaic-steps*[*OF lv*(*2*) *fresh-sys*(*2*) *mh*, *THEN srstepsD*[*THEN conjunct1*],
THEN rsteps-srstepsI[*OF sig*(*2*) *funas*]]
by (*auto dest*!: *gsrsteps-to-srsteps*)
qed

lemma *lv-NE*:
assumes *sig*: *funas-rel* $\mathcal{R} \subseteq \mathcal{F}$ *funas-rel* $\mathcal{S} \subseteq \mathcal{F}$
and *mon*: *monadic* $\mathcal{F}$ **and** *fresh*: $(c, 0) \notin \mathcal{F}$
and *lv*: *lv* $\mathcal{R}$ *lv* $\mathcal{S}$
and *ne*: *NE* (*gsrstep* (*insert* $(c, 0)$ $\mathcal{F}$) $\mathcal{R}$) (*gsrstep* (*insert* $(c, 0)$ $\mathcal{F}$) $\mathcal{S}$)
shows *NE* (*srstep* $\mathcal{F}$ $\mathcal{R}$) (*srstep* $\mathcal{F}$ $\mathcal{S}$)
proof −
from *fresh* **have** *fresh-sys*: $(c, 0) \notin$ *funas-rel* $\mathcal{R}$ $(c, 0) \notin$ *funas-rel* $\mathcal{S}$
using *sig* **by** (*auto simp*: *funas-rel-def*)
from *NE-NF-eq*[*OF ne*] **have** *ne-eq*: *NF* (*srstep* $\mathcal{F}$ $\mathcal{R}$) = *NF* (*srstep* $\mathcal{F}$ $\mathcal{S}$) **using** *lv sig fresh-sys*
by (*intro linear-sys-gNF-eq-NF-eq*) (*auto dest*: *lv-linear-sys*)
{fix *s t* **assume** $(s, t) \in$ *srsteps-with-root-step* $\mathcal{F}$ $\mathcal{R}$
from *lv-NE-aux*[*OF this fresh sig lv mon ne*] **have** *NE-redp* $\mathcal{F}$ $\mathcal{R}$ $\mathcal{S}$ *s t* **by** *simp***}**
moreover
{fix *s t* **assume** *ass*: $(s, t) \in$ *srsteps-with-root-step* $\mathcal{F}$ $\mathcal{S}$
from *lv-NE-aux*[*OF this fresh sig*(*2*, *1*) *lv*(*2*, *1*) *mon NE-symmetric*[*OF ne*]]
have *NE-redp* $\mathcal{F}$ $\mathcal{S}$ $\mathcal{R}$ *s t* **by** *simp***}**
ultimately show *?thesis* **using** *ne-eq*
by (*intro NE-rrstep-intro*) *auto*
qed

end

References

[1] C. Sternagel and R. Thiemann. Abstract rewriting. *Archive of Formal Proofs*, June 2010. https://isa-afp.org/entries/Abstract-Rewriting.html, Formal proof development.