Decreasing-Diagrams

Harald Zankl

February 23, 2021

Abstract

This theory contains a formalization of decreasing diagrams showing that any locally decreasing abstract rewrite system is confluent. We consider the valley (van Oostrom, TCS 1994) and the conversion version (van Oostrom, RTA 2008) and closely follow the original proofs. As an application we prove Newman's lemma.

A description of this formalization is available in [3].

Contents

1 Decreasing Diagrams

theory *Decreasing-Diagrams* **imports** *HOL−Library.Multiset Abstract−Rewriting.Abstract-Rewriting*
begin

1.1 Valley Version

This section follows [1].

1.1.1 Appendix

interaction of multisets with sets

definition *diff* :: *'a multiset* ⇒ *'a set* ⇒ *'a multiset*
where *diff M S* = *filter-mset* (λ*x*. *x* ∉ *S*) *M*

definition *intersect* :: *'a multiset ⇒ 'a set ⇒ 'a multiset*
where *intersect M S = filter-mset (λx. x ∈ S) M*

notation
diff (**infixl** *−s 800*) **and**
intersect (**infixl** *∩s 800*)

lemma *count-diff* [*simp*]:
count (M −s A) a = count M a ∗ of-bool (a ∉ A)
by (*simp add*: *diff-def*)

lemma *set-mset-diff* [*simp*]:
set-mset (M −s A) = set-mset M − A
by (*auto simp add*: *diff-def*)

lemma *diff-eq-singleton-imp*:
M −s A = {#a#} ⟹ a ∈ (set-mset M − A)
unfolding *diff-def filter-mset-eq-conv* **by** *auto*

lemma *count-intersect* [*simp*]:
count (M ∩s A) a = count M a ∗ of-bool (a ∈ A)
by (*simp add*: *intersect-def*)

lemma *set-mset-intersect* [*simp*]:
set-mset (M ∩s A) = set-mset M ∩ A
by (*auto simp add*: *intersect-def*)

lemma *diff-from-empty*: *{#}−s S = {#}* **unfolding** *diff-def* **by** *auto*

lemma *diff-empty*: *M −s {} = M* **unfolding** *diff-def* **by** (*rule multiset-eqI*) *simp*

lemma *submultiset-implies-subset*: **assumes** *M ⊆# N* **shows** *set-mset M ⊆ set-mset N*
using *assms mset-subset-eqD* **by** *auto*

lemma *subset-implies-remove-empty*: **assumes** *set-mset M ⊆ S* **shows** *M −s S = {#}*
unfolding *diff-def* **using** *assms* **by** (*induct M*) *auto*

lemma *remove-empty-implies-subset*: **assumes** *M −s S = {#}* **shows** *set-mset M ⊆ S* **proof**
fix *x* **assume** *A*: *x ∈ set-mset M*
have *x ∉ set-mset (M −s S)* **using** *assms* **by** *auto*
thus *x ∈ S* **using** *A* **unfolding** *diff-def* **by** *auto*
qed

lemma *lemmaA-3-8*: *(M + N) −s S = (M −s S) + (N −s S)* **unfolding** *diff-def* **by** (*rule multiset-eqI*) *simp*

lemma *lemmaA-3-9*: $(M -s S) -s T = M -s (S \cup T)$ **unfolding** *diff-def* **by** (*rule multiset-eqI*) *simp*
lemma *lemmaA-3-10*: $M = (M \cap s S) + (M -s S)$ **unfolding** *diff-def intersect-def* **by** *auto*
lemma *lemmaA-3-11*: $(M -s T) \cap s S = (M \cap s S) -s T$ **unfolding** *diff-def intersect-def* **by** (*rule multiset-eqI*) *simp*

1.1.2 Multisets

Definition 2.5(1)

definition *ds* :: *'a rel* ⇒ *'a set* ⇒ *'a set*
where *ds r S* = {*y* . ∃ *x* ∈ *S*. (*y*,*x*) ∈ *r*}

definition *dm* :: *'a rel* ⇒ *'a multiset* ⇒ *'a set*
where *dm r M* = *ds r* (*set-mset M*)

definition *dl* :: *'a rel* ⇒ *'a list* ⇒ *'a set*
where *dl r σ* = *ds r* (*set σ*)

notation
ds (**infixl** ↓*s 900*) **and**
dm (**infixl** ↓*m 900*) **and**
dl (**infixl** ↓*l 900*)

missing but useful

lemma *ds-ds-subseteq-ds*: **assumes** *t*: *trans r* **shows** *ds r* (*ds r S*) ⊆ *ds r S* **proof**
fix *x* **assume** *A*: *x* ∈ *ds r* (*ds r S*) **show** *x* ∈ *ds r S* **proof** −
from *A* **obtain** *y z* **where** (*x*,*y*) ∈ *r* **and** (*y*,*z*) ∈ *r* **and** *mem*: *z* ∈ *S* **unfolding** *ds-def* **by** *auto*
thus *?thesis* **using** *mem t trans-def* **unfolding** *ds-def* **by** *fast*
qed
qed

from PhD thesis of van Oostrom

lemma *ds-monotone*: **assumes** *S* ⊆ *T* **shows** *ds r S* ⊆ *ds r T* **using** *assms* **unfolding** *ds-def* **by** *auto*

lemma *subset-imp-ds-subset*: **assumes** *trans r* **and** *S* ⊆ *ds r T* **shows** *ds r S* ⊆ *ds r T*
using *assms ds-monotone ds-ds-subseteq-ds* **by** *blast*

Definition 2.5(2)

strict order (mult) is used from Multiset.thy

definition *mult-eq* :: *'a rel* ⇒ *'a multiset rel* **where**
mult-eq r = (*mult1 r*)*

definition *mul* :: *'a rel* ⇒ *'a multiset rel* **where**

mul r = {(*M*,*N*).∃ *I J K*. *M* = *I* + *K* ∧ *N* = *I* + *J* ∧ *set-mset K* ⊆ *dm r J* ∧ *J* ≠ {#}}

definition *mul-eq* :: *'a rel* ⇒ *'a multiset rel* **where**
mul-eq r = {(*M*,*N*).∃ *I J K*. *M* = *I* + *K* ∧ *N* = *I* + *J* ∧ *set-mset K* ⊆ *dm r J*}

lemma *in-mul-eqI*:
assumes *M* = *I* + *K N* = *I* + *J set-mset K* ⊆ *r* ↓*m J*
shows (*M*, *N*) ∈ *mul-eq r*
using *assms* **by** (*auto simp add*: *mul-eq-def*)

lemma *downset-intro*:
assumes ∀ *k*∈*set-mset K*.∃ *j*∈*set-mset J*.(*k*,*j*)∈*r* **shows** *set-mset K* ⊆ *dm r J* **proof**
fix *x* **assume** *x*∈*set-mset K* **thus** *x* ∈ *dm r J* **using** *assms* **unfolding** *dm-def ds-def* **by** *fast*
qed

lemma *downset-elim*:
assumes *set-mset K* ⊆ *dm r J* **shows** ∀ *k*∈*set-mset K*.∃ *j*∈*set-mset J*.(*k*,*j*)∈*r* **proof**
fix *k* **assume** *k*∈ *set-mset K* **thus** ∃ *j*∈*set-mset J*.(*k*,*j*)∈ *r* **using** *assms* **unfolding** *dm-def ds-def* **by** *fast*
qed

to closure-free representation

lemma *mult-eq-implies-one-or-zero-step*:
assumes *trans r* **and** (*M*,*N*) ∈ *mult-eq r* **shows** ∃ *I J K*. *N* = *I* + *J* ∧ *M* = *I* + *K* ∧ *set-mset K* ⊆ *dm r J*
proof (*cases* (*M*,*N*) ∈ *mult r*)
case *True* **thus** *?thesis* **using** *mult-implies-one-step*[*OF assms*(*1*)] *downset-intro* **by** *blast*
next
case *False* **hence** *A*: *M* = *N* **using** *assms rtrancl-eq-or-trancl* **unfolding** *mult-eq-def mult-def* **by** *metis*
hence *N* = *N* + {#} ∧ *M* = *M* + {#} ∧ *set-mset* {#} ⊆ *dm r*{#} **by** *auto*
thus *?thesis* **unfolding** *A* **by** *fast*
qed

from closure-free representation

lemma *one-step-implies-mult-eq*: **assumes** *trans r* **and** *set-mset K* ⊆ *dm r J* **shows** (*I*+*K*,*I*+*J*)∈*mult-eq r*
proof (*cases set-mset J* = {})
case *True* **hence** *set-mset K* = {} **using** *assms downset-elim* **by** (*metis all-not-in-conv emptyE*)
thus *?thesis* **using** *True* **unfolding** *mult-eq-def* **by** *auto*
next
case *False* **hence** *h*:*J* ≠ {#} **using** *set-mset-eq-empty-iff* **by** *auto*
hence (*I*+*K*,*I*+*J*)∈ *mult r* **using** *set-mset-eq-empty-iff assms one-step-implies-mult downset-elim*
by *auto blast*

thus *?thesis* **unfolding** *mult-eq-def mult-def* **by** *auto*
qed

lemma *mult-is-mul*: **assumes** *trans r* **shows** *mult r* = *mul r* **proof**
show *mult r* ⊆ *mul r* **proof**
fix *N M* **assume** *A*: (*N*,*M*) ∈ *mult r* **show** (*N*,*M*) ∈ *mul r* **proof** −
obtain *I J K* **where** *M* = *I* + *J* **and** *N* = *I* + *K* **and** *J* ≠ {#} **and** *set-mset K* ⊆ *dm r J*
using *mult-implies-one-step*[*OF assms A*] *downset-intro* **by** *metis*
thus *?thesis* **unfolding** *mul-def* **by** *auto*
qed
qed
next
show *mul r* ⊆ *mult r* **proof**
fix *N M* **assume** *A*: (*N*,*M*) ∈ *mul r* **show** (*N*,*M*) ∈ *mult r* **proof** −
obtain *I J K* **where** *M* = *I* + *J* **and** *N* = *I* + *K* **and** *J* ≠ {#} **and** *set-mset K* ⊆ *dm r J*
using *A* **unfolding** *mul-def* **by** *auto*
thus *?thesis* **using** *one-step-implies-mult assms downset-elim* **by** *metis*
qed
qed
qed

lemma *mult-eq-is-mul-eq*: **assumes** *trans r* **shows** *mult-eq r* = *mul-eq r* **proof**
show *mult-eq r* ⊆ *mul-eq r* **proof**
fix *N M* **assume** *A*: (*N*,*M*) ∈ *mult-eq r* **show** (*N*,*M*) ∈ *mul-eq r* **proof** (*cases* (*N*,*M*) ∈ *mult r*)
case *True* **thus** *?thesis* **unfolding** *mult-is-mul*[*OF assms*] *mul-def mul-eq-def* **by** *auto*
next
case *False* **hence** *eq*: *N* = *M* **using** *A rtranclD* **unfolding** *mult-def mult-eq-def* **by** *metis*
hence *M* = *M* + {#} ∧ *N* = *N* + {#} ∧ *set-mset* {#} ⊆ *dm r* {#} **by** *auto*
thus *?thesis* **unfolding** *eq* **unfolding** *mul-eq-def* **by** *fast*
qed
qed
show *mul-eq r* ⊆ *mult-eq r* **using** *one-step-implies-mult-eq*[*OF assms*] **unfolding** *mul-eq-def* **by** *auto*
qed

lemma *mul-eq r* = (*mul r*)$^=$ **proof**
show *mul-eq r* ⊆ (*mul r*)$^=$ **proof**
fix *M N* **assume** *A*:(*M*,*N*) ∈ *mul-eq r* **show** (*M*,*N*) ∈ (*mul r*)$^=$ **proof** −
from *A* **obtain** *I J K* **where** *1*: *M* = *I* + *K* **and** *2*: *N* = *I* + *J* **and** *3*: *set-mset K* ⊆ *dm r J* **unfolding** *mul-eq-def* **by** *auto*
show *?thesis* **proof** (*cases J* = {#})
case *True* **hence** *K* = {#} **using** *3* **unfolding** *dm-def ds-def* **by** *auto*
hence *M* = *N* **using** *True 1 2* **by** *auto*
thus *?thesis* **by** *auto*

next
case *False* **thus** *?thesis* **using** *1 2 3* **unfolding** *mul-def mul-eq-def* **by** *auto*
qed
qed
qed
show *mul-eq r* $\supseteq$ (*mul r*)$^=$ **proof**
fix *M N* **assume** *A*:(*M*,*N*) $\in$ (*mul r*)$^=$ **show** (*M*,*N*) $\in$ *mul-eq r*
proof (*cases M* = *N*)
case *True* **hence** *M* = *M* + {#} **and** *N* = *M* + {#} **and** *set-mset* {#} $\subseteq$ *dm r* {#} **by** *auto*
thus *?thesis* **unfolding** *mul-eq-def* **by** *fast*
next
case *False* **hence** (*M*,*N*) $\in$ *mul r* **using** *A* **by** *auto*
thus *?thesis* **unfolding** *mul-def mul-eq-def* **by** *auto*
qed
qed
qed

useful properties on multisets

lemma *mul-eq-reflexive*: (*M*,*M*) $\in$ *mul-eq r* **proof** −
have *M* = *M* + {#} **and** *set-mset* {#} $\subseteq$ *dm r* {#} **by** *auto*
thus *?thesis* **unfolding** *mul-eq-def* **by** *fast*
qed

lemma *mul-eq-trans*: **assumes** *trans r* **and** (*M*,*N*) $\in$ *mul-eq r* **and** (*N*,*P*) $\in$ *mul-eq r* **shows** (*M*,*P*) $\in$ *mul-eq r*
using *assms* **unfolding** *mult-eq-is-mul-eq*[*symmetric*,*OF assms*(*1*)] *mult-eq-def*
by *auto*

lemma *mul-eq-singleton*: **assumes** (*M*, {#α#}) $\in$ *mul-eq r* **shows** *M* = {#α#} $\vee$ *set-mset M* $\subseteq$ *dm r* {#α#} **proof** −
from *assms* **obtain** *I J K* **where** *1*:*M* = *I* + *K* **and** *2*:{#α#} = *I* + *J* **and** *3*:*set-mset K* $\subseteq$ *dm r J* **unfolding** *mul-eq-def* **by** *auto*
thus *?thesis* **proof** (*cases I* = {#})
case *True* **hence** *J* = {#α#} **using** *2* **by** *auto*
thus *?thesis* **using** *1 3 True* **by** *auto*
next
case *False* **hence** *i*: *I* = {#α#} **using** *2 union-is-single* **by** *metis*
hence *J* = {#} **using** *2 union-is-single* **by** *metis*
thus *?thesis* **using** *1 i 3* **unfolding** *dm-def ds-def* **by** *auto*
qed
qed

lemma *mul-and-mul-eq-imp-mul*: **assumes** *trans r* **and** (*M*,*N*) $\in$ *mul r* **and** (*N*,*P*) $\in$ *mul-eq r* **shows** (*M*,*P*) $\in$ *mul r*
using *assms* **unfolding** *mult-is-mul*[*symmetric*,*OF assms*(*1*)] *mult-eq-is-mul-eq*[*symmetric*,*OF assms*(*1*)] *mult-def mult-eq-def* **by** *auto*

lemma *mul-eq-and-mul-imp-mul*: **assumes** *trans r* **and** (*M*,*N*) $\in$ *mul-eq r* **and**

$(N,P) \in mul\ r$ **shows** $(M,P) \in mul\ r$
using *assms* **unfolding** *mult-is-mul*[*symmetric*,*OF assms*(*1*)] *mult-eq-is-mul-eq*[*symmetric*,*OF assms*(*1*)] *mult-def mult-eq-def* **by** *auto*

lemma *wf-mul*: **assumes** *trans r* **and** *wf r* **shows** *wf* (*mul r*)
unfolding *mult-is-mul*[*symmetric*,*OF assms*(*1*)] **using** *wf-mult*[*OF assms*(*2*)] **by** *auto*

lemma *remove-is-empty-imp-mul*: **assumes** $M - s\ dm\ r\ \{\#\alpha\#\} = \{\#\}$ **shows** $(M,\{\#\alpha\#\}) \in mul\ r$ **proof** −
from *assms* **have** *C*: *set-mset M* ⊆ *dm r* {#α#} **by** (*metis remove-empty-implies-subset*)
have $M = \{\#\}+M$ **and** $\{\#\alpha\#\}=\{\#\}+\{\#\alpha\#\}$ **and** $\{\#\alpha\#\} \neq \{\#\}$ **by** *auto*
thus *?thesis* **using** *C* **unfolding** *mul-def* **by** *fast*
qed

Lemma 2.6

lemma *lemma2-6-1-set*: *ds r* (*S* ∪ *T*) = *ds r S* ∪ *ds r T*
unfolding *set-mset-union ds-def* **by** *auto*

lemma *lemma2-6-1-list*: *dl r* (σ@τ) = *dl r* σ ∪ *dl r* τ
unfolding *dl-def ds-def set-append* **by** *auto*

lemma *lemma2-6-1-multiset*: *dm r* (*M* + *N*) = *dm r M* ∪ *dm r N*
unfolding *dm-def set-mset-union ds-def* **by** *auto*

lemma *lemma2-6-1-diff*: (*dm r M*) − *ds r S* ⊆ *dm r* (*M* − *s S*)
unfolding *diff-def dm-def ds-def* **by** (*rule subsetI*) *auto*

missing but useful

lemma *dl-monotone*: *dl r* (σ@τ) ⊆ *dl r* (σ@τ′@τ) **unfolding** *lemma2-6-1-list* **by** *auto*

Lemma 2.6.2

lemma *lemma2-6-2-a*: **assumes** *t*: *trans r* **and** *M* ⊆# *N* **shows** (*M*,*N*) ∈ *mul-eq r* **proof** −
from *assms*(*2*) **obtain** *J* **where** *N*=*M*+*J* **by** (*metis assms*(*2*) *mset-subset-eq-exists-conv*)
hence $M = M + \{\#\}$ **and** $N = M + J$ **and** *set-mset* {#} ⊆ *dm r J* **by** *auto*
thus *?thesis* **unfolding** *mul-eq-def* **by** *fast*
qed

lemma *mul-eq-not-equal-imp-elt*:
assumes (*M*,*N*)∈*mul-eq r* **and** *y*∈*set-mset M* − *set-mset N* **shows** ∃ *z*∈*set-mset N*.(*y*,*z*)∈*r* **proof** −
from *assms* **obtain** *I J K* **where** *N*=*I*+*J* **and** *M*=*I*+*K* **and** *F3*:*set-mset K* ⊆ *dm r J* **unfolding** *mul-eq-def* **by** *auto*
thus *?thesis* **using** *assms*(*2*) *downset-elim*[*OF F3*] **by** *auto*
qed

lemma *lemma2-6-2-b*: **assumes** *trans r* **and** $(M,N) \in$ *mul-eq r* **shows** *dm r M* $\subseteq$ *dm r N* **proof**
fix *x* **assume** *A*: $x \in$ *dm r M* **show** $x \in$ *dm r N* **proof** −
from *A* **obtain** *y* **where** *F2*:$y \in$*set-mset M* **and** *F3*:$(x,y) \in r$ **unfolding** *dm-def ds-def* **by** *auto*
hence $\exists\ z \in$ *set-mset N*. $(x,z) \in r$ **proof** (*cases* $y \in$*set-mset N*)
case *True* **thus** *?thesis* **using** *F3* **unfolding** *ds-def* **by** *auto*
next
case *False* **thus** *?thesis* **using** *mul-eq-not-equal-imp-elt assms F2 F3 trans-def* **by** *fast*
qed
thus *?thesis* **unfolding** *dm-def ds-def* **by** *auto*
qed
qed

Lemma 2.6.3

lemma *ds-trans-contrapos*: **assumes** *t*: *trans r* **and** $x \notin$ *ds r S* **and** $(x,y) \in r$ **shows** $y \notin$ *ds r S*
using *assms* **unfolding** *ds-def trans-def* **by** *fast*

lemma *dm-max-elt*: **assumes** *i*: *irefl r* **and** *t*: *trans r* **shows** $x \in$ *dm r M* $\Longrightarrow$ $\exists\ y \in$ *set-mset* (*M* $-s$ *dm r M*). $(x,y) \in r$
proof (*induct M arbitrary*: *x*)
case *empty* **thus** *?case* **unfolding** *dm-def ds-def* **by** *auto*
next
case (*add p P*)
hence *mem*: $x \in$ (*dm r P* $\cup$ *dm r* $\{\#p\#\}$) **unfolding** *dm-def ds-def* **by** *auto*
from *i t* **have** *not-mem-dm*: $p \notin$ *dm r* $\{\#p\#\}$ **unfolding** *dm-def ds-def irefl-def* **by** *auto*
thus *?case*
proof (*cases* $x \in$ *dm r P*)
case *False* **hence** *relp*: $(x,p) \in r$ **using** *mem* **unfolding** *dm-def ds-def* **by** *auto*
show *?thesis* **proof** (*cases* $p \in$ *dm r P*)
case *True* **thus** *?thesis* **using** *relp t ds-trans-contrapos False* **unfolding** *dm-def* **by** *fast*
next
case *False* **thus** *?thesis* **using** *not-mem-dm relp* **unfolding** *dm-def ds-def diff-def* **by** *auto*
qed
next
case *True* **obtain** *y* **where** *key*: $y \in$ *set-mset P* $y \notin$ *dm r P* $(x,y) \in r$ **using** *add*(*1*)[*OF True*] **unfolding** *diff-def* **by** *auto*
thus *?thesis*
proof (*cases* $y \in$ *dm r* $\{\#p\#\}$)
case *True* **hence** *rely*: $(y,p) \in r$ **unfolding** *dm-def ds-def* **by** *auto*
hence *relp*: $(x,p) \in r$ **using** *rely t key trans-def* **by** *metis*
have *not-memp*: $p \notin$ *set-mset P* **using** *rely key* **unfolding** *dm-def ds-def* **by** *auto*
have *memp*: $p \in$ *set-mset* (*P* + $\{\#p\#\}$) **by** *auto*

have $p \notin dm\ r\ P$ **using** *ds-trans-contrapos*[*OF t*] *key*(*2*) *rely* **unfolding** *dm-def*
by *auto*
hence $p \notin dm\ r\ (P + \{\#p\#\})$ **using** *not-mem-dm* **unfolding** *dm-def ds-def*
by *auto*
thus *?thesis* **using** *relp* **unfolding** *diff-def* **by** *auto*
next
case *False* **thus** *?thesis* **using** *key* **unfolding** *dm-def ds-def diff-def* **by** *auto*
qed
qed
qed

lemma *dm-subset*: **assumes** *i*:*irefl r* **and** *t*: *trans r* **shows** $dm\ r\ M \subseteq dm\ r\ (M -s\ dm\ r\ M)$
using *assms dm-max-elt* **unfolding** *dm-def ds-def* **by** *fast*

lemma *dm-eq*: **assumes** *i*:*irefl r* **and** *t*: *trans r* **shows** $dm\ r\ M = dm\ r\ (M -s\ dm\ r\ M)$
using *dm-subset*[*OF assms*] **unfolding** *dm-def ds-def diff-def* **by** *auto*

lemma *lemma2-6-3*: **assumes** *t*:*trans r* **and** *i*:*irefl r* **and** $(M,N) \in mul\text{-}eq\ r$
shows $\exists\ I'\ J'\ K'\ .\ N = I' + J' \wedge M = I' + K' \wedge J' \cap\# K' = \{\#\} \wedge set\text{-}mset\ K' \subseteq dm\ r\ J'$
proof −
from *assms* **obtain** *I J K* **where** *1*:$N = I + J$ **and** *2*:$M = I + K$ **and** *3*:*set-mset* $K \subseteq dm\ r\ J$ **unfolding** *mul-eq-def* **by** *auto*
have *set-mset* $(J \cap\# K) \subseteq r \downarrow m\ J$ **using** *3* **by** *auto*
then obtain *A* **where** $r \downarrow m\ J = set\text{-}mset\ (J \cap\# K) \cup A$
by *blast*
then have *key*: *set-mset* $(J -s\ dm\ r\ J) \subseteq set\text{-}mset\ (J - (J \cap\# K))$
by *clarsimp* (*metis Multiset.count-diff add.left-neutral add-diff-cancel-left' mem-Collect-eq not-gr0 set-mset-def*)
from *1 2 3* **have** $N = (I + (J \cap\# K)) + (J - (J \cap\# K))$
by (*metis diff-union-cancelL subset-mset.inf-le2 multiset-diff-union-assoc multiset-inter-commute union-commute union-lcomm*)
moreover have $M = (I + (J \cap\# K)) + (K - (J \cap\# K))$
by (*metis diff-subset-eq-self diff-union-cancelL 2 multiset-diff-union-assoc multiset-inter-commute multiset-inter-def union-assoc*)
moreover have *set-mset* $(K-(J\cap\#K)) \subseteq dm\ r\ (J-(J\cap\#K))$
proof −
have *set-mset* $(K-(J\cap\#K)) \subseteq dm\ r\ J$ **using** *3*
by (*meson Multiset.diff-subset-eq-self mset-subset-eqD subset-eq*)
moreover have $... = dm\ r\ (J -s\ dm\ r\ J)$ **using** *dm-eq*[*OF i t*] **by** *auto*
moreover have $... \subseteq dm\ r\ (J - (J \cap\# K))$ **using** *ds-monotone*[*OF key*]
unfolding *dm-def* **by** *auto*
ultimately show *?thesis* **by** *auto*
qed
moreover have $(J-(J\cap\#K)) \cap\# (K-(J\cap\#K)) = \{\#\}$ **by** (*rule multiset-eqI*) *auto*
ultimately show *?thesis* **by** *auto*

qed

Lemma 2.6.4

lemma *lemma2-6-4*: **assumes** *t*: *trans r* **and** $N \neq \{\#\}$ **and** *set-mset M* $\subseteq$ *dm r N* **shows** (*M*,*N*) $\in$ *mul r* **proof** −
have $M = \{\#\} + M$ **and** $N = \{\#\} + N$ **using** *assms* **by** *auto*
thus *?thesis* **using** *assms*(*2*,*3*) **unfolding** *mul-def* **by** *fast*
qed

lemma *lemma2-6-5-a*: **assumes** *t*: *trans r* **and** *ds r S* $\subseteq$ *S* **and** (*M*,*N*) $\in$ *mul-eq r*
shows (*M* $-s$ *S*, *N* $-s$ *S*) $\in$ *mul-eq r*
proof −
from *assms*(*1*,*3*)
obtain *I J K* **where** *a*: *N*=*I*+*J* **and** *b*:*M*=*I*+*K* **and** *c*:*set-mset K* $\subseteq$ *dm r J* **unfolding** *mul-eq-def* **by** *best*
from *a b* **have** *M* $-s$ *S* = *I* $-s$ *S* + *K* $-s$ *S*
N $-s$ *S* = *I* $-s$ *S* + *J* $-s$ *S* **by** (*auto simp add*: *lemmaA-3-8*)
moreover have *set-mset* (*K*−*sS*) $\subseteq$ *dm r* (*J*−*sS*) **proof** −
have *set-mset* (*K*−*sS*) $\subseteq$ *set-mset* (*K*−*s* (*ds r S*)) **using** *assms*(*2*) **unfolding** *diff-def* **by** *auto*
moreover have *set-mset*(*K*−*s* (*ds r S*)) $\subseteq$ (*dm r J*) − (*ds r S*) **using** *c* **unfolding** *diff-def* **by** *auto*
moreover have (*dm r J*) − (*ds r S*) $\subseteq$ *dm r* (*J* $-s$ *S*) **using** *lemma2-6-1-diff* **by** *fast*
ultimately show *?thesis* **by** *auto*
qed
ultimately show *?thesis* **by** (*rule in-mul-eqI*)
qed

lemma *lemma2-6-5-a′*: **assumes** *t*:*trans r* **and** (*M*,*N*) $\in$ *mul-eq r* **shows** (*M* $-s$ *ds r S*, *N* $-s$ *ds r S*) $\in$ *mul-eq r*
using *assms lemma2-6-5-a*[*OF t*] *ds-ds-subseteq-ds*[*OF t*] **by** *auto*

Lemma 2.6.6

lemma *lemma2-6-6-a*: **assumes** *t*: *trans r* **and** (*M*,*N*) $\in$ *mul-eq r* **shows** (*Q* + *M*,*Q* + *N*) $\in$ *mul-eq r* **proof** −
obtain *I J K* **where** *A*:*Q*+*N*=(*Q*+*I*)+*J* **and** *B*:*Q*+*M*=(*Q*+*I*)+*K* **and** *C*:*set-mset K* $\subseteq$ *dm r J*
using *assms*(*2*) **unfolding** *mul-eq-def* **by** *auto*
thus *?thesis* **using** *C* **unfolding** *mul-eq-def* **by** *blast*
qed

lemma *add-left-one*:
assumes $\exists$ *I J K*. *add-mset q N* = *I* + *J* $\wedge$ *add-mset q M* = *I* + *K* $\wedge$ (*J*∩#*K*={#}) $\wedge$ *set-mset K* $\subseteq$ *dm r J*
shows $\exists$ *I2 J K*. *N* = *I2* + *J* $\wedge$ *M* = *I2* + *K* $\wedge$ *set-mset K* $\subseteq$ *dm r J* **proof** −
from *assms* **obtain** *I J K* **where** *A*: {#*q*#} + *N* = *I* + *J* **and** *B*:{#*q*#} + *M* = *I* + *K*

and *C*:$(J \cap\# K = \{\#\})$ **and** *D*:*set-mset K* $\subseteq$ *dm r J* **by** *auto*
have *q*$\in\#I$ **proof** (*cases q* $\in\#$ *I*)
case *True* **thus** *?thesis* **by** *auto*
next
case *False*
have *q* $\in\#$ *J* **using** *False A* **by** (*metis UnE multi-member-this set-mset-union*)
moreover have *q* $\in\#$ *K* **using** *False B* **by** (*metis UnE multi-member-this set-mset-union*)
moreover have $\neg$ *q* $\in\#$ (*J* $\cap\#$ *K*) **using** *C* **by** *auto*
ultimately show *?thesis* **by** *auto*
qed
hence $\exists$ *I2*. *I* = *add-mset q I2* **by** (*metis multi-member-split union-commute*)
hence $\exists$ *I2*. *add-mset q N* = (*add-mset q I2*) + *J* $\wedge$ *add-mset q M* = (*add-mset q I2*) + *K* **using** *A B* **by** *auto*
thus *?thesis* **using** *D* **by** *auto*
qed

lemma *lemma2-6-6-b-one* :
assumes *trans r* **and** *irefl r* **and** (*add-mset q M*, *add-mset q N*) $\in$ *mul-eq r*
shows (*M*,*N*) $\in$ *mul-eq r*
using *add-left-one*[*OF lemma2-6-3*[*OF assms*]] **unfolding** *mul-eq-def* **by** *auto*

lemma *lemma2-6-6-b′* : **assumes** *trans r* **and** *i*: *irefl r* **and** (*Q* + *M*, *Q* + *N*) $\in$ *mul-eq r*
shows (*M*,*N*) $\in$ *mul-eq r* **using** *assms*(*3*) **proof** (*induct Q arbitrary*: *M N*)
case *empty* **thus** *?case* **by** *auto*
next
case (*add q Q*) **thus** *?case* **unfolding** *union-assoc* **using** *lemma2-6-6-b-one*[*OF assms*(*1*,*2*)]
by (*metis union-mset-add-mset-left*)
qed

lemma *lemma2-6-9*: **assumes** *t*: *trans r* **and** (*M*,*N*) $\in$ *mul r* **shows** (*Q*+*M*,*Q*+*N*) $\in$ *mul r* **proof** −
obtain *I J K* **where** *F1*:*N* = *I* + *J* **and** *F2*:*M* = *I* + *K***and** *F3*:*set-mset K* $\subseteq$ *dm r J* **and** *F4*:*J* $\neq$ {#}
using *assms* **unfolding** *mul-def* **by** *auto*
have *Q*+*N*=*Q*+*I*+*J* **and** *Q*+*M*=*Q*+*I*+*K* **by** (*auto simp*: *F1 F2 union-commute union-lcomm*)
thus *?thesis* **using** *assms*(*1*) *F3 F4* **unfolding** *mul-def* **by** *blast*
qed

Lemma 2.6.7

lemma *lemma2-6-7-a*: **assumes** *t*: *trans r* **and** *set-mset Q* $\subseteq$ *dm r N* − *dm r M* **and** (*M*,*N*) $\in$ *mul-eq r*
shows (*Q*+*M*,*N*) $\in$ *mul-eq r* **proof** −
obtain *I J K* **where** *A*: *N*=*I*+*J* **and** *B*:*M*=*I*+*K* **and** *C*:*set-mset K* $\subseteq$ *dm r J* **using** *assms* **unfolding** *mul-eq-def* **by** *auto*
hence *set-mset*(*Q*+*K*) $\subseteq$ *dm r J* **using** *assms lemma2-6-1-diff* **unfolding** *dm-def*

ds-def **by** *auto*
hence $(I+(Q+K),I+J) \in$ *mul-eq r* **unfolding** *mul-eq-def* **by** *fast*
thus *?thesis* **using** *A B C union-assoc union-commute* **by** *metis*
qed

missing?; similar to lemma_2.6.2?

lemma *lemma2-6-8*: **assumes** *t*: *trans r* **and** $S \subseteq T$ **shows** $(M -s T, M -s S) \in$ *mul-eq r* **proof** −
from *assms*(*2*) **have** $(M -s T) \subseteq\# (M -s S)$
by (*metis Un-absorb2 Un-commute lemmaA-3-10 lemmaA-3-9 mset-subset-eq-add-right*)
thus *?thesis* **using** *lemma2-6-2-a*[*OF t*] **by** *auto*
qed

1.1.3 Lexicographic maximum measure

Def 3.1: lexicographic maximum measure

fun *lexmax* :: *'a rel* ⇒ *'a list* ⇒ *'a multiset* **where**
lexmax r [] = {#}
| *lexmax r* (α#σ) = {#α#} + (*lexmax r* σ −*s ds r* {α})

notation
lexmax (-|-| [*1000*] *1000*)

lemma *lexmax-singleton*: $r|[\alpha]| = \{\#\alpha\#\}$ **unfolding** *lexmax.simps diff-def* **by** *simp*

Lemma 3.2

Lemma 3.2(1)

lemma *lemma3-2-1*: **assumes** *t*: *trans r* **shows** $r \downarrow m\ r|\sigma| = r \downarrow l\ \sigma$ **proof** (*induct* σ)
case *Nil* **show** *?case* **unfolding** *dm-def dl-def* **by** *auto*
next
case (*Cons* α σ)
have *dm r* {#α#} ∪ *dm r* (*r*|σ| −*s ds r* {α}) = *dm r* {#α#} ∪ *dl r* σ (**is** *?L* = *?R*) **proof** −
have *?L* ⊆ *?R* **unfolding** *Cons*[*symmetric*] *diff-def dm-def ds-def* **by** *auto*
moreover have *?R* ⊆ *?L* **proof**
fix *x* **assume** *A*: *x* ∈ *?R* **show** *x* ∈ *?L* **proof** (*cases x* ∈ *dm r* {#α#})
case *True* **thus** *?thesis* **by** *auto*
next
case *False*
hence *mem*: *x* ∈ *dm r* (*lexmax r* σ) **using** *A Cons* **by** *auto*
have *x* ∉ (*ds r* (*ds r* {α})) **using** *False ds-ds-subseteq-ds*[*OF t*] **unfolding** *dm-def* **by** *auto*
thus *?thesis* **using** *mem lemma2-6-1-diff* **by** *fast*
qed
qed
ultimately show *?thesis* **by** *auto*

qed
thus *?case* **unfolding** *lemma2-6-1-multiset lexmax.simps dl-def dm-def ds-def* **by** *auto*
qed

Lemma 3.2(2)

lemma *lemma3-2-2*: $r|\sigma@\tau| = r|\sigma| + (r|\tau| -s\ r \downarrow l\ \sigma)$ **proof**(*induct* σ)
case *Nil* **thus** *?case* **unfolding** *dl-def ds-def lexmax.simps* **apply** *auto* **unfolding** *diff-empty* **by** *auto*
next
case (*Cons* $\alpha\ \sigma$)
have *lexmax* $r\ (\alpha\#\sigma@\tau) = \{\#\alpha\#\} + ((lexmax\ r\ (\sigma@\tau)) -s\ (ds\ r\ \{\alpha\}))$ **by** *auto*
moreover have $\ldots = \{\#\alpha\#\} + ((lexmax\ r\ \sigma + ((lexmax\ r\ \tau) -s\ (dl\ r\ \sigma))) -s\ (ds\ r\ \{\alpha\}))$
using *Cons* **by** *auto*
moreover have $\ldots = \{\#\alpha\#\} + ((lexmax\ r\ \sigma) -s\ (ds\ r\ \{\alpha\})) + (((lexmax\ r\ \tau) -s\ (dl\ r\ \sigma)) -s\ (ds\ r\ \{\alpha\}))$
unfolding *lemmaA-3-8* **unfolding** *union-assoc* **by** *auto*
moreover have $\ldots = lexmax\ r\ (\alpha\#\sigma) + (((lexmax\ r\ \tau) -s\ (dl\ r\ \sigma)) -s\ (ds\ r\ \{\alpha\}))$ **by** *simp*
moreover have $\ldots = lexmax\ r\ (\alpha\#\sigma) + ((lexmax\ r\ \tau) -s\ (dl\ r\ (\alpha\#\sigma)))$ **unfolding** *lemmaA-3-9 dl-def dm-def lemma2-6-1-set*[*symmetric*] **by** *auto*
ultimately show *?case* **unfolding** *diff-def* **by** *simp*
qed

Definition 3.3

definition $D :: 'a\ rel \Rightarrow 'a\ list \Rightarrow 'a\ list \Rightarrow 'a\ list \Rightarrow 'a\ list \Rightarrow bool$ **where**
$D\ r\ \tau\ \sigma\ \sigma'\ \tau' = ((r|\sigma@\tau'|,\ r|\tau| + r|\sigma|\ ) \in mul\text{-}eq\ r$
$\wedge\ (r|\tau@\sigma'|,\ r|\tau| + r|\sigma|\ ) \in mul\text{-}eq\ r)$

lemma *D-eq*: **assumes** *trans r* **and** *irrefl r* **and** $D\ r\ \tau\ \sigma\ \sigma'\ \tau'$
shows $(r|\tau'| -s\ dl\ r\ \sigma, r|\tau|\ ) \in mul\text{-}eq\ r$ **and** $(r|\sigma'| -s\ dl\ r\ \tau, r|\sigma|\ ) \in mul\text{-}eq\ r$
using *assms* **unfolding** *D-def lemma3-2-2* **using** *union-commute lemma2-6-6-b'* **apply** *metis*
using *assms* **unfolding** *D-def lemma3-2-2* **using** *union-commute lemma2-6-6-b'* **by** *metis*

lemma *D-inv*: **assumes** *trans r* **and** *irrefl r* **and** $(r|\tau'| -s\ dl\ r\ \sigma, r|\tau|\ ) \in mul\text{-}eq\ r$
and $(r|\sigma'| -s\ dl\ r\ \tau, r|\sigma|\ ) \in mul\text{-}eq\ r$
shows $D\ r\ \tau\ \sigma\ \sigma'\ \tau'$
using *assms* **unfolding** *D-def lemma3-2-2* **using** *lemma2-6-6-a*[*OF assms*(*1*)] *union-commute* **by** *metis*

lemma *D*: **assumes** *trans r* **and** *irrefl r*
shows $D\ r\ \tau\ \sigma\ \sigma'\ \tau' = ((r|\tau'| -s\ dl\ r\ \sigma, r|\tau|\ ) \in mul\text{-}eq\ r$
$\wedge\ (r|\sigma'| -s\ dl\ r\ \tau, r|\sigma|\ ) \in mul\text{-}eq\ r)$
using *assms D-eq D-inv* **by** *auto*

lemma *mirror-D*: **assumes** *trans r* **and** *irrefl r* **and** *D r τ σ σ′ τ′* **shows** *D r σ τ τ′ σ′*
using *assms D* **by** *metis*

Proposition 3.4

definition *LD-1′* :: *'a rel ⇒ 'a ⇒ 'a ⇒ 'a list ⇒ 'a list ⇒ 'a list ⇒ bool*
where *LD-1′ r β α σ1 σ2 σ3* =
(*set σ1 ⊆ ds r* {*β*} ∧ *length σ2 ≤ 1* ∧ *set σ2 ⊆* {*α*} ∧ *set σ3 ⊆ ds r* {*α*,*β*})

definition *LD′* :: *'a rel ⇒ 'a ⇒ 'a*
⇒ 'a list ⇒ 'a list ⇒ 'a list ⇒ 'a list ⇒ 'a list ⇒ bool
where *LD′ r β α σ1 σ2 σ3 τ1 τ2 τ3* = (*LD-1′ r β α σ1 σ2 σ3* ∧ *LD-1′ r α β τ1 τ2 τ3*)

auxiliary properties on multisets

lemma *lexmax-le-multiset*: **assumes** *t*:*trans r* **shows** *r*|*σ*| ⊆# *mset σ* **proof** (*induct σ*)
case *Nil* **thus** *?case* **unfolding** *lexmax.simps* **by** *auto*
next
case (*Cons s τ*) **hence** *lexmax r τ −s ds r* {*s*} ⊆# *mset τ* **using** *lemmaA-3-10 mset-subset-eq-add-right subset-mset.order-trans* **by** *metis*
thus *?case* **by** *simp*
qed

lemma *split*: **assumes** *LD-1′ r β α σ1 σ2 σ3* **shows** *σ2* = [] ∨ *σ2* = [*α*]
using *assms* **unfolding** *LD-1′-def* **by** (*cases σ2*) *auto*

lemma *proposition3-4-step*: **assumes** *trans r* **and** *irrefl r* **and** *LD-1′ r β α σ1 σ2 σ3*
shows (*r*|*σ1*@*σ2*@*σ3*| *−s* (*dm r* {#*β*#}), *r*|[*α*]|) ∈ *mul-eq r* **proof** −
from *assms* **have** *set σ1 ⊆ dm r* {#*β*#} **unfolding** *LD′-def LD-1′-def dm-def* **by** *auto*
hence *set-mset* (*lexmax r σ1*) ⊆ *dm r* {#*β*#} **using** *submultiset-implies-subset*[*OF lexmax-le-multiset*[*OF assms*(*1*)]] **by** *auto*
hence *one*: *lexmax r σ1 −s dm r* {#*β*#} = {#} **using** *subset-implies-remove-empty* **by** *auto*
from *assms* **have** *set σ3 ⊆ dl r σ2 ∪ dl r σ1 ∪ dm r* {#*β*#} ∪ *dm r* {#*α*#} (**is** *?l ⊆ ?r*) **unfolding** *LD′-def LD-1′-def dm-def ds-def* **by** *auto*
hence *set-mset* (*lexmax r σ3*) ⊆ *?r* **using** *submultiset-implies-subset*[*OF lexmax-le-multiset*[*OF assms*(*1*)]] **by** *auto*
hence *pre3*: *lexmax r σ3 −s ?r* = {#} **using** *subset-implies-remove-empty* **by** *auto*
show *?thesis* **proof** (*cases σ2* = [])
case *True*
hence *two*:(*lexmax r σ2 −s dl r σ1*) *−s dm r* {#*β*#} = {#} **unfolding** *diff-def* **by** *auto*
from *pre3* **have** (((*lexmax r σ3 −s dl r σ2*) *−s dl r σ1*) *−s dm r* {#*β*#}) *−s dm r* {#*α*#} = {#} **unfolding** *True dl-def lemmaA-3-9* **by** *auto*
hence *three*:(((*lexmax r σ3 −s dl r σ2*) *−s dl r σ1*) *−s dm r* {#*β*#},{#*α*#})

$\in$ *mul r* **using** *remove-is-empty-imp-mul* **by** *metis*
show *?thesis* **using** *three* **unfolding** *lemma3-2-2 lexmax-singleton lemmaA-3-8 one two mul-def mul-eq-def* **by** *auto*
next
case *False* **hence** *eq*: $\sigma 2 = [\alpha]$ **using** *split*[*OF assms*(*3*)] **by** *fast*
have *two*: $((lexmax\ r\ \sigma 2\ -s\ dl\ r\ \sigma 1)\ -s\ dm\ r\ \{\#\beta\#\},\{\#\alpha\#\}) \in mul\text{-}eq\ r$ **using** *lemma2-6-8*[*OF assms*(*1*) *empty-subsetI*] **unfolding** *eq lexmax.simps diff-from-empty lemmaA-3-9 diff-empty* **by** *auto*
from *pre3* **have** $lexmax\ r\ \sigma 3\ -s\ ((dl\ r\ \sigma 2 \cup dm\ r\ \{\#\alpha\#\}) \cup dl\ r\ \sigma 1 \cup dm\ r\ \{\#\beta\#\}) = \{\#\}$ **unfolding** *eq lemmaA-3-9* **using** *Un-assoc Un-commute* **by** *metis*
hence *three*: $((lexmax\ r\ \sigma 3\ -s\ dl\ r\ \sigma 2)\ -s\ dl\ r\ \sigma 1)\ -s\ dm\ r\ \{\#\beta\#\} = \{\#\}$ **using** *Un-absorb* **unfolding** *lemmaA-3-9 eq dm-def dl-def* **by** *auto*
show *?thesis* **unfolding** *lemma3-2-2 lexmax-singleton lemmaA-3-8 one three* **using** *two* **by** *auto*
qed
qed

lemma *proposition3-4*:
assumes *t*: *trans r* **and** *i*: *irrefl r* **and** *ld*: *LD' r* $\beta\ \alpha\ \sigma 1\ \sigma 2\ \sigma 3\ \tau 1\ \tau 2\ \tau 3$
shows *D r* $[\beta]\ [\alpha]\ (\sigma 1@\sigma 2@\sigma 3)\ (\tau 1@\tau 2@\tau 3)$
using *proposition3-4-step*[*OF t i*] *ld* **unfolding** *LD'-def D*[*OF assms*(*1,2*)] *dl-def dm-def* **by** *auto*

lemma *lexmax-decompose*: **assumes** $\alpha \in\#\ r|\sigma|$ **shows** $\exists \sigma 1\ \sigma 3.\ (\sigma = \sigma 1@[\alpha]@\sigma 3 \wedge \alpha \notin dl\ r\ \sigma 1)$
using *assms* **proof** (*induct* σ)
case *Nil* **thus** *?case* **by** *auto*
next
case (*Cons* β *as*) **thus** *?case* **proof** (*cases* $\alpha = \beta$)
case *True*
from *this* **obtain** $\sigma 1\ \sigma 3$ **where** *dec*: $\beta \# as = \sigma 1@[\alpha]@\sigma 3$ **and** *empty*: $\sigma 1 = []$ **by** *auto*
hence $\alpha \notin dl\ r\ \sigma 1$ **unfolding** *dl-def ds-def* **by** *auto*
thus *?thesis* **using** *dec* **by** *auto*
next
case *False* **hence** $\alpha \in\#\ r|as| - s\ ds\ r\ \{\beta\}$ **using** *Cons*(*2*) **by** *auto*
hence *x*: $\alpha \in\#\ r|as| \wedge \alpha \notin ds\ r\ \{\beta\}$
by *simp*
from *this* **obtain** $\sigma 1\ \sigma 3$ **where** $as = \sigma 1\ @\ [\alpha]\ @\ \sigma 3$ **and** *w*: $\alpha \notin dl\ r\ \sigma 1$ **using** *Cons*(*1*) **by** *auto*
hence $\beta \# as = (\beta \# \sigma 1)\ @\ [\alpha]\ @\ \sigma 3$ **and** $\alpha \notin dl\ r\ (\beta \# \sigma 1)$ **using** *x w* **unfolding** *dm-def dl-def ds-def* **by** *auto*
thus *?thesis* **by** *fast*
qed
qed

lemma *lexmax-elt*: **assumes** *trans r* **and** $x \in (set\ \sigma)$ **and** $x \notin set\text{-}mset\ r|\sigma|$

shows $\exists y.\ (x,y) \in r \wedge y \in$ *set-mset* $r|\sigma|$ **using** *assms(2,3)* **proof** (*induct* σ)
case *Nil* **thus** *?case* **by** *auto*
next
case (*Cons a as*) **thus** *?case* **proof** (*cases* $x \notin$ *set-mset* $r|as|$)
case *True*
from *Cons True* **obtain** y **where** *s*: $(x,\ y) \in r \wedge y \in$ *set-mset* $r|as|$ **by** *auto*
thus *?thesis* **proof** (*cases* $y \in ds\ r\ \{a\}$)
case *True* **thus** *?thesis* **using** *transD*[*OF assms(1)*] *s* **unfolding** *dm-def ds-def* **by** *auto*
next
case *False* **thus** *?thesis* **using** *s* **unfolding** *lexmax.simps diff-def* **by** *auto*
qed
next
case *False* **thus** *?thesis* **using** *Cons* **unfolding** *diff-def dm-def ds-def lexmax.simps* **by** *auto*
qed
qed

lemma *lexmax-set*: **assumes** *trans r* **and** *set-mset* $r|\sigma| \subseteq r \downarrow s\ S$ **shows** *set* $\sigma \subseteq r \downarrow s\ S$ **proof**
fix x **assume** *A*: $x \in$ *set* σ **show** $x \in ds\ r\ S$ **proof** (*cases* $x \in$ *set-mset* $r|\sigma|$)
case *True* **thus** *?thesis* **using** *assms* **by** *auto*
next
case *False* **from** *lexmax-elt*[*OF assms(1) A False*] **obtain** y
where *rel*: $(x,y) \in r \wedge y \in$ *set-mset* $r|\sigma|$ **by** *auto*
hence $y \in ds\ r\ S$ **using** *assms* **by** *auto*
thus *?thesis* **using** *rel assms transD* **unfolding** *dm-def ds-def* **by** *fast*
qed
qed

lemma *drop-left-mult-eq*:
assumes *trans r* **and** *irrefl r* **and** $(N+M,M) \in$ *mul-eq r* **shows** $N = \{\#\}$ **proof** −
have $(M+N,M+\{\#\}) \in$ *mul-eq r* **using** *assms(3)* **apply** *auto* **using** *union-commute* **by** *metis*
hence *k*:$(N,\{\#\}) \in$ *mul-eq r* **using** *lemma2-6-6-b'*[*OF assms(1,2)*] **by** *fast*
from *this* **obtain** *I J K* **where** $\{\#\} = I + J \wedge N = I + K \wedge$ *set-mset* $K \subseteq dm\ r\ J$ **unfolding** *mul-eq-def* **by** *fast*
thus *?thesis* **unfolding** *dm-def ds-def* **by** *auto*
qed

generalized to lists

lemma *proposition3-4-inv-lists*:
assumes *t*: *trans r* **and** *i*: *irrefl r* **and** *k*:$(r|\sigma| - s\ r \downarrow l\ \beta,\ \{\#\alpha\#\}) \in$ *mul-eq r* (**is** (*?M*,-) ∈ -)
shows $\exists\ \sigma 1\ \sigma 2\ \sigma 3.\ ((\sigma = \sigma 1@\sigma 2@\sigma 3) \wedge$ *set* $\sigma 1 \subseteq dl\ r\ \beta \wedge$ *length* $\sigma 2 \leq 1 \wedge$ *set* $\sigma 2 \subseteq \{\alpha\}) \wedge$ *set* $\sigma 3 \subseteq dl\ r\ (\alpha\#\beta)$ **proof** (*cases* $\alpha \in\#$ *?M*)
case *True* **hence** $\alpha \in\#\ r|\sigma|$ **by** *simp*
from *this* **obtain** $\sigma 1\ \sigma 3$ **where** *sigma*: $\sigma = \sigma 1@[\alpha]@\sigma 3$ **and** *alpha*: $\alpha \notin dl\ r\ \sigma 1$

using *lexmax-decompose* **by** *metis*
hence *dec*: $((r|\sigma 1|-sdl\ r\ \beta) + (r|[\alpha]|-s\ (dl\ r\ \sigma 1 \cup dl\ r\ \beta)) + (r|\sigma 3| -s\ (dl\ r\ [\alpha] \cup dl\ r\ \sigma 1 \cup dl\ r\ \beta)), \{\#\alpha\#\}) \in mul\text{-}eq\ r$ (**is** $(?M1 + ?M2 + ?M3,\text{-}) \in \text{-}$)
using *k* **unfolding** *sigma lemma3-2-2 lemmaA-3-8 lemmaA-3-9 LD-1′-def union-assoc* **by** *auto*

from *True* **have** *key*: $\alpha \notin dl\ r\ \beta$ **by** *simp*
hence $?M2 = \{\#\alpha\#\}$ **unfolding** *lexmax-singleton diff-def* **using** *alpha* **by** *auto*
hence $(?M1+?M3 + \{\#\alpha\#\},\{\#\alpha\#\}) \in mul\text{-}eq\ r$ **using** *dec union-assoc union-commute* **by** *metis*
hence *w*: $?M1+?M3 = \{\#\}$ **using** *drop-left-mult-eq assms*(*1*,*2*) **by** *blast*
from *w* **have** $(r|\sigma 1|-sdl\ r\ \beta) = \{\#\}$ **by** *auto*
hence $set\text{-}mset\ r|\sigma 1| \subseteq ds\ r\ (set\ \beta)$ **using** *remove-empty-implies-subset* **unfolding** *dl-def dm-def* **by** *auto*
hence *sigma1*: $set\ \sigma 1 \subseteq ds\ r\ (set\ \beta)$ **using** *lexmax-set*[*OF assms*(*1*)] **by** *auto*

have *sigma2*: $length\ [\alpha] \leq 1 \wedge set\ [\alpha] \subseteq \{\alpha\}$ **by** *auto*

have *sub*:$dl\ r\ \sigma 1 \subseteq dl\ r\ \beta$ **using** *subset-imp-ds-subset*[*OF assms*(*1*) *sigma1*] **unfolding** *dm-def dl-def* **by** *auto*
hence *sub2*: $dl\ r\ \sigma 1 \cup dl\ r\ \beta = dl\ r\ \beta$ **by** *auto*
from *w* **have** $?M3 = \{\#\}$ **by** *auto*
hence $r|\sigma 3|-s\ (ds\ r\ \{\alpha\} \cup ds\ r\ (set\ \beta)) = \{\#\}$ **unfolding** *Un-assoc sub2* **unfolding** *dl-def* **by** *auto*
hence $r|\sigma 3|-s\ (ds\ r\ (\{\alpha\} \cup (set\ \beta))) = \{\#\}$ **unfolding** *lemma2-6-1-set*[*symmetric*] **by** *metis*
hence $set\text{-}mset\ r|\sigma 3| \subseteq ds\ r\ (\{\alpha\} \cup (set\ \beta))$ **using** *remove-empty-implies-subset* **by** *auto*
hence *sigma3*: $set\ \sigma 3 \subseteq ds\ r\ (\{\alpha\} \cup (set\ \beta))$ **using** *lexmax-set*[*OF assms*(*1*)] **by** *auto*
show *?thesis* **using** *sigma sigma1 sigma2 sigma3* **unfolding** *dl-def* **apply** *auto* **by** (*metis One-nat-def append-Cons append-Nil sigma2*)
next
case *False* **hence** $set\text{-}mset\ ?M \subseteq dm\ r\ \{\#\alpha\#\}$ **using** *mul-eq-singleton*[*OF k*]
by (*auto dest*: *diff-eq-singleton-imp*)
hence $set\text{-}mset\ r|\sigma| \subseteq ds\ r\ (\{\alpha\} \cup (set\ \beta))$ **unfolding** *diff-def dm-def dl-def ds-def* **by** *auto*
hence $set\ \sigma \subseteq ds\ r\ (\{\alpha\} \cup (set\ \beta))$ **using** *lexmax-set*[*OF assms*(*1*)] **by** *auto*
thus *?thesis* **unfolding** *dl-def* **apply** *auto* **by** (*metis append-Nil bot-least empty-set le0 length-0-conv*)
qed

lemma *proposition3-4-inv-step*:
assumes *t*: *trans r* **and** *i*: *irrefl r* **and** *k*:$(r|\sigma| -s\ r \downarrow l\ [\beta], \{\#\alpha\#\}) \in mul\text{-}eq\ r$ (**is** $(?M,\text{-}) \in \text{-}$)
shows $\exists\ \sigma 1\ \sigma 2\ \sigma 3.\ ((\sigma = \sigma 1@\sigma 2@\sigma 3) \wedge LD\text{-}1'\ r\ \beta\ \alpha\ \sigma 1\ \sigma 2\ \sigma 3)$
using *proposition3-4-inv-lists*[*OF assms*] **unfolding** *LD-1′-def dl-def* **by** *auto*

lemma *proposition3-4-inv*:

assumes *t*: *trans r* **and** *i*: *irrefl r* **and** *D r* [*β*] [*α*] *σ τ*
shows ∃ *σ1 σ2 σ3 τ1 τ2 τ3*. (*σ* = *σ1*@*σ2*@*σ3* ∧ *τ* = *τ1*@*τ2*@*τ3* ∧ *LD′ r β α σ1 σ2 σ3 τ1 τ2 τ3*)
using *proposition3-4-inv-step*[*OF assms*(*1,2*)] *D-eq*[*OF assms*] **unfolding** *lexmax-singleton LD′-def* **by** *metis*

Lemma 3.5

lemma *lemma3-5-1*:
assumes *t*: *trans r* **and** *irrefl r* **and** *D r τ σ σ′ τ′* **and** *D r υ σ′ σ″ υ′*
shows (*lexmax r* (*τ* @ *υ* @ *σ″*), *lexmax r* (*τ* @ *υ*) + *lexmax r σ*) ∈ *mul-eq r* **proof** −
have *lexmax r* (*τ* @ *υ* @ *σ″*) = (*lexmax r* (*τ* @ *υ*) + ((*lexmax r σ″*) −*s* (*dl r* (*τ*@*υ*))))
unfolding *append-assoc*[*symmetric*] **using** *lemma3-2-2* **by** *fast*
moreover have *x*:... = *lexmax r* (*τ*@*υ*) + (((*lexmax r σ″*) −*s dl r υ*) −*s dl r τ*)
by (*auto simp*: *lemma2-6-1-list lemmaA-3-9 Un-commute*)
moreover have (...,*lexmax r* (*τ*@*υ*) + (*lexmax r σ′* −*s dl r τ*)) ∈ *mul-eq r* (**is** (-,*?R*) ∈ -)
using *lemma2-6-6-a*[*OF t lemma2-6-5-a′*[*OF t D-eq*(*2*)[*OF assms*(*1,2,4*)]]] **unfolding** *dl-def* **by** *auto*
moreover have (*?R*,*lexmax r* (*τ*@*υ*) + *lexmax r σ*) ∈ *mul-eq r*
using *lemma2-6-6-a*[*OF t D-eq*(*2*)[*OF assms*(*1*−*3*)]] **by** *auto*
ultimately show *?thesis* **using** *mul-eq-trans*[*OF t*] **by** *metis*
qed

lemma *claim1*: **assumes** *t*: *trans r* **and** *D r τ σ σ′ τ′*
shows (*r*|*σ*@*τ′*| + ((*r*|*υ′*| −*s r* ↓*l* (*σ*@*τ′*)) ∩*s r* ↓*l τ*),*r*|*σ*| + *r*|*τ*|) ∈ *mul-eq r* (**is** (*?F*+*?H*,*?G*) ∈ -)
proof −
have *1*: (*?F*,*?G*) ∈ *mul-eq r* **using** *assms*(*2*) **unfolding** *D-def* **by** (*auto simp*: *union-commute*)
have *2*: *set-mset ?H* ⊆ (*dm r ?G*) − (*dm r ?F*) (**is** (*?L* ⊆ -)) **proof** −
have *set-mset ?H* = *set-mset* ((*lexmax r υ′* ∩*s dl r τ*) −*s dl r* (*σ*@*τ′*)) **unfolding** *lemmaA-3-11* **by** *auto*
moreover have ... ⊆ (*dl r τ* − *dl r* (*σ*@*τ′*)) **unfolding** *diff-def intersect-def* **by** *auto*
moreover have ... ⊆ ((*dl r σ* ∪ *dl r τ*) − *dl r* (*σ*@*τ′*)) **by** *auto*
ultimately show *?thesis* **unfolding** *lemma2-6-1-multiset lemma3-2-1*[*OF t*] **by** *auto*
qed
show *?thesis* **using** *lemma2-6-7-a*[*OF t 2 1*] **by** (*auto simp*: *union-commute*)
qed

lemma *step3*: **assumes** *t*:*trans r* **and** *D r τ σ σ′ τ′*
shows *r* ↓*l* (*σ*@*τ*) ⊇ (*r* ↓*m* (*r*|*σ′*| + *r*|*τ*|)) **proof** −
have *a*: *dl r* (*σ*@*τ*) = *dm r* (*lexmax r τ* + *lexmax r σ*) **and** *b*: *dl r* (*τ*@*σ′*) = *dm r* (*lexmax r σ′* + *lexmax r τ*)
unfolding *lemma2-6-1-list lemma3-2-1*[*OF t*,*symmetric*] *lemma2-6-1-multiset* **by** *auto*

show *?thesis* **using** *assms(2) lemma2-6-2-b[OF t] lemma3-2-1[OF t,symmetric]* **unfolding** *D-def a b[symmetric]* **by** *auto*
qed

lemma *claim2*: **assumes** *t*: *trans r* **and** $D\ r\ \tau\ \sigma\ \sigma'\ \tau'$
shows $((r|\upsilon'|\ -s\ \ r\downarrow l\ (\sigma@\tau'))\ -s\ r\downarrow l\ \tau,\ (r|\upsilon'|\ -s\ r\downarrow l\ \sigma')\ -s\ r\downarrow l\ \tau) \in$ *mul-eq r* (**is** (*?L*,*?R*) ∈ -)
proof −
have $?L = lexmax\ r\ \upsilon' -s\ (dl\ r\ (\sigma@\tau'@\tau))$ **unfolding** *lemmaA-3-9 append-assoc[symmetric] lemma2-6-1-list* **by** *auto*
moreover have (. . .,$lexmax\ r\ \upsilon' -s\ dl\ r\ (\sigma@\tau)) \in$ *mul-eq r* (**is** (-,*?R*) ∈ -) **using** *lemma2-6-8[OF t dl-monotone]* **by** *auto*
moreover have (*?R*,$lexmax\ r\ \upsilon' -s\ dm\ r\ (lexmax\ r\ \sigma' + lexmax\ r\ \tau)) \in$ *mul-eq r* (**is** (-,*?R*) ∈ -) **using** *lemma2-6-8[OF t step3[OF assms]]* **by** *auto*
moreover have $?R = (lexmax\ r\ \upsilon' -s\ dl\ r\ \sigma') -s\ dl\ r\ \tau$ **unfolding** *lemma3-2-1[OF t,symmetric] lemma2-6-1-multiset lemmaA-3-9[symmetric]* **by** *auto*
ultimately show *?thesis* **using** *mul-eq-trans[OF t]* **by** *metis*
qed

lemma *lemma3-5-2*: **assumes** *trans r* **and** *irefl r* **and** $D\ r\ \tau\ \sigma\ \sigma'\ \tau'$ **and** $D\ r\ \upsilon\ \sigma'\ \sigma''\ \upsilon'$
shows $(r|(\sigma\ @\ \tau'\ @\ \upsilon')|,\ r|\sigma| + r|(\tau@\upsilon)|\ ) \in$ *mul-eq r*
proof −
have *0*: $lexmax\ r\ (\sigma@\tau'@\upsilon') = lexmax\ r\ (\sigma@\tau') + (lexmax\ r\ \upsilon' -s\ dl\ r\ (\sigma@\tau'))$ (**is** *?L* = -)
unfolding *append-assoc[symmetric] lemma3-2-2* **by** *auto*
moreover have . . . = $lexmax\ r\ (\sigma@\tau') + ((lexmax\ r\ \upsilon' -s\ dl\ r\ (\sigma@\tau')) \cap s\ dl\ r\ \tau) + ((lexmax\ r\ \upsilon' -s\ dl\ r\ (\sigma@\tau')) -s\ dl\ r\ \tau)$
using *lemmaA-3-10* **unfolding** *union-assoc* **by** *auto*
moreover have (. . ., $lexmax\ r\ \sigma + lexmax\ r\ \tau + ((lexmax\ r\ \upsilon' -s\ dl\ r\ (\sigma@\tau')) -s\ dl\ r\ \tau)) \in$ *mul-eq r* (**is** (-,*?R*) ∈ -)
using *assms claim1 lemma2-6-6-a union-commute* **by** *metis*
moreover have (*?R*, $lexmax\ r\ \sigma + lexmax\ r\ \tau + (((lexmax\ r\ \upsilon' -s\ dl\ r\ \sigma') -s\ dl\ r\ \tau))) \in$ *mul-eq r* (**is** (-,*?R*) ∈ -)
using *lemma2-6-6-a[OF assms(1) claim2[OF assms(1,3)]]* **by** *auto*
moreover have (*?R*, $lexmax\ r\ \sigma + lexmax\ r\ \tau + lexmax\ r\ \upsilon -s\ dl\ r\ \tau) \in$ *mul-eq r* (**is** (-,*?R*) ∈ -)
using *lemma2-6-6-a[OF assms(1) lemma2-6-5-a'[OF assms(1) D-eq(1)[OF assms(1,2,4)]]]* **unfolding** *dl-def* **by** *auto*
moreover have $?R = lexmax\ r\ \sigma + lexmax\ r\ (\tau@\upsilon)$ **unfolding** *union-assoc lemma3-2-2* **by** *auto*
ultimately show *?thesis* **using** *mul-eq-trans[OF assms(1)]* **by** *metis*
qed

lemma *lemma3-5*: **assumes** *trans r* **and** *irefl r* **and** $D\ r\ \tau\ \sigma\ \sigma'\ \tau'$ **and** $D\ r\ \upsilon\ \sigma'\ \sigma''\ \upsilon'$
shows $D\ r\ (\tau@\upsilon)\ \sigma\ \sigma''\ (\tau'@\upsilon')$
unfolding *D-def append-assoc* **using** *assms lemma3-5-1 lemma3-5-2 union-commute* **by** *metis*

lemma *step2*: **assumes** *trans r* **and** $\tau \neq []$ **shows** $(M \cap s\ dl\ r\ \tau$,*lexmax r* $\tau) \in$ *mul r* **proof** −
from *assms* **obtain** *x xs* **where** τ=*x*#*xs* **using** *list.exhaust* **by** *auto*
hence *x*: *lexmax r* $\tau \neq \{\#\}$ **by** *auto*
from *assms*(*2*) **have** *y*: *set-mset* $(M \cap sdl\ r\ \tau) \subseteq dm\ r$ (*lexmax r* τ) **unfolding** *lemma3-2-1*[*OF assms*(*1*)] *intersect-def* **by** *auto*
show *?thesis* **using** *lemma2-6-4*[*OF assms*(*1*) *x y*] **by** *auto*
qed

Lemma 3.6

lemma *lemma3-6*: **assumes** *t*: *trans r* **and** *ne*: $\tau \neq []$ **and** *D*: *D r* τ σ σ' τ'
shows $(r|\sigma'| + r|\upsilon|,\ r|\sigma| + r|\tau@\upsilon|\ ) \in$ *mul r* (**is** (*?L*,*?R*) ∈ -) **proof** −
have *?L* = ((*lexmax r* σ' + *lexmax r* υ) ∩*s dl r* τ) + ((*lexmax r* σ' + *lexmax r* υ) −*s dl r* τ)
unfolding *lemmaA-3-10*[*symmetric*] **by** *auto*
moreover have (. . .,*lexmax r* τ + ((*lexmax r* σ' + *lexmax r* υ) −*s dl r* τ)) ∈ *mul r* (**is** (-,*?R2*) ∈ -)
using *lemma2-6-9*[*OF t step2*[*OF t ne*]] *union-commute* **by** *metis*
moreover have *?R2* = *lexmax r* τ + (*lexmax r* σ' −*s dl r* τ) + (*lexmax r* υ −*s dl r* τ)
unfolding *lemmaA-3-8 union-assoc*[*symmetric*] **by** *auto*
moreover have . . . = *lexmax r* (τ@σ') + (*lexmax r* υ −*s dl r* τ) **unfolding** *lemma3-2-2* **by** *auto*
moreover have (. . .,*lexmax r* σ + *lexmax r* τ + (*lexmax r* υ −*s dl r* τ)) ∈ *mul-eq r* (**is** (-,*?R5*) ∈ -)
using *D* **unfolding** *D-def* **using** *lemma2-6-6-a*[*OF t*] *union-commute* **by** *metis*
moreover have *?R5* = *?R* **unfolding** *lemma3-2-2 union-assoc* **by** *auto*
ultimately show *?thesis* **using** *mul-and-mul-eq-imp-mul t* **by** *metis*
qed

lemma *lemma3-6-v*: **assumes** *trans r* **and** *irrefl r* **and** $\sigma \neq []$ **and** *D r* τ σ σ' τ'
shows $(r|\tau'| + r|\upsilon|,\ r|\tau| + r|\sigma@\upsilon|\ ) \in$ *mul r*
using *assms lemma3-6 mirror-D* **by** *fast*

1.1.4 Labeled Rewriting

Theorem 3.7

type-synonym ('*a*,'*b*) *lars* = ('*a*×'*b*×'*a*) *set*
type-synonym ('*a*,'*b*) *seq* = ('*a*×('*b*×'*a*)*list*)

inductive-set *seq* :: ('*a*,'*b*) *lars* ⇒ ('*a*,'*b*) *seq set* **for** *ars*
where (*a*,[]) ∈ *seq ars*
| (*a*,α,*b*) ∈ *ars* ⟹ (*b*,*ss*) ∈ *seq ars* ⟹ (*a*,(α,*b*) # *ss*) ∈ *seq ars*

definition *lst* :: ('*a*,'*b*) *seq* ⇒ '*a*
where *lst ss* = (*if snd ss* = [] *then fst ss else snd* (*last* (*snd ss*)))

results on seqs

lemma *seq-tail1*: **assumes** *seq*: (*s*,*x*#*xs*) ∈ *seq lars*
shows (*snd x*,*xs*) ∈ *seq lars* **and** (*s*,*fst x*,*snd x*) ∈ *lars* **and** *lst* (*s*,*x*#*xs*) = *lst* (*snd x*,*xs*)
proof −
show (*snd x*,*xs*)∈ *seq lars* **using** *assms* **by** (*cases*) *auto*
next
show (*s*,*fst x*,*snd x*) ∈ *lars* **using** *assms* **by** (*cases*) *auto*
next
show *lst* (*s*,*x*#*xs*) = *lst* (*snd x*,*xs*) **using** *assms* **unfolding** *lst-def* **by** (*cases*) *auto*
qed

lemma *seq-chop*: **assumes** (*s*,*ss*@*ts*) ∈ *seq ars* **shows** (*s*,*ss*) ∈ *seq ars* (*lst*(*s*,*ss*),*ts*) ∈ *seq ars* **proof** −
show (*s*,*ss*) ∈ *seq ars* **using** *assms* **proof** (*induct ss arbitrary*: *s*)
case *Nil* **show** *?case* **using** *seq.intros*(*1*) **by** *fast*
next
case (*Cons x xs*) **hence** *k*:(*s*,*x*#(*xs*@*ts*)) ∈ *seq ars* **by** *auto*
from *Cons* **have** (*snd x*,*xs*) ∈ *seq ars* **using** *seq-tail1*(*1*) **unfolding** *append.simps* **by** *fast*
thus *?case* **using** *seq.intros*(*2*)[*OF seq-tail1*(*2*)[*OF k*]] **by** *auto*
qed
next
show (*lst*(*s*,*ss*),*ts*) ∈ *seq ars* **using** *assms* **proof** (*induct ss arbitrary*:*s*)
case *Nil* **thus** *?case* **unfolding** *lst-def* **by** *auto*
next
case (*Cons x xs*)
hence (*lst* (*snd x*,*xs*),*ts*) ∈ *seq ars* **using** *seq-tail1*(*1*) **unfolding** *append.simps* **by** *fast*
thus *?case* **unfolding** *lst-def* **by** *auto*
qed
qed

lemma *seq-concat-helper*:
assumes (*s*,*ls*) ∈ *seq ars* **and** *ss2* ∈ *seq ars* **and** *lst* (*s*,*ls*) = *fst ss2*
shows (*s*,*ls*@*snd ss2*) ∈ *seq ars* ∧ (*lst* (*s*,*ls*@*snd ss2*) = *lst ss2*)
using *assms* **proof** (*induct ls arbitrary*: *s ss2 rule*:*list.induct*)
case *Nil* **thus** *?case* **unfolding** *lst-def* **by** *auto*
next
case (*Cons x xs*)
hence (*snd x*,*xs*) ∈ *seq ars* **and** *mem*:(*s*,*fst x*,*snd x*) ∈ *ars* **and** *lst* (*snd x*,*xs*) = *fst ss2*
using *seq-tail1*[*OF Cons*(*2*)] *Cons*(*4*) **by** *auto*
thus *?case* **using** *Cons seq.intros*(*2*)[*OF mem*] **unfolding** *lst-def* **by** *auto*
qed

lemma *seq-concat*:
assumes *ss1* ∈ *seq ars* **and** *ss2* ∈ *seq ars* **and** *lst ss1* = *fst ss2*
shows (*fst ss1*,*snd ss1*@*snd ss2*) ∈ *seq ars* **and** (*lst* (*fst ss1*,*snd ss1*@*snd ss2*) = *lst ss2*)

proof −
show (*fst ss1*,*snd ss1*@*snd ss2*) ∈ *seq ars* **using** *seq-concat-helper assms* **by** *force*
next
show (*lst* (*fst ss1*,*snd ss1*@*snd ss2*) = *lst ss2*)
using *assms surjective-pairing seq-concat-helper* **by** *metis*
qed

diagrams

definition *diagram* :: ('*a*,'*b*) *lars* ⇒ ('*a*,'*b*) *seq* × ('*a*,'*b*) *seq* × ('*a*,'*b*) *seq* × ('*a*,'*b*) *seq* ⇒ *bool*
where *diagram ars d* = (*let* (τ,σ,σ′,τ′) = *d in* {σ,τ,σ′,τ′} ⊆ *seq ars* ∧
fst σ = *fst* τ ∧ *lst* σ = *fst* τ′ ∧ *lst* τ = *fst* σ′ ∧ *lst* σ′ = *lst* τ′)

definition *labels* :: ('*a*,'*b*) *seq* ⇒ '*b list*
where *labels ss* = *map fst* (*snd ss*)

definition *D2* :: '*b rel* ⇒ ('*a*,'*b*) *seq* × ('*a*,'*b*) *seq* × ('*a*,'*b*) *seq* × ('*a*,'*b*) *seq* ⇒ *bool*
where *D2 r d* = (*let* (τ,σ,σ′,τ′) = *d in D r* (*labels* τ) (*labels* σ) (*labels* σ′) (*labels* τ′))

lemma *lemma3-5-d*: **assumes** *diagram ars* (τ,σ,σ′,τ′) **and** *diagram ars* (υ,σ′,σ′′,υ′)
shows *diagram ars* ((*fst* τ,*snd* τ@*snd* υ),σ,σ′′,(*fst* τ′),*snd* τ′@*snd* υ′) **proof** −
from *assms* **have** *tau*: τ ∈ *seq ars* **and** *upsilon*: υ ∈ *seq ars* **and** *o*: *lst* τ = *fst* υ
and *tau′*: τ′ ∈ *seq ars* **and** *upsilon′*: υ′ ∈ *seq ars* **and** *l*: *lst* τ′ = *fst* υ′
unfolding *diagram-def* **by** *auto*
show *?thesis* **using** *assms seq-concat*[*OF tau′ upsilon′ l*] *seq-concat*[*OF tau upsilon o*] **unfolding** *diagram-def* **by** *auto*
qed

lemma *lemma3-5-d-v*: **assumes** *diagram ars* (τ,σ,σ′,τ′) **and** *diagram ars* (τ′,υ,υ′,τ′′)
shows *diagram ars* (τ,(*fst* σ,*snd* σ@*snd* υ),(*fst* σ′,*snd* σ′@*snd* υ′),τ′′) **proof** −
from *assms* **have** *d1*: *diagram ars* (σ,τ,τ′,σ′) **and** *d2*: *diagram ars* (υ,τ′,τ′′,υ′)
unfolding *diagram-def* **by** *auto*
show *?thesis* **using** *lemma3-5-d*[*OF d1 d2*] **unfolding** *diagram-def* **by** *auto*
qed

lemma *lemma3-5′*: **assumes** *trans r* **and** *irrefl r* **and** *D2 r* (τ,σ,σ′,τ′) **and** *D2 r* (υ,σ′,σ′′,υ′)
shows *D2 r* ((*fst* τ,*snd* τ@*snd* υ),σ,σ′′,(*fst* τ′),*snd* τ′@*snd* υ′)
using *assms lemma3-5*[*OF assms*(*1*,*2*)] **unfolding** *labels-def D2-def* **by** *auto*

lemma *lemma3-5′-v*: **assumes** *trans r* **and** *irrefl r* **and** *D2 r* (τ,σ,σ′,τ′) **and** *D2 r* (τ′,υ,υ′,τ′′)
shows *D2 r* (τ, (*fst* σ,*snd* σ@*snd* υ),(*fst* σ′,*snd* σ′@*snd* υ′),τ′′) **proof** −
from *assms*(*3*,*4*) **have** *D1*:*D2 r* (σ,τ,τ′,σ′) **and** *D2*: *D2 r* (υ,τ′,τ′′,υ′)
unfolding *D2-def* **using** *mirror-D*[*OF assms*(*1*,*2*)] **by** *auto*
show *?thesis* **using** *lemma3-5′*[*OF assms*(*1*,*2*) *D1 D2*] *mirror-D*[*OF assms*(*1*,*2*)]
unfolding *D2-def* **by** *auto*
qed

lemma *trivial-diagram*: **assumes** $\sigma \in$ *seq ars* **shows** *diagram ars* (σ,(*fst* σ,[]),(*lst* σ,[]),σ)
using *assms seq.intros(1)* **unfolding** *diagram-def Let-def lst-def* **by** *auto*

lemma *trivial-D2*: **assumes** $\sigma \in$ *seq ars* **shows** *D2 r* (σ,(*fst* σ,[]),(*lst* σ,[]),σ)
using *assms* **unfolding** *D2-def D-def labels-def* **using** *mul-eq-reflexive* **by** *auto*

definition *DD* :: ('*a*,'*b*) *lars* ⇒ '*b rel* ⇒ ('*a*,'*b*) *seq* × ('*a*,'*b*) *seq* × ('*a*,'*b*) *seq* × ('*a*,'*b*) *seq* ⇒ *bool*
where *DD ars r d* = (*diagram ars d* ∧ *D2 r d*)

lemma *lemma3-5-DD*: **assumes** *trans r* **and** *irrefl r* **and** *DD ars r* (τ,σ,σ',τ') **and** *DD ars r* (υ,σ',σ'',υ')
shows *DD ars r* ((*fst* τ,*snd* τ@*snd* υ),σ,σ'',(*fst* τ'),*snd* τ'@*snd* υ')
using *assms lemma3-5-d lemma3-5'*[*OF assms*(*1*,*2*)] **unfolding** *DD-def* **by** *fast*

lemma *lemma3-5-DD-v*: **assumes** *trans r* **and** *irrefl r* **and** *DD ars r* (τ,σ,σ',τ') **and** *DD ars r* (τ',υ,υ',τ'')
shows *DD ars r* (τ, (*fst* σ,*snd* σ@*snd* υ),(*fst* σ',*snd* σ'@*snd* υ'),τ'')
using *assms lemma3-5-d-v lemma3-5'-v* **unfolding** *DD-def* **by** *fast*

lemma *trivial-DD*: **assumes** $\sigma \in$ *seq ars* **shows** *DD ars r* (σ,(*fst* σ,[]),(*lst* σ,[]),σ)
using *assms trivial-diagram trivial-D2* **unfolding** *DD-def* **by** *fast*

lemma *mirror-DD*: **assumes** *trans r* **and** *irrefl r* **and** *DD ars r* (τ,σ,σ',τ') **shows** *DD ars r* (σ,τ,τ',σ')
using *assms mirror-D* **unfolding** *DD-def D2-def diagram-def* **by** *auto*

well-foundedness of rel r

definition *measure* :: '*b rel* ⇒ ('*a*,'*b*) *seq* × ('*a*,'*b*) *seq* ⇒ '*b multiset*
where *measure r P* = *r*|*labels* (*fst P*)| + *r*|*labels* (*snd P*)|

definition *pex* :: '*b rel* ⇒ (('*a*,'*b*) *seq* × ('*a*,'*b*) *seq*) *rel*
where *pex r* = {(*P1*,*P2*). (*measure r P1*,*measure r P2*) ∈ *mul r*}

lemma *wfi*: **assumes** *relr* = *pex r* **and** ¬ *wf* (*relr*) **shows** ¬ *wf* (*mul r*) **proof** −
have ¬ *SN* ((*relr*)$^{-1}$) **using** *assms* **unfolding** *SN-iff-wf converse-converse* **by** *auto*
from *this* **obtain** *s* **where** ∀ *i*. (*s i*, *s* (*Suc i*)) ∈ *relr*$^{-1}$ **unfolding** *SN-def SN-on-def* **by** *auto*
hence *fact*:∀ *i*. (*measure r* (*s i*), *measure r* (*s* (*Suc i*))) ∈ (*mul r*)$^{-1}$ **unfolding** *assms*(*1*) *pex-def* **by** *auto*
have ¬ *SN* ((*mul r*)$^{-1}$) **using** *chain-imp-not-SN-on*[*OF fact*] **unfolding** *SN-on-def* **by** *auto*
thus *?thesis* **unfolding** *SN-iff-wf converse-converse* **by** *auto*
qed

lemma *wf*: **assumes** *trans r* **and** *wf r* **shows** *wf* (*pex r*) **using** *wf-mul*[*OF assms*] *wfi* **by** *auto*

main result

definition *peak* :: ('*a*,'*b*) *lars* ⇒ ('*a*,'*b*) *seq* × ('*a*,'*b*) *seq* ⇒ *bool*
where *peak ars p* = (*let* (τ,σ) = *p in* {τ,σ} ⊆ *seq ars* ∧ *fst* τ = *fst* σ)

definition *local-peak* :: ('*a*,'*b*) *lars* ⇒ ('*a*,'*b*) *seq* × ('*a*,'*b*) *seq* ⇒ *bool*
where *local-peak ars p* = (*let* (τ,σ) = *p in peak ars p* ∧ *length* (*snd* τ) = *1* ∧ *length* (*snd* σ) = *1*)

proof of Theorem 3.7

lemma *LD-imp-D*: **assumes** *trans r* **and** *wf r* **and** ∀ *P*. (*local-peak ars P* ⟶ (∃ σ′ τ′. *DD ars r* (*fst P*,*snd P*,σ′,τ′)))
and *peak ars P* **shows** (∃ σ′ τ′. *DD ars r* (*fst P*,*snd P*,σ′,τ′)) **proof** −
have *i*: *irrefl r* **using** *assms*(*1*,*2*) *acyclic-irrefl trancl-id wf-acyclic* **by** *metis*
have *wf*: *wf* (*pex r*) **using** *wf*[*OF assms*(*1*,*2*)] **.**
show *?thesis* **using** *assms*(*4*) **proof** (*induct rule*:*wf-induct-rule*[*OF wf*])
case (*1 P*)
obtain *s* τ σ **where** *decompose*:*P* = (τ,σ) **and** *tau*:τ ∈ *seq ars* **and** *sigma*:σ ∈ *seq ars*
and *tau-s*: *fst* τ = *s* **and** *sigma-s*: *fst* σ = *s* **using** *1* **unfolding** *peak-def* **by** *auto*
show *?case* **proof** (*cases snd* τ)
case *Nil* **from** *mirror-DD*[*OF assms*(*1*) *i trivial-DD*[*OF sigma*]]
show *?thesis* **using** *tau-s sigma-s Nil surjective-pairing* **unfolding** *decompose fst-conv snd-conv DD-def* **by** *metis*
next
case (*Cons β-step υ-step*)
hence *tau-dec*: τ = (*s*,[*β-step*]@*υ-step*) **apply** *auto* **using** *tau-s surjective-pairing* **by** *metis*
hence *tau2*: (*s*,[*β-step*]@*υ-step*) ∈ *seq ars* **using** *tau* **by** *auto*
show *?thesis* **proof** (*cases snd* σ)
case *Nil* **from** *trivial-DD*[*OF tau*]
show *?thesis* **using** *tau-s sigma-s Nil surjective-pairing* **unfolding** *decompose fst-conv snd-conv DD-def* **by** *metis*
next
case (*Cons α-step ϱ-step*)
hence *sigma-dec*: σ = (*s*,[*α-step*]@*ϱ-step*) **apply** *auto* **using** *sigma-s surjective-pairing* **by** *metis*
hence *sigma2*:(*s*,[*α-step*]@*ϱ-step*) ∈ *seq ars* **using** *sigma* **by** *auto*

have *alpha*:(*s*,[*α-step*]) ∈ *seq ars* (**is** *?α* ∈ -)
and *rho*: (*lst* (*s*,[*α-step*]),*ϱ-step*) ∈ *seq ars* (**is** *?ϱ* ∈ -) **using** *seq-chop*[*OF sigma2*] **by** *auto*
have *beta*:(*s*,[*β-step*]) ∈ *seq ars* (**is** *?β* ∈ -)
and *upsilon*: (*lst* (*s*,[*β-step*]),*υ-step*) ∈ *seq ars* (**is** *?υ* ∈ -) **using** *seq-chop*[*OF tau2*] **by** *auto*
have *local-peak ars* (*?β*,*?α*) **using** *alpha beta* **unfolding** *local-peak-def peak-def*

by *auto*
from *this* **obtain** κ μ **where** *D*:*DD ars r* (*?β*,*?α*,κ,μ) **using** *assms*(*3*) **apply** *auto* **by** *metis*
hence *kappa*: κ∈*seq ars* **and** *mu*: μ∈*seq ars* **unfolding** *DD-def diagram-def* **by** *auto*
have *P-IH1*: *peak ars* (*?υ*,κ) **using** *upsilon kappa D* **unfolding** *DD-def diagram-def peak-def* **by** *auto*
have *beta-ne*: *labels ?β* ≠ [] **unfolding** *labels-def* **by** *auto*
have *dec*: *D r* (*labels ?β*) (*labels ?α*) (*labels* κ) (*labels* μ) **using** *D* **unfolding** *DD-def D2-def* **by** *auto*
have *x1*:((*?υ*,κ), (τ,*?α*)) ∈ *pex r* **using** *lemma3-6*[*OF assms*(*1*) *beta-ne dec*]
unfolding *pex-def measure-def decompose labels-def tau-dec* **by** (*simp add*: *add.commute*)
have (*lexmax r* (*labels* τ) + *lexmax r* (*labels* (*?α*)), *lexmax r* (*labels* τ) + *lexmax r* (*labels* σ)) ∈ *mul-eq r* (**is** (*?l*,*?r*) ∈ -)
unfolding *sigma-dec labels-def snd-conv list.map lexmax.simps diff-from-empty*
using *assms*(*1*) **by** (*simp add*: *lemma2-6-2-a*)
hence ((*?υ*,κ),*P*) ∈ *pex r* **using** *x1* **unfolding** *sigma-s pex-def measure-def decompose* **using** *mul-and-mul-eq-imp-mul*[*OF assms*(*1*)] **by** *auto*
from *this* **obtain** κ' υ' **where** *IH1*: *DD ars r* (*?υ*,κ,κ',υ') **using** *1*(*1*)[*OF* - *P-IH1*] **unfolding** *decompose* **by** *auto*
hence *kappa'*:κ'∈*seq ars* **and** *upsilon'*: υ'∈*seq ars* **using** *D* **unfolding** *DD-def diagram-def* **by** *auto*
have *tau'*: (*fst* μ,*snd* μ@(*snd* υ')) ∈ *seq ars* (**is** *?τ'* ∈ -) **using** *seq-concat*(*1*)[*OF mu upsilon'*] *D IH1* **unfolding** *DD-def diagram-def* **by** *auto*
have *DIH1*: *DD ars r* (τ,*?α*,κ',*?τ'*) **using** *lemma3-5-DD*[*OF assms*(*1*) *i D IH1*] *tau-dec* **by** *auto*
hence *dec-dih1*: *D r* (*labels* τ) (*labels ?α*) (*labels* κ') (*labels ?τ'*) **using** *DIH1* **unfolding** *DD-def D2-def* **by** *simp*

have *P-IH2*: *peak ars* (*?τ'*,*?ϱ*) **using** *tau' rho D* **unfolding** *DD-def diagram-def peak-def* **by** *auto*
have *alpha-ne*: *labels ?α* ≠ [] **unfolding** *labels-def* **by** *simp*
have ((*?τ'*,*?ϱ*),*P*) ∈ *pex r* **using** *lemma3-6-v*[*OF assms*(*1*) *i alpha-ne dec-dih1*] **unfolding** *pex-def measure-def decompose labels-def sigma-dec* **by** *auto*
from *this* **obtain** ϱ' τ'' **where** *IH2*: *DD ars r* (*?τ'*,*?ϱ*,ϱ',τ'') **using** *1*(*1*)[*OF* - *P-IH2*] **by** *auto*
show *?thesis* **using** *lemma3-5-DD-v*[*OF assms*(*1*) *i DIH1 IH2*] **unfolding** *decompose fst-conv snd-conv sigma-dec* **by** *fast*
qed
qed
qed
qed

CR with unlabeling

definition *unlabel* :: ('*a*,'*b*) *lars* ⇒ '*a rel*
where *unlabel ars* = {(*a*,*c*). ∃ *b*. (*a*,*b*,*c*) ∈ *ars*}

lemma *step-imp-seq*: **assumes** (*a*,*b*) ∈ (*unlabel ars*)

shows $\exists\, ss \in seq\ ars.\ fst\ ss = a \wedge lst\ ss = b$ **proof** −
obtain α **where** *step*:$(a,\alpha,b) \in ars$ **using** *assms* **unfolding** *unlabel-def* **by** *auto*
hence *ss*: $(a,[(\alpha,b)]) \in seq\ ars$ (**is** *?ss* ∈ -) **using** *seq.intros* **by** *fast*
have *fst ?ss* = *a* **and** *lst ?ss* = *b* **unfolding** *lst-def* **by** *auto*
thus *?thesis* **using** *ss* **unfolding** *lst-def* **by** *fast*
qed

lemma *steps-imp-seq*: **assumes** $(a,b) \in (unlabel\ ars)^\frown*$
shows $\exists\, ss \in seq\ ars.\ fst\ ss = a \wedge lst\ ss = b$ **using** *assms(1)* **proof**
fix *n* **assume** *A*: $(a,b) \in (unlabel\ ars)^\frown\frown n$ **thus** *?thesis* **proof** (*induct n arbitrary*: *a b ars*)
case *0* **hence** *eq*: *a* = *b* **by** *auto*
have $(a,[]) \in seq\ ars$ **using** *seq.intros(1)* **by** *fast*
thus *?case* **using** *fst-eqD snd-conv lst-def eq* **by** *metis*
next
case (*Suc m*)
obtain *c* **where** *steps*: $(a,c) \in (unlabel\ ars)^\frown\frown m$ **and** *step*: $(c,b) \in (unlabel\ ars)$ **using** *Suc* **by** *auto*
obtain *ss ts* **where** *ss1*: *ss* ∈ *seq ars* **and** *ss2*:*fst ss* = *a*
and *ts1*: *ts* ∈ *seq ars* **and** *ts3*:*lst ts* = *b* **and** *eq*: *lst ss* = *fst ts*
using *Suc steps step-imp-seq*[*OF step*] **by** *metis*
show *?case* **using** *seq-concat*[*OF ss1 ts1 eq*] **unfolding** *ss2 ts3* **by** *force*
qed
qed

lemma *step-imp-unlabeled-step*: **assumes** $(a,b,c) \in ars$ **shows** $(a,c) \in (unlabel\ ars)$
using *assms* **unfolding** *unlabel-def* **by** *auto*

lemma *seq-imp-steps*:
assumes *ss* ∈ *seq ars* **and** *fst ss* = *a* **and** *lst ss* = *b* **shows** $(a,b) \in (unlabel\ ars)^\frown*$
proof −
from *assms surjective-pairing* **obtain** *ls* **where** $(a,ls) \in seq\ (ars)$ **and** $lst\ (a,ls) = b$ **by** *metis*
thus *?thesis* **proof** (*induct ls arbitrary*: *a b rule*:*list.induct*)
case *Nil* **thus** *?case* **unfolding** *lst-def* **by** *auto*
next
case (*Cons x xs*)
have *fst*:$(a,fst\ x,snd\ x) \in ars$ **using** *Cons seq-tail1(2) surjective-pairing* **by** *metis*
have $(snd\ x,b) \in (unlabel\ ars)^\frown*$ **using** *Cons seq-tail1(1,3)* **by** *metis*
thus *?case* **using** *step-imp-unlabeled-step*[*OF fst*] **by** *auto*
qed
qed

lemma *seq-vs-steps*: **shows** $(a,b) \in (unlabel\ ars)^\frown* = (\exists\, ss.\ fst\ ss = a \wedge lst\ ss = b \wedge ss \in seq\ ars)$
using *seq-imp-steps steps-imp-seq* **by** *metis*

lemma *D-imp-CR*: **assumes** $\forall\, P.\ (peak\ ars\ P \longrightarrow (\exists\ \sigma'\ \tau'.\ DD\ ars\ r\ (fst\ P,snd$

P,σ',τ'))) **shows** *CR* (*unlabel ars*) **proof**
fix *a b c* **assume** *A*: (*a*,*b*) ∈ (*unlabel ars*)* **and** *B*: (*a*,*c*) ∈ (*unlabel ars*)* **show** (*b*,*c*) ∈ (*unlabel ars*)$^\downarrow$ **proof** −
obtain *ss1 ss2* **where** *peak ars* (*ss1*,*ss2*) **and** *b*: *lst ss1* = *b* **and** *c*: *lst ss2* = *c*
unfolding *peak-def* **using** *A B* **unfolding** *seq-vs-steps* **by** *auto*
from *this* **obtain** *ss3 ss4* **where** *dia*: *diagram ars* (*ss1*,*ss2*,*ss3*,*ss4*) **using** *assms*(*1*) **unfolding** *DD-def* **apply** *auto* **using** *surjective-pairing* **by** *metis*
from *dia* **obtain** *d* **where** *ss3*: *ss3* ∈ *seq ars* **and** *ss4*: *ss4* ∈ *seq ars*
and *ss3-1*: *fst ss3* = *b* **and** *ss3-2*: *lst ss3* = *d* **and** *ss4-1*: *fst ss4* = *c* **and** *ss4-2*:*lst ss4* = *d*
using *b c* **unfolding** *diagram-def* **by** *auto*
show *?thesis* **using** *seq-imp-steps*[*OF ss3 ss3-1 ss3-2*] *seq-imp-steps*[*OF ss4 ss4-1 ss4-2*] **by** *auto*
qed
qed

definition *LD* :: *'b set* ⇒ *'a rel* ⇒ *bool*
where *LD L ars* = (∃ (*r*:: (*'b rel*)) (*lrs*::(*'a*,*'b*) *lars*). (*ars* = *unlabel lrs*) ∧ *trans r* ∧ *wf r* ∧ (∀ *P*. (*local-peak lrs P* ⟶ (∃ *σ' τ'*. (*DD lrs r* (*fst P*,*snd P*,σ',τ'))))))

lemma *sound*: **assumes** *LD L ars* **shows** *CR ars*
using *assms LD-imp-D D-imp-CR* **unfolding** *LD-def* **by** *metis*

1.1.5 Application: Newman's Lemma

lemma *measure*:
assumes *lab-eq*: *lrs* = {(*a*,*c*,*b*). *c* = *a* ∧ (*a*,*b*) ∈ *ars*} **and** (*s*,(*α*,*t*) # *ss*) ∈ *seq lrs*
shows *set* (*labels* (*t*,*ss*)) ⊆ *ds* ((*ars*$^\frown$+)$^{-1}$) {*α*} **using** *assms*(*2*) **proof** (*induct ss arbitrary*: *s α t*)
case *Nil* **thus** *?case* **unfolding** *labels-def* **by** *auto*
next
case (*Cons x xs*)
from *this* **obtain** *β u* **where** *x*:*x* = (*β*,*u*) **using** *surjective-pairing* **by** *metis*
have *t*: *trans* ((*ars*$^\frown$+)$^{-1}$) **by** (*metis trans-converse trans-trancl*)
from *Cons*(*1*) *x* **have** *s0*: (*s*, *α*, *t*) ∈ *lrs* **and** *cs*:(*t*,(*β*,*u*)#*xs*) ∈ *seq lrs* **using** *Cons.prems seq-tail1*(*1*) *snd-conv fst-conv seq-tail1*(*2*) **by** *auto*
have *ih*: *set* (*labels* (*u*, *xs*)) ⊆ *ds* ((*ars*$^\frown$+)$^{-1}$) {*β*} **using** *Cons*(*1*)[*OF cs*] **by** *auto*
have *key*: {*β*} ⊆ *ds* ((*ars*$^\frown$+)$^{-1}$) {*α*} **using** *s0 cs seq-tail1*(*2*)[*OF cs*] **unfolding** *ds-def lab-eq* **by** *auto*
show *?case* **using** *ih subset-imp-ds-subset*[*OF t key*] *key* **unfolding** *x labels-def* **by** *auto*
qed

lemma *newman*: **assumes** *WCR ars* **and** *SN ars* **shows** *CR ars* **proof** −
from *assms* **obtain** *L* **where** *L* = {*a* . ∃ *b*. (*a*,*b*) ∈ *ars*} **by** *auto*
from *assms* **obtain** *lrs* **where** *lab-eq*: (*lrs* = {(*a*,*c*,*b*). *c* = *a* ∧ (*a*,*b*) ∈ *ars*}) **by** *auto*

have *lab*: *ars* = *unlabel lrs* **unfolding** *unlabel-def lab-eq* **by** *auto*

have *t*: *trans* $((ars\hat{}+)^{-1})$ **using** *trans-converse trans-trancl* **by** *auto*
have *w*: *wf* $((ars\hat{}+)^{-1})$ **using** *assms(2) wf-trancl trancl-converse* **unfolding** *SN-iff-wf* **by** *metis*
have *ps*: $\forall P.$ (*local-peak lrs P* −−> ($\exists$ σ' τ'. *DD lrs* $((ars\hat{}+)^{-1})$ (*fst P*,*snd P*,σ',τ'))) **proof**
fix *P* **show** *local-peak lrs P* −−> ($\exists$ σ' τ'. *DD lrs* $((ars\hat{}+)^{-1})$ (*fst P*,*snd P*,σ',τ')) **proof**
assume *A*: *local-peak lrs P* **show** ($\exists$ σ' τ'. *DD lrs* $((ars\hat{}+)^{-1})$ (*fst P*,*snd P*,σ',τ')) (**is** *?DD*) **proof** −
from *lab-eq* **have** *lab*: *ars* = *unlabel lrs* **unfolding** *unlabel-def* **by** *auto*
from *A* **obtain** τ σ **where** *ts*: $\{\tau,\sigma\} \subseteq$ *seq lrs* **and** *l1*: *length* (*snd* τ) = *1* **and** *l2*: *length* (*snd* σ) = *1* **and** *P*: *P* = (τ,σ)
and *p*: *fst* τ = *fst* σ **unfolding** *local-peak-def peak-def* **by** *auto*

from *l1* **obtain** β *b* **where** *1*: *snd* τ = [(β,*b*)] **by**(*auto simp add*: *length-Suc-conv*)
from *this* **obtain** *a* **where** *tau*: τ = (*a*,[(β,*b*)]) **by** (*metis surjective-pairing*)
hence *alb*: (*a*,β,*b*) ∈ *lrs* **using** *ts* **by** (*metis fst-conv insert-subset seq-tail1(2) snd-conv*)
have *ab*: (*a*,*b*) ∈ *ars* **and** *a-eq*: *a* = β **using** *alb* **unfolding** *lab-eq* **by** *auto*

from *l2* **obtain** α *c* **where** *2*: *snd* σ = [(α,*c*)] **by**(*auto simp add*: *length-Suc-conv*)
hence *sigma*: σ = (*a*,[(α,*c*)]) **using** *ts* **by** (*metis fst-conv p prod.collapse tau*)
hence *alc*: (*a*,α,*c*) ∈ *lrs* **using** *ts* **by** (*metis fst-conv insert-subset seq-tail1(2) snd-conv*)
hence *ac*: (*a*,*c*) ∈ *ars* **and** *a-eq*: *a* = α **using** *alb* **unfolding** *lab-eq* **by** *auto*

from *tau sigma* **have** *fl*: *fst* τ = *a* ∧ *fst* σ = *a* ∧ *lst* τ = *b* ∧ *lst* σ = *c* **unfolding** *lst-def* **by** *auto*
from *ab ac* **obtain** *d* **where** (*b*,*d*) ∈ $ars\hat{}*$ **and** (*c*,*d*) ∈ $ars\hat{}*$ **using** *assms(1)* **by** *auto*
from *this* **obtain** σ' τ' **where** *sigma'*: σ' ∈ *seq lrs* **and** *sigma'1*: *fst* σ' = *b* **and** *lst* σ' = *d*
and *tau'*:τ' ∈ *seq lrs* **and** *fst* τ' = *c* **and** *lst* τ' = *d* **using** *steps-imp-seq* **unfolding** *lab* **by** *metis*
hence *d*:*diagram lrs* (*fst P*, *snd P*, σ', τ') **using** *P A ts fl* **unfolding** *local-peak-def peak-def diagram-def* **by** *auto*

have *s1*:(*a*,(β,*b*)# *snd* σ') ∈ *seq lrs* **using** ‹*fst* σ' = *b*› *seq.intros(2)*[*OF alb*] *sigma'* **by** *auto*
have *vv*: *set* (*labels* σ') ⊆ *ds* $((ars\hat{}+)^{-1})$ {β} **using** *measure*[*OF lab-eq s1*] **by** (*metis* ‹*fst* σ' = *b*› *surjective-pairing*)
have *s2*:(*a*,(α,*c*)# *snd* τ') ∈ *seq lrs* **using** ‹*fst* τ' = *c*› *seq.intros(2)*[*OF alc*] *tau'* **by** *auto*
hence *ww*: *set* (*labels* τ') ⊆ *ds* $((ars\hat{}+)^{-1})$ {α} **using** *measure*[*OF lab-eq*] *s2* **by** (*metis* ‹*fst* τ' = *c*› *surjective-pairing*)
from *w* **have** *i*: *irrefl* $((ars\hat{}+)^{-1})$ **by** (*metis SN-imp-acyclic acyclic-converse acyclic-irrefl assms(2) trancl-converse*)
from *vv ww* **have** *ld*: *LD'* $((ars\hat{}+)^{-1})$ β α (*labels* σ') [] [] (*labels* τ') [] [] **unfolding** *LD'-def LD-1'-def* **by** *auto*

have *D*: *D* ((*ars*^+)$^{-1}$) (*labels* (*fst P*)) (*labels* (*snd P*)) (*labels* σ') (*labels* τ') **using** *proposition3-4*[*OF t i ld*] **unfolding** *P sigma tau lst-def labels-def* **by** *auto*

from *d D* **have** *DD lrs* ((*ars*^+)$^{-1}$) (*fst P*, *snd P*, σ', τ') **unfolding** *DD-def D2-def* **by** *auto*
thus *?thesis* **by** *fast*
qed
qed
qed
have *LD L ars* **using** *lab t w ps* **unfolding** *LD-def* **by** *fast*
thus *?thesis* **using** *sound* **by** *auto*
qed

1.2 Conversion Version

This section follows [2].

auxiliary results on multisets

lemma *mul-eq-add-right*: (*M*,*M*+*P*) ∈ *mul-eq r* **proof** −
have *M* = *M* + {#} *set-mset* {#} ⊆ *dm r P* **by** *auto*
thus *?thesis* **unfolding** *mul-eq-def* **by** *fast*
qed

lemma *mul-add-right*: **assumes** (*M*,*N*) ∈ *mul r* **shows** (*M*,*N*+*P*) ∈ *mul r* **proof** −
from *assms* **obtain** *I J K* **where** *M* = *I* + *K N* = *I* + *J set-mset K* ⊆ *dm r J J* ≠ {#} **unfolding** *mul-def* **by** *auto*
hence *b*: *M* = *I* + *K N* + *P* = *I* + (*J* + *P*) *set-mset K* ⊆ *ds r* (*set-mset J* ∪ *set-mset P*) *J*+*P* ≠ {#} **unfolding** *dm-def lemma2-6-1-set* **using** *union-assoc* **by** *auto*
hence *set-mset K* ⊆ *ds r* (*set-mset* (*J*+*P*)) **by** *auto*
thus *?thesis* **using** *b* **unfolding** *mul-def* **unfolding** *dm-def* **by** *fast*
qed

lemma *mul-eq-and-ds-imp-ds*:
assumes *t*: *trans r* **and** (*M*,*N*) ∈ *mul-eq r* **and** *set-mset N* ⊆ *ds r S*
shows *set-mset M* ⊆ *ds r S* **proof** −
from *assms* **obtain** *I J K* **where** *a*: *M* = *I* + *K* **and** *N* = *I* + *J* **and** *c*: *set-mset K* ⊆ *dm r J* **unfolding** *mul-eq-def* **by** *auto*
hence *k1*:*set-mset I* ⊆ *ds r S set-mset J* ⊆ *ds r S* **using** *assms* **by** *auto*
hence *ds r* (*set-mset J*) ⊆ *ds r S* **using** *subset-imp-ds-subset*[*OF t*] **by** *auto*
thus *?thesis* **using** *k1 a c* **unfolding** *dm-def* **by** *auto*
qed

lemma *lemma2-6-2-set*: **assumes** *S* ⊆ *T* **shows** *ds r S* ⊆ *ds r T* **using** *assms* **unfolding** *ds-def* **by** *auto*

lemma *leq-imp-subseteq*: **assumes** *M* ⊆# *N* **shows** *set-mset M* ⊆ *set-mset N* **using** *assms mset-subset-eqD* **by** *auto*

lemma *mul-add-mul-eq-imp-mul*: **assumes** $(M,N) \in mul\ r$ **and** $(P,Q) \in mul\text{-}eq\ r$ **shows** $(M+P,N+Q) \in mul\ r$ **proof** −
from *assms(1)* **obtain** *I J K* **where** *a*:$M = I + K\ N = I + J$ *set-mset* $K \subseteq dm\ r\ J\ J \neq \{\#\}$ **unfolding** *mul-def* **by** *auto*
from *assms(2)* **obtain** *I2 J2 K2* **where** *b*:$P = I2 + K2\ Q = I2 + J2$ *set-mset* $K2 \subseteq dm\ r\ J2$ **unfolding** *mul-eq-def* **by** *auto*
have $M + P = (I + I2) + (K + K2)$ **using** *a b union-commute union-assoc* **by** *metis*
moreover have $N + Q = (I + I2) + (J + J2)$ **using** *a b union-commute union-assoc* **by** *metis*
moreover have *set-mset* $(K + K2) \subseteq dm\ r\ (J + J2)$ **using** *a b* **unfolding** *lemma2-6-1-multiset* **by** *auto*
ultimately show *?thesis* **using** *a b* **unfolding** *mul-def* **by** *fast*
qed

labeled conversion

type-synonym $('a,'b)\ conv = ('a \times ((bool \times 'b \times 'a)\ list))$

inductive-set *conv* :: $('a,'b)\ lars \Rightarrow ('a,'b)\ conv\ set$ **for** *ars*
where $(a,[]) \in conv\ ars$
| $(a,\alpha,b) \in ars \Longrightarrow (b,ss) \in conv\ ars \Longrightarrow (a,(True,\alpha,b)\ \#\ ss) \in conv\ ars$
| $(b,\alpha,a) \in ars \Longrightarrow (b,ss) \in conv\ ars \Longrightarrow (a,(False,\alpha,b)\ \#\ ss) \in conv\ ars$

definition *labels-conv* :: $('a,'b)\ conv \Rightarrow 'b\ list$
where $labels\text{-}conv\ c = map\ (\lambda q.\ (fst\ (snd\ q)))\ (snd\ c)$

definition *measure-conv* :: $'b\ rel \Rightarrow ('a,'b)\ conv \Rightarrow 'b\ multiset$
where $measure\text{-}conv\ r\ c = lexmax\ r\ (labels\text{-}conv\ c)$

fun *lst-conv* :: $('a,'b)\ conv \Rightarrow 'a$
where $lst\text{-}conv\ (s,[]) = s$
| $lst\text{-}conv\ (s,(d,\alpha,t)\ \#\ ss) = lst\text{-}conv\ (t,ss)$

definition *local-diagram1* :: $('a,'b)\ lars \Rightarrow ('a,'b)\ seq \Rightarrow ('a,'b)\ seq \Rightarrow ('a,'b)\ seq \Rightarrow ('a,'b)\ seq \Rightarrow ('a,'b)\ seq \Rightarrow bool$
where $local\text{-}diagram1\ ars\ \beta\ \alpha\ \sigma1\ \sigma2\ \sigma3 =$
$(local\text{-}peak\ ars\ (\beta,\alpha) \wedge \{\sigma1,\sigma2,\sigma3\} \subseteq seq\ ars \wedge lst\ \beta = fst\ \sigma1 \wedge lst\ \sigma1 = fst\ \sigma2 \wedge lst\ \sigma2 = fst\ \sigma3)$

definition *LDD1* :: $('a,'b)\ lars \Rightarrow 'b\ rel \Rightarrow ('a,'b)\ seq \Rightarrow ('a,'b)\ seq \Rightarrow ('a,'b)\ seq \Rightarrow ('a,'b)\ seq \Rightarrow ('a,'b)\ seq \Rightarrow bool$
where $LDD1\ ars\ r\ \beta\ \alpha\ \sigma1\ \sigma2\ \sigma3 =$ $(local\text{-}diagram1\ ars\ \beta\ \alpha\ \sigma1\ \sigma2\ \sigma3 \wedge$
$LD\text{-}1'\ r\ (hd\ (labels\ \beta))\ (hd\ (labels\ \alpha))\ (labels\ \sigma1)\ (labels\ \sigma2)\ (labels\ \sigma3))$

definition *LDD* :: $('a,'b)\ lars \Rightarrow 'b\ rel \Rightarrow ('a,'b)\ seq \times ('a,'b)\ seq \times ('a,'b)\ seq \times ('a,'b)\ seq \times ('a,'b)\ seq \times ('a,'b)\ seq \times ('a,'b)\ seq \Rightarrow bool$
where $LDD\ ars\ r\ d = (let\ (\beta,\alpha,\sigma1,\sigma2,\sigma3,\tau1,\tau2,\tau3) = d\ in\ LDD1\ ars\ r\ \beta\ \alpha\ \sigma1\ \sigma2\ \sigma3 \wedge LDD1\ ars\ r\ \alpha\ \beta\ \tau1\ \tau2\ \tau3 \wedge lst\ \sigma3 = lst\ \tau3)$

definition *local-triangle1* :: ('a,'b) *lars* ⇒ ('a,'b) *seq* ⇒ ('a,'b) *seq* ⇒ ('a,'b) *conv* ⇒ ('a,'b) *seq* ⇒ ('a,'b) *conv* ⇒ *bool*
where *local-triangle1 ars β α σ1 σ2 σ3* =
(*local-peak ars* (*β*,*α*) ∧ *σ2* ∈ *seq ars* ∧ {*σ1*,*σ3*} ⊆ *conv ars* ∧ *lst β* = *fst σ1* ∧ *lst-conv σ1* = *fst σ2* ∧ *lst σ2* = *fst σ3*)

definition *LT1* :: ('a,'b) *lars* ⇒ 'b *rel* ⇒ ('a,'b) *seq* ⇒ ('a,'b) *seq* ⇒ ('a,'b) *conv* ⇒ ('a,'b) *seq* ⇒ ('a,'b) *conv* ⇒ *bool*
where *LT1 ars r β α σ1 σ2 σ3* = (*local-triangle1 ars β α σ1 σ2 σ3* ∧
LD-1' r (*hd* (*labels β*)) (*hd* (*labels α*)) (*labels-conv σ1*) (*labels σ2*) (*labels-conv σ3*))

definition *LT* :: ('a,'b) *lars* ⇒ 'b *rel* ⇒ ('a,'b) *seq* × ('a,'b) *seq* × ('a,'b) *conv* × ('a,'b) *seq* × ('a,'b) *conv* × ('a,'b) *conv* × ('a,'b) *seq* × ('a,'b) *conv* ⇒ *bool*
where *LT ars r t* = (*let* (*β*,*α*,*σ1*,*σ2*,*σ3*,*τ1*,*τ2*,*τ3*) = *t in LT1 ars r β α σ1 σ2 σ3* ∧ *LT1 ars r α β τ1 τ2 τ3* ∧ *lst-conv σ3* = *lst-conv τ3*)

lemma *conv-tail1*: **assumes** *conv*: (*s*,(*d*,*α*,*t*)#*xs*) ∈ *conv ars*
shows (*t*,*xs*) ∈ *conv ars* **and** *d* ⟹ (*s*,*α*,*t*) ∈ *ars* **and** ¬*d* ⟹ (*t*,*α*,*s*) ∈ *ars* **and** *lst-conv* (*s*,(*d*,*α*,*t*)#*xs*) = *lst-conv* (*t*,*xs*) **proof** −
show (*t*,*xs*) ∈ *conv ars* **using** *assms* **by** (*cases*) *auto*
show *d* ⟹ (*s*,*α*,*t*) ∈ *ars* **using** *assms* **by** (*cases*) *auto*
show ¬*d* ⟹ (*t*,*α*,*s*) ∈ *ars* **using** *assms* **by** (*cases*) *auto*
show *lst-conv* (*s*,(*d*,*α*,*t*)#*xs*) = *lst-conv* (*t*,*xs*) **unfolding** *lst-conv.simps* **by** *auto*
qed

lemma *conv-chop*: **assumes** (*s*,*ss1*@*ss2*) ∈ *conv ars* **shows** (*s*,*ss1*) ∈ *conv ars* (*lst-conv* (*s*,*ss1*),*ss2*) ∈ *conv ars* **proof** −
show (*s*,*ss1*) ∈ *conv ars* **using** *assms* **proof** (*induct ss1 arbitrary*: *s*)
case *Nil* **thus** *?case* **using** *conv.intros* **by** *fast*
next
case (*Cons t' ts*) **from** *this* **obtain** *d α t* **where** *dec*: *t'* = (*d*,*α*,*t*) **using** *prod-cases3* **by** *metis*
from *Cons* **have** (*s*, *t'* # *ts* @ *ss2*) ∈ *conv ars* **by** *auto*
hence (*t*, *ts* @ *ss2*) ∈ *conv ars* **and** *d1*: *d* ⟹ (*s*,*α*,*t*) ∈ *ars* **and** *d2*:¬*d* ⟹ (*t*,*α*,*s*) ∈ *ars* **using** *conv-tail1*(*1*−*3*) **unfolding** *dec* **by** *auto*
hence (*t*, *ts*) ∈ *conv ars* **using** *Cons* **by** *auto*
thus *?case* **unfolding** *dec* **using** *Cons conv.intros d1 d2* **by** (*cases d*) *auto*
qed
show (*lst-conv* (*s*,*ss1*),*ss2*) ∈ *conv ars* **using** *assms* **proof** (*induct ss1 arbitrary*:*s*)
case *Nil* **thus** *?case* **using** *conv.intros* **unfolding** *last.simps* **by** *auto*
next
case (*Cons t' ts*) **from** *this* **obtain** *d α t* **where** *dec*: *t'* = (*d*,*α*,*t*) **using** *prod-cases3* **by** *metis*
from *Cons* **have** (*s*, *t'* # *ts* @ *ss2*) ∈ *conv ars* **by** *auto*
hence (*snd* (*snd t'*), *ts* @ *ss2*) ∈ *conv ars* **using** *conv-tail1*(*1*) **unfolding** *dec* **by** *auto*
thus *?case* **using** *Cons*(*1*) **unfolding** *dec last.simps* **by** *auto*

qed
qed

lemma *conv-concat-helper*:
assumes (*s*,*ls*) ∈ *conv ars* **and** *ss2* ∈ *conv ars* **and** *lst-conv* (*s*,*ls*) = *fst ss2*
shows (*s*,*ls*@*snd ss2*) ∈ *conv ars* ∧ (*lst-conv* (*s*,*ls*@*snd ss2*) = *lst-conv ss2*)
using *assms* **proof** (*induct ls arbitrary*: *s ss2 rule*:*list.induct*)
case *Nil* **thus** *?case* **unfolding** *lst-def* **by** *auto*
next
case (*Cons x xs*) **from** *this* **obtain** *d* α *t* **where** *dec*: *x* = (*d*,α,*t*) **using** *prod-cases3*
by *metis*
hence *tl*: (*t*,*xs*) ∈ *conv ars* **and** *d1*:*d* ⟹ (*s*,α,*t*) ∈ *ars* **and** *d2*: ¬*d* ⟹ (*t*,α,*s*) ∈
ars **and** *lst*:*lst-conv* (*t*,*xs*) = *fst ss2*
using *conv-tail1 Cons*(*2*) *Cons*(*4*) **by** *auto*
have (*t*,*xs*@*snd ss2*) ∈ *conv ars* **and** *lst*: *lst-conv* (*t*,*xs*@*snd ss2*) = *lst-conv ss2*
using *Cons*(*1*)[*OF tl Cons*(*3*) *lst*] **by** *auto*
thus *?case* **using** *conv.intros d1 d2* **unfolding** *dec lst-conv.simps* **by** (*cases d*)
auto
qed

lemma *conv-concat*:
assumes *ss1* ∈ *conv ars* **and** *ss2* ∈ *conv ars* **and** *lst-conv ss1* = *fst ss2*
shows (*fst ss1*,*snd ss1*@*snd ss2*) ∈ *conv ars* **and** (*lst-conv* (*fst ss1*,*snd ss1*@*snd*
ss2) = *lst-conv ss2*)
proof −
show (*fst ss1*,*snd ss1*@*snd ss2*) ∈ *conv ars* **using** *conv-concat-helper assms* **by**
force
next
show (*lst-conv* (*fst ss1*,*snd ss1*@*snd ss2*) = *lst-conv ss2*)
using *assms surjective-pairing conv-concat-helper* **by** *metis*
qed

lemma *conv-concat-labels*:
assumes *ss1* ∈ *conv ars* **and** *ss2* ∈ *conv ars* **and** *set* (*labels-conv ss1*) ⊆ *S* **and**
set (*labels-conv ss2*) ⊆ *T*
shows *set* (*labels-conv* (*fst ss1*,*snd ss1*@*snd ss2*)) ⊆ *S* ∪ *T* **using** *assms* **unfolding**
labels-conv-def **by** *auto*

lemma *seq-decompose*:
assumes σ ∈ *seq ars* **and** *labels* σ = σ*1*′@σ*2*′
shows ∃ σ*1* σ*2*. ({σ*1*,σ*2*} ⊆ *seq ars* ∧ σ = (*fst* σ*1*,*snd* σ*1*@*snd* σ*2*) ∧ *lst* σ*1* = *fst*
σ*2* ∧ *lst* σ*2* = *lst* σ ∧ *labels* σ*1* = σ*1*′ ∧ *labels* σ*2* = σ*2*′) **proof** −
obtain *s ss* **where** σ-*dec*: σ = (*s*,*ss*) **using** *assms*(*1*) *surjective-pairing* **by** *metis*
show *?thesis* **using** *assms* **unfolding** σ-*dec* **proof** (*induct ss arbitrary*: *s* σ*1*′ σ*2*′
rule:*list.induct*)
case *Nil* **thus** *?case* **unfolding** *labels-def lst-def* **by** *auto*
next
case (*Cons x xs*)
have *step*: (*s*,*x*) ∈ *ars* **and** *x*:(*snd x*,*xs*) ∈ *seq ars* **using** *seq-tail1*[*OF Cons*(*2*)]

surjective-pairing **by** *auto*
hence *steps*: $(s,[x]) \in seq\ ars$ **by** (*metis Cons(2) append-Cons append-Nil seq-chop(1)*)
from *Cons(3)* **have** *a*:*fst x*#*labels* (*snd x*,*xs*) = $\sigma 1'$@$\sigma 2'$ **unfolding** *labels-def snd-conv* **by** *auto*
show *?case* **proof** (*cases* $\sigma 1'$=[])
case *True*
from *a True* **obtain** *l ls* **where** $\sigma 2'$*-dec*: $\sigma 2' = l\#ls$ **and** *y1*:*fst x* = *l* **and** *y2*:*labels* (*snd x*,*xs*) = []@*ls* **by** *auto*
obtain $\sigma 1\ \sigma 2$ **where** *ih*: $\sigma 1 \in seq\ ars\ \sigma 2 \in seq\ ars$ (*snd x*, *xs*) = (*fst* $\sigma 1$, *snd* $\sigma 1$ @ *snd* $\sigma 2$) *lst* $\sigma 1$ = *fst* $\sigma 2$
labels $\sigma 1$ = [] *labels* $\sigma 2$ = *ls* **using** *Cons(1)*[*OF x y2*] **by** *blast*
hence *c*:*fst* (*snd x*,*xs*) = *fst* $\sigma 1$ **by** *auto*

have *1*: $\sigma 1$ = (*snd x*,[]) **using** *ih* **unfolding** *labels-def* **apply** *auto* **by** (*metis surjective-pairing*)
hence *2*: *snd x* = *fst* $\sigma 1$ *xs* = *snd* $\sigma 2$ **using** *ih* **by** *auto*
have *3*: *snd x* = *fst* $\sigma 2$ **using** *ih 1* **unfolding** *lst-def* **by** *auto*
have $\sigma 2$ = (*snd x*,*xs*) **using** *x 1 2 3 surjective-pairing* **by** *metis*
hence *l*:*lst* (*s*, [*x*]) = *fst* $\sigma 2$ **unfolding** *lst-def* **by** *auto*
have *m*:{(*s*,[]), (*s*,*x*# *snd* $\sigma 2$)} ⊆ *seq ars* (**is** {*?*$\sigma 1$,*?*$\sigma 2$} ⊆ -) **using** *seq.intros(1) seq-concat(1)*[*OF steps - l*] *ih* **by** *auto*
moreover have (*s*, *x* # *xs*) = (*fst ?*$\sigma 1$, *snd ?*$\sigma 1$ @ *snd ?*$\sigma 2$) **using** *m 2* **by** *auto*
moreover have *lst ?*$\sigma 1$ = *fst ?*$\sigma 2$ **using** *m* **unfolding** *lst-def* **by** *auto*
moreover have *lst ?*$\sigma 2$ = *lst* (*s*,*x*#*xs*) **unfolding** *lst-def* **using** *2 3* **by** *auto*
moreover have *labels* (*s*,[]) = $\sigma 1'$ **unfolding** *labels-def* **using** *True* **by** *auto*
moreover have *labels ?*$\sigma 2$ = $\sigma 2'$ **using** *ih y1* **unfolding** $\sigma 2'$*-dec labels-def* **by** *auto*
ultimately show *?thesis* **by** *metis*
next
case *False* **from** *False* **obtain** *l ls* **where** $\sigma 1'$*-dec*:$\sigma 1' = l\#ls$ **using** *list.exhaust* **by** *auto*
hence *y1*:*fst x* = *l* **and** *y2*:*labels* (*snd x*,*xs*) = *ls* @ $\sigma 2'$ **using** *a* **by** *auto*
obtain $\sigma 1\ \sigma 2$ **where** *ih*: $\sigma 1 \in seq\ ars\ \sigma 2 \in seq\ ars$ (*snd x*, *xs*) = (*fst* $\sigma 1$, *snd* $\sigma 1$ @ *snd* $\sigma 2$)
lst $\sigma 1$ = *fst* $\sigma 2$ *lst* $\sigma 2$ = *lst* (*snd x*, *xs*) *labels* $\sigma 1$ = *ls labels* $\sigma 2$ = $\sigma 2'$ **using** *Cons(1)*[*OF x y2*] **by** *blast*
hence {(*s*,*x*# *snd* $\sigma 1$),$\sigma 2$} ⊆ *seq ars* (**is** {*?*$\sigma 1$,-} ⊆ *seq ars*) **using** *seq-concat(1)*[*OF steps*] *ih* **unfolding** *lst-def* **by** *auto*
moreover have (*s*,*x*#*xs*) = (*fst ?*$\sigma 1$,*snd ?*$\sigma 1$@*snd* $\sigma 2$) **using** *ih* **by** *auto*
moreover have *lst ?*$\sigma 1$ = *fst* $\sigma 2$ **using** *ih* **unfolding** *lst-def* **by** *auto*
moreover have *lst* $\sigma 2$ = *lst* (*s*, *x* # *xs*) **using** *ih* **unfolding** *lst-def* **by** *auto*
moreover have *labels ?*$\sigma 1$ = $\sigma 1'$ **using** *ih* $\sigma 1'$*-dec y1* **unfolding** *labels-def* **by** *auto*
moreover have *labels* $\sigma 2$ = $\sigma 2'$ **using** *ih* **by** *auto*
ultimately show *?thesis* **by** *blast*
qed
qed
qed

lemma *seq-imp-conv*:
assumes (*s*,*ss*) ∈ *seq ars*
shows (*s*,*map* (λ*step*. (*True*,*step*)) *ss*) ∈ *conv ars* ∧
lst-conv (*s*,*map* (λ*step*.(*True*,*step*)) *ss*) = *lst* (*s*,*ss*) ∧
labels (*s*,*ss*) = *labels-conv* (*s*,*map* (λ*step*.(*True*,*step*)) *ss*)
using *assms* **proof** (*induct ss arbitrary*: *s rule*:*list.induct*)
case *Nil* **show** *?case* **unfolding** *lst-def labels-def labels-conv-def* **apply** *auto*
using *conv.intros* **by** *fast*
next
case (*Cons t' ts*) **have** *t'-dec*: *t'* = (*fst t'*,*snd t'*) **using** *surjective-pairing* **by** *auto*
have *step*: (*s*,*fst t'*,*snd t'*) ∈ *ars* **and** *x*:(*snd t'*,*ts*) ∈ *seq ars* **using** *seq-tail1*[*OF Cons*(*2*)] **by** *auto*
have *y1*:(*snd t'*, *map* (*Pair True*) *ts*) ∈ *conv ars* **and**
y2: *lst* (*snd t'*, *ts*) = *lst-conv* (*snd t'*, *map* (*Pair True*) *ts*) **and**
y3: *labels* (*snd t'*, *ts*) = *labels-conv* (*snd t'*, *map* (*Pair True*) *ts*) **using** *Cons*(*1*)[*OF x*] **by** *auto*
have *k*: (*s*,(*True*,*fst t'*,*snd t'*)#*map* (*Pair True*) *ts*) ∈ *conv ars* **using** *step y1 conv.intros* **by** *fast*
moreover have *lst* (*s*,(*fst t'*,*snd t'*)#*ts*) = *lst-conv* (*s*, *map* (*Pair True*) ((*fst t'*,*snd t'*)#*ts*)) **using** *y2* **unfolding** *list.map lst-def lst-conv.simps* **by** *auto*
moreover have *labels* (*s*,(*fst t'*,*snd t'*)#*ts*) = *labels-conv* (*s*,*map* (*Pair True*) ((*fst t'*,*snd t'*)#*ts*)) **using** *y3* **unfolding** *list.map labels-def labels-conv-def* **by** *auto*
ultimately show *?case* **by** *auto*
qed

fun *conv-mirror* :: ('*a*,'*b*) *conv* ⇒ ('*a*,'*b*) *conv*
where *conv-mirror* σ = (*let* (*s*,*ss*) = σ *in case ss of*
[] ⇒ (*s*,*ss*)
| *x*#*xs* ⇒ *let* (*d*,α,*t*) = *x in*
(*fst* (*conv-mirror* (*t*,*xs*)),*snd* (*conv-mirror* (*t*,*xs*))@[(¬*d*,α,*s*)]))

lemma *conv-mirror*: **assumes** σ ∈ *conv ars*
shows *conv-mirror* σ ∈ *conv ars* ∧
set (*labels-conv* (*conv-mirror* σ)) = *set* (*labels-conv* σ) ∧
fst σ = *lst-conv* (*conv-mirror* σ) ∧
lst-conv σ = *fst* (*conv-mirror* σ) **proof** −
from *assms* **obtain** *s ss* **where** σ*-dec*: σ = (*s*,*ss*) **using** *surjective-pairing* **by** *metis*
show *?thesis* **using** *assms* **unfolding** σ*-dec* **proof** (*induct ss arbitrary*:*s rule*:*list.induct*)
case *Nil* **thus** *?case* **using** *conv.intros conv-mirror.simps* **by** *auto*
next
case (*Cons t' ts*) **from** *this* **obtain** *d* α *t* **where** *t'-dec*: *t'* = (*d*,α,*t*) **by** (*metis prod-cases3*)
have *1*:(*t*, *ts*) ∈ *conv ars* **and** *2*: *d* ⟹ (*s*,α,*t*) ∈ *ars* **and** *3*: ¬*d* ⟹ (*t*,α,*s*) ∈ *ars* **and** *4*:*lst-conv* (*s*,*t'*#*ts*) = *lst-conv* (*t*,*ts*)
using *Cons*(*2*) *conv-tail1* **unfolding** *t'-dec* **by** *auto*
have *r*: (*t*,[(¬*d*,α,*s*)]) ∈ *conv ars* **using** *2 3 conv.intros*(*3*)[*OF - conv.intros*(*1*)] *conv.intros*(*2*)[*OF - conv.intros*(*1*)] **by** (*cases d*) *auto*
have *conv-mirror* (*s*,*t'*#*ts*) ∈ *conv ars* **using** *conv-concat*[*OF - r*] *Cons*(*1*)[*OF*

1] *t′-dec* **by** *auto*
moreover have *set* (*labels-conv* (*conv-mirror* (*s*, *t′* # *ts*))) = *set* (*labels-conv* (*s*, *t′* # *ts*)) **using** *Cons(1)*[*OF 1*] **unfolding** *labels-conv-def t′-dec* **by** *auto*
moreover have *fst* (*s*, *t′* # *ts*) = *lst-conv* (*conv-mirror* (*s*, *t′* # *ts*)) **using** *t′-dec Cons(1)*[*OF 1*] *conv-concat(2)*[*OF - r*] **by** *auto*
moreover have *lst-conv* (*s*, *t′* # *ts*) = *fst* (*conv-mirror* (*s*, *t′* # *ts*)) **using** *t′-dec 4 Cons(1)*[*OF 1*] **by** *auto*
ultimately show *?case* **by** *auto*
qed
qed

lemma *DD-subset-helper*:
assumes *t*:*trans r* **and** (*r*|*τ*@*σ′*|, *r*|*τ*| + *r*|*σ*|) ∈ *mul-eq r* **and** *set-mset* (*r*|*τ*| + *r*|*σ*|) ⊆ *ds r S*
shows *set-mset r*|*σ′*| ⊆ *ds r S* **proof** −
from *assms* **have** *d*: (*r*|*τ*| + *r*|*σ′*| −*s dl r* (*τ*) , *r*|*τ*| + *r*|*σ*|) ∈ *mul-eq r* **unfolding** *lemma3-2-2* **by** *auto*
from *assms* **have** *assm*:*set-mset* (*r*|*τ*| + *r*|*σ*|) ⊆ *ds r S* **unfolding** *measure-def* **by** *auto*
hence *tau*:*ds r* (*set-mset r*|*τ*|) ⊆ *ds r S* **using** *subset-imp-ds-subset*[*OF t*] **by** *auto*
have *set-mset* (*r*|*τ*| + (*r*|*σ′*| −*s dl r τ*)) ⊆ *ds r S* **using** *mul-eq-and-ds-imp-ds*[*OF t d assm*] **by** *auto*
hence *set-mset* (*r*|*σ′*| −*s ds r* (*set τ*)) ⊆ *ds r S* **unfolding** *dl-def* **by** *auto*
hence *set-mset* (*r*|*σ′*| −*s ds r* (*set τ*)) ∪ *ds r* (*set τ*) ⊆ *ds r S* **using** *tau* **by** (*metis t dl-def dm-def le-sup-iff lemma3-2-1*)
thus *?thesis* **unfolding** *diff-def* **by** *auto*
qed

lemma *DD-subset-ds*:
assumes *t*:*trans r* **and** *DD*: *DD ars r* (*τ*,*σ*,*σ′*,*τ′*) **and** *set-mset* (*measure r* (*τ*,*σ*)) ⊆ *ds r S* **shows** *set-mset* (*measure r* (*σ′*,*τ′*)) ⊆ *ds r S* **proof** −
have *d1*:(*r*|*labels τ* @ *labels σ′*|, *r*|*labels τ*| + *r*|*labels σ*|) ∈ *mul-eq r* **using** *DD* **unfolding** *DD-def D2-def D-def* **by** *auto*
have *d2*:(*r*|*labels σ* @ *labels τ′*|, *r*|*labels σ*| + *r*|*labels τ*|) ∈ *mul-eq r* **using** *DD* **unfolding** *DD-def D2-def D-def*
by (*auto simp*: *union-commute*)
show *?thesis* **using** *DD-subset-helper*[*OF t d1*] *DD-subset-helper*[*OF t d2*] *assms(3)* **unfolding** *measure-def* **by** *auto*
qed

lemma *conv-imp-valley*:
assumes *t*: *trans r*
and *IH*: !!*y* . ((*y*,((*s*,[*α-step*]@*ϱ-step*),(*s*,[*β-step*]@*υ-step*))) ∈ *pex r* ⟹ *peak ars y* ⟹ ∃*σ′ τ′*. *DD ars r* (*fst y*, *snd y*, *σ′*, *τ′*)) (**is** !!*y*. ((*y*,*?P*) ∈ - ⟹ - ⟹ -))
and *δ1* ∈ *conv ars*
and *set-mset* (*measure-conv r δ1*) ⊆ *dm r M*
and (*M*,{#*fst α-step*,*fst β-step*#}) ∈ *mul-eq r*
shows ∃ *σ τ*. ({*σ*,*τ*} ⊆ *seq ars* ∧ *fst σ* = *fst δ1* ∧ *fst τ* = *lst-conv δ1* ∧ *lst σ* =

lst $\tau \wedge$ *set-mset* (*measure r* (σ,τ)) $\subseteq$ *dm r M*) **proof** −
from *assms* **obtain** *s ss* **where** *sigma1*: $\delta 1 = (s,ss)$ **using** *surjective-pairing* **by** *metis*
show *?thesis* **using** *assms*(*3*,*4*) **unfolding** *sigma1* **proof** (*induct ss arbitrary*: *s rule*:*list.induct*)
case *Nil*
hence (*s*,[]) ∈ *seq ars* **using** *seq.intros*(*1*) **by** *fast*
moreover have *set-mset* (*measure r* ((*s*,[]),(*s*,[]))) ⊆ *dm r M* **unfolding** *measure-def labels-def* **by** *auto*
ultimately show *?case* **by** *auto*
next
case (*Cons t' ts*) **obtain** *d* β *t* **where** *dec*: $t' = (d,\beta,t)$ **using** *surjective-pairing* **by** *metis*
hence *dic*: {β} ⊆ *ds r* (*set-mset M*) **using** *Cons*(*3*) **unfolding** *dec measure-conv-def labels-conv-def dm-def* **by** *auto*
have *one*:*ds r* {β} ⊆ *dm r M* **unfolding** *dm-def* **using** *subset-imp-ds-subset*[*OF t dic*] **by** *auto*
have *set-mset* (*measure-conv r* (*t*,*ts*) −*s ds r* {β}) ⊆ *dm r M* **using** *Cons*(*3*) **unfolding** *measure-conv-def labels-conv-def dec* **by** *auto*
hence *set-mset* (*measure-conv r* (*t*,*ts*)) ⊆ *dm r M* ∪ *ds r* {β} **unfolding** *set-mset-def diff-def* **by** *auto*
hence *ts2*: *set-mset* (*measure-conv r* (*t*,*ts*)) ⊆ *dm r M* **using** *dic one* **by** *auto*
from *Cons*(*2*) **have** *ts*: (*t*,*ts*) ∈ *conv ars* **unfolding** *dec* **using** *conv-tail1*(*1*) **by** *fast*
from *Cons*(*1*)[*OF ts ts2*] **obtain** σ' τ **where**
ih:{σ', τ} ⊆ *seq ars* ∧ *fst* σ' = *fst* (*t*,*ts*) ∧ *fst* τ = *lst-conv* (*t*, *ts*) ∧ *lst* σ' = *lst* τ ∧ *set-mset* (*measure r* (σ',τ)) ⊆ *dm r M* **by** *metis*
have *diff*:!!*x*. *x* ∈# *r*|*map fst* (*snd* σ')|−*sds r* {β} ⟹ *x* ∈# *r*|*map fst* (*snd* σ')|
by *simp*
show *?case* **proof** (*cases d*)
case *True* **hence** *step*:(*s*,β,*t*) ∈ *ars* **using** *conv-tail1*(*2*) *Cons*(*2*) **unfolding** *dec* **by** *auto*
have (*s*,(β,*t*)# *snd* σ') ∈ *seq ars* (**is** *?*σ ∈ -) **using** *seq.intros*(*2*)[*OF step*] **using** *ih*(*1*) **by** *auto*
hence {*?*σ,τ} ⊆ *seq ars* **using** *ih* **by** *auto*
moreover have (*fst ?*σ) = *fst* (*s*, *t'* # *ts*) **by** *auto*
moreover have *fst* τ = *lst-conv* (*s*, *t'* # *ts*) **using** *ih* **unfolding** *dec lst-conv.simps* **by** *auto*
moreover have *lst ?*σ = *lst* τ **by** (*metis* ‹(*s*, (β, *t*) # *snd* σ') ∈ *seq ars*› *fst-conv ih seq-tail1*(*3*) *snd-conv surjective-pairing*)
moreover have *set-mset* (*measure r* (*?*σ,τ)) ⊆ *dm r M* **using** *diff ih dic* **unfolding** *measure-def labels-def dm-def* **by** *auto*
ultimately show *?thesis* **by** *blast*
next
case *False* **hence** *step*:(*t*,β,*s*) ∈ *ars* **using** *conv-tail1*(*3*) *Cons*(*2*) **unfolding** *dec* **by** *auto*
hence (*t*,[(β,*s*)]) ∈ *seq ars* **using** *seq.intros* **by** *metis*
hence *p*:*peak ars* ((*t*,[(β,*s*)]),σ') (**is** *peak ars ?y*) **using** *seq.intros* **unfolding** *peak-def* **using** *ih* **by** *auto*

hence *mp*: *set-mset* (*measure r ?y*) ⊆ *ds r* (*set-mset M*) **using** *ih dic* **unfolding** *measure-def labels-def dm-def*
by *simp*
hence *ne*: *M* ≠ {#} **using** *dec dic* **unfolding** *dm-def ds-def* **by** *auto*
hence *x*:(*measure r ?y*,*M*) ∈ *mul r* **using** *mp* **unfolding** *dm-def mul-def* **apply** *auto* **by** (*metis add-0*)
have ({#*fst* α*-step*#}+{#*fst* β*-step*#},*measure r ?P*) ∈ *mul-eq r* **unfolding** *assms*(*2*) *measure-def labels-def* **apply** *auto*
unfolding *union-lcomm union-assoc*[*symmetric*] **using** *mul-eq-add-right*[**where** *M*={#*fst* α*-step*#}+{#*fst* β*-step*#}] **unfolding** *union-assoc* **by** *auto*
hence (*M*,*measure r ?P*) ∈ *mul-eq r* **using** *assms*(*5*) *mul-eq-trans t* **by** (*auto simp*: *add-mset-commute*)
hence *w*:(*?y*,*?P*) ∈ *pex r* **unfolding** *assms*(*1*) *pex-def* **using** *mul-and-mul-eq-imp-mul*[*OF t x*] **by** *auto*
obtain σ′′ τ′′ **where** *DD*:*DD ars r* ((*t*,[(β,*s*)]),σ′,σ′′,τ′′) **using** *IH*[*OF w p*] **by** *auto*
have *sigma*: σ′′ ∈ *seq ars fst* σ′′ = *fst* (*s*, *t*′ # *ts*) *lst* σ′′ = *lst* τ′′ **using** *DD* **unfolding** *DD-def diagram-def lst-def* **by** *auto*
have *tau*′′: τ′′ ∈ *seq ars* **and** *eq*: *lst* τ = *fst* τ′′ **using** *DD* **unfolding** *DD-def diagram-def* **using** *ih* **by** *auto*
have *tau*:(*fst* τ,*snd* τ @ *snd* τ′′) ∈ *seq ars* (**is** *?*τ′′′ ∈ -) **and** *lst* τ′′ = *lst* (*fst* τ,*snd* τ@ *snd* τ′′) **using** *seq-concat*[*OF* - *tau*′′ *eq*] *ih* **by** *auto*
hence *tau2*: *fst ?*τ′′′ = *lst-conv* (*s*,*t*′#*ts*) *lst* σ′′ = *lst ?*τ′′′ **using** *DD ih* **unfolding** *DD-def diagram-def dec lst-conv.simps* **by** *auto*
have *set-mset* (*measure r* (σ′′,τ′′)) ⊆ *ds r* (*set-mset M*) **using** *DD-subset-ds*[*OF t DD mp*] **unfolding** *measure-def* **by** *auto*
hence *set-mset* (*r*|*labels* σ′′| + *r*|*labels* τ| + (*r*|*labels* τ′′| −*s* (*dl r* (*labels* τ)))) ⊆ *dm r M* **using** *ih* **unfolding** *measure-def dm-def diff-def* **by** *auto*
hence *fin*: *set-mset* (*measure r* (σ′′,*?*τ′′′)) ⊆ *dm r M* **unfolding** *measure-def labels-def* **apply** *auto* **unfolding** *lemma3-2-2* **by** *auto*
show *?thesis* **using** *sigma tau tau2 fin* **by** *blast*
qed
qed
qed

lemma *labels-multiset*: **assumes** *length* (*labels* σ) ≤ *1* **and** *set* (*labels* σ) ⊆ {α} **shows** (*r*|*labels* σ|, {#α#}) ∈ *mul-eq r* **proof** (*cases snd* σ)
case *Nil* **hence** *r*|*labels* σ| = {#} **unfolding** *labels-def* **by** *auto*
thus *?thesis* **unfolding** *mul-eq-def* **by** *auto*
next
case (*Cons x xs*) **hence** *l*:*length* (*labels* σ) = *1* **using** *assms*(*1*) **unfolding** *labels-def* **by** *auto*
from *this* **have** *labels* σ ≠ [] **by** *auto*
from *this* **obtain** *a as* **where** *labels* σ = *a*#*as* **using** *neq-Nil-conv* **by** *metis*
hence *leq*: *labels* σ = [*a*] **using** *l* **by** *auto*
hence *set* (*labels* σ) = {α} **using** *assms*(*2*) **by** *auto*
hence (*r*|*labels* σ|) = {#α#} **unfolding** *leq lexmax.simps diff-def* **by** *auto*
thus *?thesis* **using** *mul-eq-reflexive* **by** *auto*
qed

lemma *decreasing-imp-local-decreasing*:
assumes *t*:*trans r* **and** *i*:*irefl r* **and** *DD*: *DD ars r* (*τ*,*σ*,*σ′*,*τ′*) **and** *set* (*labels τ*) ⊆ *ds r* {*β*}
and *length* (*labels σ*) ≤ *1* **and** *set* (*labels σ*) ⊆ {*α*}
shows ∃*σ1 σ2 σ3*. (*σ′*=(*fst σ1*,*snd σ1*@*snd σ2*@*snd σ3*) ∧ *lst σ1*=*fst σ2* ∧ *lst σ2* = *fst σ3* ∧ *lst σ3* = *lst σ′*
∧ *LD-1′ r β α* (*labels σ1*) (*labels σ2*) (*labels σ3*))
set (*labels τ′*) ⊆ *ds r* ({*α*,*β*})
proof −
show ∃ *σ1 σ2 σ3*. (*σ′* = (*fst σ1*,*snd σ1*@*snd σ2*@*snd σ3*) ∧
lst σ1 = *fst σ2* ∧ *lst σ2* = *fst σ3* ∧ *lst σ3* = *lst σ′* ∧ *LD-1′ r β α* (*labels σ1*) (*labels σ2*) (*labels σ3*)) **proof** −
from *DD* **have** *σ′*:*σ′* ∈ *seq ars* **using** *assms* **unfolding** *DD-def diagram-def* **by** *auto*
from *DD* **have** *x*: (*r*|*labels σ′*| −*s dl r* (*labels τ*),*r*|*labels σ*|) ∈ *mul-eq r* **unfolding** *DD-def D2-def* **using** *D-eq*(*2*)[*OF t i*] **by** *auto*
have *dl r* (*labels τ*) ⊆ *ds r* (*ds r* {*β*}) **using** *assms*(*4*) **unfolding** *dl-def ds-def* **by** *auto*
hence *dl r* (*labels τ*) ⊆ *ds r* {*β*} **using** *ds-ds-subseteq-ds*[*OF t*] **by** *auto*
hence *x*:(*r*|*labels σ′*| −*s ds r* {*β*},*r*|*labels σ*|) ∈ *mul-eq r* **using** *x* **unfolding** *diff-def* **by** (*metis diff-def lemma2-6-8 mul-eq-trans t*)
hence *x*:(*r*|*labels σ′*| −*s dl r* [*β*],{#*α*#}) ∈ *mul-eq r* **using** *labels-multiset*[*OF assms*(*5*,*6*)] **unfolding** *dl-def* **using** *mul-eq-trans*[*OF t x*] **by** *auto*
obtain *σ1′ σ2′ σ3′* **where** *l*:*labels σ′* = *σ1′*@(*σ2′*@*σ3′*) **and** *σ1′l*: *set σ1′* ⊆ *ds r* {*β*} **and**
σ2′l: *length σ2′* ≤ *1* ∧ *set σ2′* ⊆ {*α*} **and** *σ3′l*: *set σ3′* ⊆ *ds r* {*α*,*β*} **using** *proposition3-4-inv-lists*[*OF t i x*] **unfolding** *dl-def* **by** *auto*
obtain *σ1 σ23* **where** *σ1*:*σ1* ∈ *seq ars* **and** *σ23*: *σ23* ∈ *seq ars* **and** *lf1*: *lst σ1* = *fst σ23* **and** *lf1b*: *lst σ′* = *lst σ23* **and**
σ′-eq:*σ′* = (*fst σ1*,*snd σ1*@*snd σ23*) **and** *σ1l*:*labels σ1* = *σ1′* **and** *l2*:*labels σ23* = *σ2′*@*σ3′*
using *seq-decompose*[*OF σ′ l*] **by** *auto*
obtain *σ2 σ3* **where** *σ2*:*σ2* ∈ *seq ars* **and** *σ3*:*σ3* ∈ *seq ars* **and** *lf2*:*lst σ2* = *fst σ3* **and** *lf2b*:*lst σ23* = *lst σ3* **and**
σ23-eq:*σ23* = (*fst σ2*,*snd σ2*@*snd σ3*) **and** *σ2l*: *labels σ2* = *σ2′* **and** *σ3l*: *labels σ3* = *σ3′*
using *seq-decompose*[*OF σ23 l2*] **by** *auto*
have *σ′* = (*fst σ1*, *snd σ1* @ *snd σ2* @ *snd σ3*) **using** *σ1 σ2 σ3 σ′-eq σ23-eq* **by** *auto*
moreover have *lst σ1* = *fst σ2* **using** *lf1 σ23-eq* **by** *auto*
moreover have *lst σ2* = *fst σ3* **using** *lf2* **by** *auto*
moreover have *lst σ3* = *lst σ′* **using** *lf1b lf2b* **by** *auto*
moreover have *set* (*labels σ1*) ⊆ *ds r* {*β*} **using** *σ1l σ1′l* **by** *auto*
moreover have *length* (*labels σ2*) ≤ *1* ∧ *set* (*labels σ2*) ⊆ {*α*} **using** *σ2l σ2′l* **by** *auto*
moreover have *set* (*labels σ3*) ⊆ *ds r* {*α*, *β*} **using** *σ3l σ3′l* **by** *auto*
ultimately show *?thesis* **unfolding** *LD-1′-def* **by** *fast*
qed

show *set* (*labels* τ') $\subseteq$ *ds r* ({α,β}) **proof** −
have *x*:(*r*|*labels* τ'| −*s dl r* (*labels* σ),*r*|*labels* τ|) ∈ *mul-eq r* **using** *DD D-eq*[*OF t i*] **unfolding** *DD-def D2-def* **by** *auto*
have *y*:*set-mset r*|*labels* τ| ⊆ *ds r* {β} **using** *leq-imp-subseteq*[*OF lexmax-le-multiset*[*OF t*]] *assms*(*4*) **by** *auto*
hence *set-mset* (*r*|*labels* τ'|−*s ds r* (*set* (*labels* σ))) ⊆ *ds r* {β} **using** *mul-eq-and-ds-imp-ds*[*OF t x y*] **unfolding** *dl-def* **by** *auto*
hence *set-mset* (*r*|*labels* τ'|) ⊆ *ds r* {β} ∪ *ds r* (*set* (*labels* σ)) **unfolding** *diff-def* **by** *auto*
hence *set-mset* (*r*|*labels* τ'|) ⊆ *ds r* {α,β} **using** *assms*(*6*) **unfolding** *ds-def* **by** *auto*
thus *?thesis* **using** *lexmax-set*[*OF t*] **by** *auto*
qed
qed

lemma *local-decreasing-extended-imp-decreasing*:
assumes *LT1 ars r* (*s*,[β-*step*]) (*s*,[α-*step*]) γ*1* γ*2* γ*3*
and *t*: *trans r* **and** *i*: *irrefl r*
and *IH*:!!*y* . ((*y*,((*s*,[β-*step*]@υ-*step*),(*s*,[α-*step*]@ϱ-*step*))) ∈ *pex r* ⟹ *peak ars y* ⟹ ∃σ' τ'. *DD ars r* (*fst y*, *snd y*, σ', τ')) (**is** !!*y*. ((*y*,*?P*) ∈ - ⟹ - ⟹ -))
shows ∃ σ*1* σ*2* σ*3'* γ*1'''*. ({σ*1*,σ*2*,σ*3'*,γ*1'''*} ⊆ *seq ars* ∧
set (*labels* σ*1*) ⊆ *ds r* {*fst* β-*step*} ∧ *length* (*labels* σ*2*) ≤ *1* ∧ *set* (*labels* σ*2*) ⊆ {*fst* α-*step*} ∧ *set* (*labels* σ*3'*) ⊆ *ds r* {*fst* α-*step*,*fst* β-*step*}) ∧
set (*labels* γ*1'''*) ⊆ *ds r* {*fst* α-*step*,*fst* β-*step*} ∧
snd β-*step* = *fst* σ*1* ∧ *lst* σ*1* = *fst* σ*2* ∧ *lst* σ*2* = *fst* σ*3'* ∧ *lst* σ*3'* = *lst* γ*1'''* ∧ *fst* γ*1'''* = *fst* γ*3*
proof −
from *assms labels-multiset* **have** *s2*:(*r*|*labels* γ*2*|,{#*fst* α-*step*#}) ∈ *mul-eq r* **unfolding** *LT1-def local-triangle1-def LD-1'-def labels-def* **by** *auto*
from *assms* **have** γ*1*: γ*1* ∈ *conv ars* **and** γ*3*: γ*3* ∈ *conv ars* **and** γ*2-l*: *length* (*labels* γ*2*) ≤ *1*
and γ*2-s*: *set* (*labels* γ*2*) ⊆ {*fst* α-*step*} **and** γ*3-s*: *set* (*labels-conv* γ*3*) ⊆ *ds r* {*fst* α-*step*,*fst* β-*step*}
and *set* (*labels-conv* γ*1*) ⊆ *ds r* {*fst* β-*step*} **unfolding** *LT-def LD'-def LT1-def LD-1'-def labels-def local-triangle1-def* **by** *auto*
hence *set-mset* (*measure-conv r* γ*1*) ⊆ *ds r* {*fst* β-*step*} **unfolding** *measure-conv-def* **using** *lexmax-le-multiset*[*OF t*] **by** (*metis set-mset-mset submultiset-implies-subset subset-trans*)
hence γ*1-s*: *set-mset* (*measure-conv r* γ*1*) ⊆ *dm r* {#*fst* β-*step*#} **unfolding** *dm-def* **by** *auto*
have *x*: ({#*fst* β-*step*#}, {#*fst* β-*step*, *fst* α-*step*#}) ∈ *mul-eq r* **using** *mul-eq-add-right*[*of* {#-#}] **by** *auto*
obtain γ*1'* γ*1''* **where** γ*1'*: γ*1'* ∈ *seq ars* **and** γ*1''*: γ*1''* ∈ *seq ars* **and** *eqx*:*fst* γ*1'* = *fst* γ*1*
and *fst* γ*1''* = *lst-conv* γ*1* **and** γ*1'-eq*: *lst* γ*1'* = *lst* γ*1''* **and** *m2*: *set-mset* (*measure r* (γ*1'*,γ*1''*)) ⊆ *dm r* {#*fst* β-*step*#}
using *conv-imp-valley*[*OF t IH* γ*1* γ*1-s x*] **apply** *auto* **by** *fast*
hence *Q*:*peak ars* (γ*1''*,γ*2*) (**is** *peak ars ?Q*) **unfolding** *peak-def* **using** *assms*(*1*) **unfolding** *LT-def LD'-def LT1-def local-triangle1-def* **by** *auto*

from *m2* **have** *γ1′s*: *set* (*labels γ1′*) ⊆ *ds r* {*fst β-step*} **and** *γ1′′s*: *set* (*labels γ1′′*) ⊆ *ds r* {*fst β-step*} **unfolding** *measure-def dm-def* **using** *lexmax-set*[*OF t*] **by** *auto*
from *m2* **have** *τ1′-m*:(*r*|*labels γ1′′*|,{#*fst β-step*#}) ∈ *mul r* **unfolding** *measure-def mul-def* **apply** *auto* **by** (*metis dm-def empty-neutral*(*1*) *set-mset-single add-mset-not-empty*)
hence *y*:(*r*|*labels γ1′′*| + *r*|*labels γ2*|,{#*fst α-step*#}+{#*fst β-step*#}) ∈ *mul r* **using** *mul-add-mul-eq-imp-mul*[*OF τ1′-m s2*] *union-commute* **by** *metis*
have (*r*|*labels γ1′′*| + *r*|*labels γ2*|, {#*fst α-step*,*fst β-step*#} + (*r*|*map fst ϱ-step*|−*sds r* {*fst α-step*} + *r*|*map fst υ-step*|−*sds r* {*fst β-step*})) ∈ *mul r* **using** *mul-add-right*[*OF y*] **by** (*auto simp*: *add-mset-commute*)
hence *q*:(*?Q*,*?P*) ∈ *pex r* **unfolding** *pex-def measure-def labels-def* **apply** *auto* **by** (*metis union-assoc union-commute union-lcomm*)
obtain *γ1′′′ σ′* **where** *DD*:*DD ars r* (*γ1′′*,*γ2*,*σ′*,*γ1′′′*) **using** *IH*[*OF q Q*] **by** *auto*
from *DD* **have** *σ′*: *σ′* ∈ *seq ars* **and** *γ1′′′*:*γ1′′′* ∈ *seq ars* **unfolding** *DD-def diagram-def* **by** *auto*
from *decreasing-imp-local-decreasing*[*OF t i DD γ1′′s γ2-l γ2-s*] **obtain** *σ1′ σ2′ σ3′* **where** *σ′-dec*: *σ′* = (*fst σ1′*, *snd σ1′* @ *snd σ2′* @ *snd σ3′*)
and *eq1*: *lst σ1′* = *fst σ2′ lst σ2′* = *fst σ3′ lst σ3′* = *lst σ′*
and *σ1s*: *set* (*labels σ1′*) ⊆ *ds r* {*fst β-step*} **and** *σ2l*: *length* (*labels σ2′*) ≤ *1* **and** *σ2′s*: *set* (*labels σ2′*) ⊆ {*fst α-step*} **and** *set* (*labels σ3′*) ⊆ *ds r* {*fst α-step*,*fst β-step*}
and *σ3′s*: *set* (*labels σ3′*) ⊆ *ds r* {*fst α-step*,*fst β-step*} **and** *γ1′′′s*: *set* (*labels γ1′′′*) ⊆ *ds r* {*fst α-step*,*fst β-step*} **unfolding** *LD-1′-def* **by** *blast*
have *σ′-ds*: (*fst σ1′*, *snd σ1′* @ *snd σ2′* @ *snd σ3′*) ∈ *seq ars* **using** *σ′ σ′-dec* **by** *auto*
have *σ1′*:*σ1′* ∈ *seq ars* **and** *tmp*: (*fst σ2′*,*snd σ2′*@*snd σ3′*) ∈ *seq ars* **using** *seq-chop*[*OF σ′-ds*] *surjective-pairing eq1* **by** *auto*
have *σ2′*: *σ2′* ∈ *seq ars* **and** *σ3′*: *σ3′* ∈ *seq ars* **using** *seq-chop*[*OF tmp*] **using** *surjective-pairing eq1* **by** *auto*
have *eq*:*lst γ1′* = *fst σ1′* **using** *DD γ1′-eq* **unfolding** *DD-def diagram-def* **apply** *auto* **unfolding** *σ′-dec* **by** *auto*
have *σ1*: (*fst γ1′*,*snd γ1′*@*snd σ1′*) ∈ *seq ars* (**is** *?σ1* ∈ -) **and** *eq0*:*lst* (*fst γ1′*, *snd γ1′* @ *snd σ1′*) = *lst σ1′* **using** *seq-concat*[*OF γ1′ σ1′ eq*] **by** *auto*
moreover have *set* (*labels ?σ1*) ⊆ *ds r* {*fst β-step*} **using** *σ1s γ1′s* **unfolding** *labels-def dm-def* **by** *auto*
moreover have *snd β-step* = *fst ?σ1* **using** *assms*(*1*) **unfolding** *LT1-def local-triangle1-def lst-def σ′-dec eqx* **by** *auto*
moreover have *lst ?σ1* = *fst σ2′* **unfolding** *eq0 eq1* **by** *auto*
moreover have *lst σ3′* = *lst γ1′′′* **unfolding** *eq1* **using** *DD* **unfolding** *DD-def diagram-def* **by** *auto*
moreover have *fst γ1′′′* = *fst γ3* **using** *DD assms*(*1*) **unfolding** *DD-def diagram-def LT1-def local-triangle1-def* **by** *auto*
ultimately show *?thesis* **using** *σ2′ σ2′s σ3′ σ3′s γ1′′′ γ1′′′s eq1 surjective-pairing σ2l* **by** *blast*
qed

lemma *LDD-imp-DD*:
assumes *t*:*trans r* **and** *i*:*irrefl r* **and** *LDD ars r* (*τ*,*σ*,*σ1*,*σ2*,*σ3*,*τ1*,*τ2*,*τ3*)

shows $\exists\ \sigma'\ \tau'.$ *DD ars r* $(\tau,\sigma,\sigma',\tau')$ **proof** −
from *assms* **have** *length* (*labels* σ) = *1* **unfolding** *LDD-def LDD1-def local-diagram1-def local-peak-def* **unfolding** *labels-def* **by** *auto*
from *this* **obtain** α **where** *l*: *labels* $\sigma = [\alpha]$ **by** (*metis append-Nil append-butlast-last-id butlast-conv-take diff-self-eq-0 length-0-conv take-0 zero-neq-one*)
hence *σl*: *labels* σ = [*hd* (*labels* σ)] **by** *auto*
from *assms* **have** *length* (*labels* τ) = *1* **unfolding** *LDD-def LDD1-def local-diagram1-def local-peak-def* **unfolding** *labels-def* **by** *auto*
from *this* **obtain** β **where** *l*: *labels* $\tau = [\beta]$ **by** (*metis append-Nil append-butlast-last-id butlast-conv-take diff-self-eq-0 length-0-conv take-0 zero-neq-one*)
hence *τl*: *labels* τ = [*hd* (*labels* τ)] **by** *auto*
from *assms* **have** σ':(*fst σ1*,*snd σ1*@*snd σ2*@*snd σ3*) ∈ *seq ars* (**is** *?σ′* ∈ -) **and** τ':(*fst τ1*,*snd τ1*@*snd τ2*@*snd τ3*) ∈ *seq ars* (**is** *?τ′* ∈ -)
unfolding *LDD-def LDD1-def local-diagram1-def local-peak-def peak-def* **apply** *auto* **apply** (*metis fst-eqD seq-concat*(*1*) *snd-eqD*) **by** (*metis fst-eqD seq-concat*(*1*) *snd-eqD*)
from *assms* **have** *sigmas*: *fst ?σ′* = *fst σ1 lst ?σ′* = *lst σ3* **unfolding** *LDD-def LDD1-def local-diagram1-def local-peak-def peak-def* **apply** *auto* **by** (*metis* (*hide-lams*, *no-types*) *σ′ append-assoc seq-chop*(*1*) *seq-concat*(*2*) *seq-concat-helper*)
from *assms* **have** *taus*: *fst ?τ′* = *fst τ1 lst ?τ′* = *lst τ3* **unfolding** *LDD-def LDD1-def local-diagram1-def local-peak-def peak-def* **apply** *auto* **by** (*metis* (*hide-lams*, *no-types*) *τ′ append-assoc seq-chop*(*1*) *seq-concat*(*2*) *seq-concat-helper*)
have *diagram ars* $(\tau,\sigma,?\sigma',?\tau')$ **using** *assms* **unfolding** *LDD-def LDD1-def local-diagram1-def diagram-def local-peak-def* **apply** *auto*
unfolding *peak-def* **apply** *auto* **using** *sigmas taus σ′ τ′* **by** *auto*
moreover have *D2 r* $(\tau,\sigma,?\sigma',?\tau')$ **using** *assms proposition3-4*[*OF t i*] *σl τl* **unfolding** *LDD-def LDD1-def D2-def LD′-def labels-def* **by** *auto*
ultimately show *?thesis* **unfolding** *DD-def* **by** *auto*
qed

lemma *LT-imp-DD*:
assumes *t*:*trans r*
and *i*:*irrefl r*
and *IH*:!!*y* . ((*y*,((*s*,[*β-step*]@*υ-step*),(*s*,[*α-step*]@*ϱ-step*))) ∈ *pex r* ⟹ *peak ars y* ⟹ ∃*σ′ τ′*. *DD ars r* (*fst y*, *snd y*, *σ′*, *τ′*)) (**is** !!*y*. ((*y*,*?P*) ∈ - ⟹ - ⟹ -))
and *LT*: *LT ars r* ((*s*,[*β-step*]),(*s*,[*α-step*]),*γ1*,*γ2*,*γ3*,*δ1*,*δ2*,*δ3*)
shows ∃ *κ μ*. *DD ars r* ((*s*,[*β-step*]),(*s*,[*α-step*]),*κ*,*μ*)
proof −
from *LT* **have** *LTa*: *LT1 ars r* (*s*,[*β-step*]) (*s*,[*α-step*]) *γ1 γ2 γ3* **and** *LTb*: *LT1 ars r* (*s*,[*α-step*]) (*s*,[*β-step*]) *δ1 δ2 δ3* **unfolding** *LT-def* **by** *auto*

from *local-decreasing-extended-imp-decreasing*[*OF LTa t i IH*] **obtain** *σ1 σ2 σ3′ γ1‴* **where** *sigmas*:{*σ1*,*σ2*,*σ3′*,*γ1‴*} ⊆ *seq ars* **and**
onetwo1: *set* (*labels σ1*) ⊆ *ds r* {*fst β-step*} ∧ *length* (*labels σ2*) ≤ *1* ∧ *set* (*labels σ2*) ⊆ {*fst α-step*} ∧
set (*labels σ3′*) ⊆ *ds r* {*fst α-step*, *fst β-step*} ∧ *set* (*labels γ1‴*) ⊆ *ds r* {*fst α-step*,*fst β-step*} ∧
snd β-step = *fst σ1* ∧ *lst σ1* = *fst σ2* ∧ *lst σ2* = *fst σ3′* ∧ *lst σ3′* = *lst γ1‴* ∧ *fst γ1‴* = *fst γ3* **by** *metis*

have *ih2*: !!y. (y, (s, [α-*step*] @ ϱ-*step*),(s,[β-*step*] @ υ-*step*)) $\in$ *pex r* $\Longrightarrow$ *peak ars y* $\Longrightarrow$ $\exists \sigma'\ \tau'$. *DD ars r* (*fst y*, *snd y*, σ', τ')
using *IH* **unfolding** *pex-def measure-def* **by** (*auto simp*: *union-commute*)

from *local-decreasing-extended-imp-decreasing*[*OF LTb t i ih2*] **obtain** $\tau 1\ \tau 2\ \tau 3'\ \delta 1'''$ **where** *taus*:{$\tau 1$,$\tau 2$,$\tau 3'$,$\delta 1'''$} $\subseteq$ *seq ars* **and**
onetwo2: *set* (*labels* $\tau 1$) $\subseteq$ *ds r* {*fst* α-*step*} $\wedge$ *length* (*labels* $\tau 2$) $\leq$ 1 $\wedge$ *set* (*labels* $\tau 2$) $\subseteq$ {*fst* β-*step*} $\wedge$
set (*labels* $\tau 3'$) $\subseteq$ *ds r* {*fst* β-*step*,*fst* α-*step*} $\wedge$ *set* (*labels* $\delta 1'''$) $\subseteq$ *ds r* {*fst* β-*step*,*fst* α-*step*} $\wedge$
snd α-*step* = *fst* $\tau 1$ $\wedge$ *lst* $\tau 1$ = *fst* $\tau 2$ $\wedge$ *lst* $\tau 2$ = *fst* $\tau 3'$ $\wedge$ *lst* $\tau 3'$ = *lst* $\delta 1'''$ $\wedge$ *fst* $\delta 1'''$ = *fst* $\delta 3$ **by** *metis*

have $\gamma 3$: $\gamma 3 \in$ *conv ars* **and** $\gamma 3$*m*:*set* (*labels-conv* $\gamma 3$) $\subseteq$ *ds r* {*fst* α-*step*,*fst* β-*step*} **and**
$\delta 3$: $\delta 3 \in$ *conv ars* (**is** *?c2* $\in$ -) **and** $\delta 3$*m*: *set* (*labels-conv* $\delta 3$) $\subseteq$ *ds r* {*fst* β-*step*,*fst* α-*step*} **and**
eq: *lst-conv* $\gamma 3$ = *lst-conv* $\delta 3$ **using** *LT* **unfolding** *LT-def LT1-def local-triangle1-def LD-1'-def labels-def* **by** *auto*
hence $\delta 3$*m*: *set* (*labels-conv* $\delta 3$) $\subseteq$ *ds r* {*fst* α-*step*, *fst* β-*step*} **using** *insert-commute* **by** *metis*

have $\delta 1'''$: $\delta 1''' \in$ *seq ars* **using** *taus* **by** *auto*
have $\gamma 1'''$: $\gamma 1''' \in$ *seq ars* **using** *sigmas* **by** *auto*
have *c1*: (*fst* $\delta 1'''$,*map* (*Pair True*) (*snd* $\delta 1'''$)) $\in$ *conv ars* (**is** *?c0* $\in$ -)
using *seq-imp-conv* $\delta 1'''$ *surjective-pairing* **by** *metis*
have *c11*: *lst* $\delta 1'''$ = *lst-conv ?c0* **using** *seq-imp-conv* $\delta 1'''$ *surjective-pairing* **by** *metis*
have *c1l*: *set* (*labels-conv ?c0*) $\subseteq$ *ds r* {*fst* β-*step*,*fst* α-*step*} **using** *onetwo2 seq-imp-conv* $\delta 1'''$ *surjective-pairing* **by** *metis*
hence *c1l*:*set* (*labels-conv ?c0*) $\subseteq$ *ds r* {*fst* α-*step*,*fst* β-*step*} **using** *insert-commute* **by** *metis*

have *c1*: *conv-mirror ?c0* $\in$ *conv ars* (**is** *?c1* $\in$ -)
set (*labels-conv* (*conv-mirror ?c0*)) $\subseteq$ *ds r* {*fst* α-*step*,*fst* β-*step*}
fst (*conv-mirror ?c0*) = *lst* $\tau 3'$
lst-conv (*conv-mirror ?c0*) = *fst* $\delta 3$
using *conv-mirror*[*OF c1*] *c11 c1l c1 onetwo2* **by** *auto*

have *c2*: *?c2* $\in$ *conv ars*
set (*labels-conv ?c2*) $\subseteq$ *ds r* {*fst* α-*step*, *fst* β-*step*}
fst ?c2 = *fst* $\delta 3$
lst-conv ?c2 = *lst-conv* $\gamma 3$ **using** $\delta 3$ *eq* $\delta 3$*m* **by** *auto*

have *conv-mirror* $\gamma 3 \in$ *conv ars* (**is** *?c3* $\in$ -) *set* (*labels-conv* (*conv-mirror* $\gamma 3$)) = *set* (*labels-conv* $\gamma 3$) **using** *conv-mirror*[*OF* $\gamma 3$] **by** *auto*

hence *c3*: *?c3 ∈ conv ars*
set (labels-conv ?c3) ⊆ ds r {fst α-step, fst β-step}
fst ?c3 = lst-conv δ3
lst-conv ?c3 = fst γ1‴ **using** *conv-mirror[OF γ3] eq onetwo1 γ3m* **by** *auto*

have *c4*: *(fst γ1‴,map (Pair True) (snd γ1‴)) ∈ conv ars* (**is** *?c4 ∈ -*) *set (labels-conv (fst γ1‴,map (Pair True) (snd γ1‴))) = set (labels γ1‴)*
using *seq-imp-conv γ1‴ surjective-pairing* **apply** *metis* **using** *seq-imp-conv γ1‴ surjective-pairing* **by** *metis*
have *c4*: *?c4 ∈ conv ars*
lst-conv ?c4 = lst γ1‴
set (labels-conv ?c4) ⊆ ds r {fst α-step,fst β-step}
using *seq-imp-conv γ1‴ surjective-pairing* **apply** *metis* **using** *seq-imp-conv γ1‴ surjective-pairing* **apply** *metis* **using** *c4(2) onetwo1* **by** *auto*

have *eq*: *lst-conv ?c1 = fst ?c2* **using** *c1 c2* **by** *auto*
have *c12*: *(fst ?c1,snd ?c1@snd ?c2) ∈ conv ars* (**is** *?c12 ∈ -*)
fst (fst ?c1,snd ?c1@snd ?c2) = fst ?c1 lst-conv (fst ?c1,snd ?c1@snd ?c2) = lst-conv ?c2 **using** *conv-concat[OF c1(1) c2(1) eq]* **by** *auto*
have *eq*: *lst-conv ?c12 = fst ?c3* **using** *c12 c3* **by** *auto*
have *c123*: *(fst ?c12,snd ?c12@snd ?c3) ∈ conv ars* (**is** *?c123 ∈ -*)
fst (fst ?c12,snd ?c12@snd ?c3) = fst ?c12 lst-conv (fst ?c12,snd ?c12@snd ?c3) = lst-conv ?c3 **using** *conv-concat[OF c12(1) c3(1) eq]* **by** *auto*

have *eq*: *lst-conv ?c123 = fst ?c4* **using** *c123 c2 c4 onetwo1 onetwo2* **apply** *auto* **unfolding** *Let-def* **using** *c3(4)* **apply** *auto* **by** *metis*
have *c1234*: *(fst ?c123,snd ?c123@snd ?c4) ∈ conv ars* (**is** *?c1234 ∈ -*)
fst (fst ?c123,snd ?c123@snd ?c4) = fst ?c123 lst-conv (fst ?c123,snd ?c123@snd ?c4) = lst-conv ?c4 **using** *conv-concat[OF c123(1) c4(1) eq]* **by** *auto*
hence *c1234*: *?c1234 ∈ conv ars* (**is** *?c1234 ∈ -*)
fst (?c1234) = lst τ3′ lst-conv ?c1234 = lst σ3′ **apply** *auto* **unfolding** *Let-def* **using** *c1(3)* **apply** *auto* **apply** *metis* **using** *c4(2) onetwo1* **by** *metis*

have *c12l*: *set (labels-conv ?c12) ⊆ ds r {fst α-step, fst β-step}* **using** *conv-concat-labels[OF c1(1) c2(1)] c1 c2* **by** *auto*
have *c123l*: *set (labels-conv ?c123) ⊆ ds r {fst α-step,fst β-step}* **using** *conv-concat-labels[OF c12(1) c3(1)] c12l c3* **by** *auto*
have *c1234l*:*set (labels-conv ?c1234) ⊆ ds r {fst α-step,fst β-step}* **using** *conv-concat-labels[OF c123(1) c4(1)] c123l c4* **by** *auto*
hence *set-mset (r|labels-conv ?c1234|) ⊆ ds r {fst α-step,fst β-step}* **using** *submultiset-implies-subset[OF lexmax-le-multiset[OF t]]* **by** *auto*
hence *set-mset (measure-conv r ?c1234) ⊆ dm r {#fst β-step, fst α-step#}* **unfolding** *measure-conv-def dm-def* **by** (*auto simp*: *add-mset-commute*)
hence *m*: *set-mset (measure-conv r ?c1234) ⊆ dm r {#fst α-step, fst β-step#}* **by** (*auto simp*: *add-mset-commute*)

from *c1234 m* **obtain** *ϱ* **where** *ϱ*:*ϱ ∈ conv ars* **and** *ϱm*:*set-mset (measure-conv r ϱ) ⊆ dm r {# fst α-step, fst β-step#}*

and *eq1*: *fst* ϱ = *lst* $\tau 3'$ **and** *eq2*: *lst-conv* ϱ = *lst* $\sigma 3'$ **by** *auto*

have *M*:({#*fst* α-*step*,*fst* β-*step*#},{#*fst* β-*step*,*fst* α-*step*#}) ∈ *mul-eq r* **using** *mul-eq-reflexive add-mset-commute* **by** *metis*
from *conv-imp-valley*[*OF t IH* ϱ *ϱm M*] **obtain** $\tau 3''$ $\sigma 3''$ **where**
$\tau 3''$:$\tau 3''$ ∈ *seq ars* **and** $\sigma 3''$: $\sigma 3''$ ∈ *seq ars* **and** *eq*:*fst* $\tau 3''$ = *fst* ϱ ∧ *fst* $\sigma 3''$ = *lst-conv* ϱ ∧ *lst* $\tau 3''$ = *lst* $\sigma 3''$ ∧
set-mset (*measure r* ($\tau 3''$, $\sigma 3''$)) ⊆ *dm r* {#*fst* α-*step*, *fst* β-*step*#} **apply** *auto* **by** *fast*
have *s1*: *set* (*labels* $\sigma 3''$) ⊆ *ds r* {*fst* α-*step*, *fst* β-*step*} **using** *eq* **unfolding** *dm-def measure-def* **apply** *auto* **by** (*metis* (*hide-lams*, *no-types*) *insert-commute lexmax-set subsetD t*)
have *s2*: *set* (*labels* $\tau 3''$) ⊆ *ds r* {*fst* β-*step*, *fst* α-*step*} **using** *eq* **unfolding** *dm-def measure-def* **apply** *auto* **by** (*metis* (*hide-lams*, *no-types*) *insert-commute lexmax-set subsetD t*)
have $\sigma 1$-*eq*: *lst* (*s*, [β-*step*]) = *fst* $\sigma 1$ **and** $\tau 1$-*eq*: *lst* (*s*, [α-*step*]) = *fst* $\tau 1$ **using** *onetwo1 onetwo2 surjective-pairing* **unfolding** *lst-def* **by** *auto*
have *eqn*: *lst* $\tau 3''$ = *lst* $\sigma 3''$ **and** σ-*eq*: *lst* $\sigma 3'$ = *fst* $\sigma 3''$ **and** τ-*eq*: *lst* $\tau 3'$ = *fst* $\tau 3''$ **using** *eq eq1 eq2* **by** *auto*
have $\sigma 3'$:(*fst* $\sigma 3'$,*snd* $\sigma 3'$@*snd* $\sigma 3''$) ∈ *seq ars* (**is** *?*$\sigma 3$ ∈ -) **using** *seq-concat*[*OF* - $\sigma 3''$ σ-*eq*] *sigmas* **by** *blast*
have $\tau 3'$:(*fst* $\tau 3'$,*snd* $\tau 3'$@*snd* $\tau 3''$) ∈ *seq ars* (**is** *?*$\tau 3$ ∈ -) **using** *seq-concat*[*OF* - $\tau 3''$ τ-*eq*] *taus* **by** *blast*
have *fst ?*$\sigma 3$ = *lst* $\sigma 2$ **and** *fst ?*$\tau 3$ = *lst* $\tau 2$ **using** *onetwo1 onetwo2* **by** *auto*
have *lst*:*lst ?*$\sigma 3$ = *lst ?*$\tau 3$ **using** *eqn* $\sigma 3''$ $\sigma 3'$ σ-*eq* $\tau 3''$ τ-*eq* $\tau 3'$ *seq-chop*(*1*) *seq-concat*(*2*) *surjective-pairing* **by** *metis*
have *local-diagram1 ars* (*s*, [β-*step*]) (*s*, [α-*step*]) $\sigma 1$ $\sigma 2$ (*fst* $\sigma 3'$, *snd* $\sigma 3'$ @ *snd* $\sigma 3''$)
using *sigmas* $\sigma 3'$ *onetwo1* $\sigma 1$-*eq LT* **unfolding** *local-diagram1-def LT-def LT1-def local-triangle1-def* **by** *auto*
moreover have *local-diagram1 ars* (*s*,[α-*step*]) (*s*,[β-*step*]) $\tau 1$ $\tau 2$ (*fst* $\tau 3'$,*snd* $\tau 3'$@*snd* $\tau 3''$)
using *taus* $\tau 3'$ *onetwo2* $\tau 1$-*eq LT* **unfolding** *local-diagram1-def LT-def LT1-def local-triangle1-def* **by** *auto*
moreover have *LD-1′ r* (*hd* (*labels* (*s*, [β-*step*]))) (*hd* (*labels* (*s*, [α-*step*]))) (*labels* $\sigma 1$) (*labels* $\sigma 2$) (*labels* (*fst* $\sigma 3'$, *snd* $\sigma 3'$ @ *snd* $\sigma 3''$))
using *onetwo1 s1* **unfolding** *LD-1′-def labels-def* **by** *auto*
moreover have *LD-1′ r* (*hd* (*labels* (*s*, [α-*step*]))) (*hd* (*labels* (*s*, [β-*step*]))) (*labels* $\tau 1$) (*labels* $\tau 2$) (*labels* (*fst* $\tau 3'$, *snd* $\tau 3'$ @ *snd* $\tau 3''$))
using *onetwo2 s2* **unfolding** *LD-1′-def labels-def* **by** *auto*
ultimately have *LDD*: *LDD ars r* ((*s*,[β-*step*]),(*s*,[α-*step*]),$\sigma 1$,$\sigma 2$,*?*$\sigma 3$,$\tau 1$,$\tau 2$,*?*$\tau 3$) **using** *lst* **unfolding** *LDD-def LDD1-def local-diagram1-def* **by** *auto*
from *LDD-imp-DD*[*OF t i LDD*] **show** *?thesis* **by** *auto*
qed

lemma *LT-imp-D*: **assumes** *t*:*trans r* **and** *wf r* **and** ∀ *p*. (*local-peak ars p* ⟶ (∃ $\gamma 1$ $\gamma 2$ $\gamma 3$ $\delta 1$ $\delta 2$ $\delta 3$. *LT ars r* (*fst p*,*snd p*,$\gamma 1$,$\gamma 2$,$\gamma 3$,$\delta 1$,$\delta 2$,$\delta 3$)))
and *peak ars P* **shows** (∃ σ' τ'. *DD ars r* (*fst P*,*snd P*,σ',τ')) **proof** −
have *i*: *irrefl r* **using** *assms*(*1*,*2*) *acyclic-irrefl trancl-id wf-acyclic* **by** *metis*

have *wf*: *wf* (*pex r*) **using** *wf*[*OF assms*(*1*,*2*)] **.**
show *?thesis* **using** *assms*(*4*) **proof** (*induct rule*:*wf-induct-rule*[*OF wf*])
case (*1 P*)
obtain *s* τ σ **where** *decompose*:$P = (\tau,\sigma)$ **and** *tau*:$\tau \in$ *seq ars* **and** *sigma*:$\sigma \in$ *seq ars*
and *tau-s*: *fst* $\tau = s$ **and** *sigma-s*: *fst* $\sigma = s$ **using** *1* **unfolding** *peak-def* **by** *auto*
show *?case* **proof** (*cases snd* τ)
case *Nil* **from** *mirror-DD*[*OF assms*(*1*) *i trivial-DD*[*OF sigma*]]
show *?thesis* **using** *tau-s sigma-s Nil surjective-pairing* **unfolding** *decompose fst-conv snd-conv DD-def* **by** *metis*
next
case (*Cons* β*-step* υ*-step*)
hence *tau-dec*: $\tau = (s,[\beta\text{-}step]@\upsilon\text{-}step)$ **apply** *auto* **using** *tau-s surjective-pairing* **by** *metis*
hence *tau2*: $(s,[\beta\text{-}step]@\upsilon\text{-}step) \in$ *seq ars* **using** *tau* **by** *auto*
show *?thesis* **proof** (*cases snd* σ)
case *Nil* **from** *trivial-DD*[*OF tau*]
show *?thesis* **using** *tau-s sigma-s Nil surjective-pairing* **unfolding** *decompose fst-conv snd-conv DD-def* **by** *metis*
next
case (*Cons* α*-step* ϱ*-step*)
hence *sigma-dec*: $\sigma = (s,[\alpha\text{-}step]@\varrho\text{-}step)$ **apply** *auto* **using** *sigma-s surjective-pairing* **by** *metis*
hence *sigma2*:$(s,[\alpha\text{-}step]@\varrho\text{-}step) \in$ *seq ars* **using** *sigma* **by** *auto*

have *alpha*:$(s,[\alpha\text{-}step]) \in$ *seq ars* (**is** *?*$\alpha \in$ -)
and *rho*: (*lst* $(s,[\alpha\text{-}step]),\varrho\text{-}step) \in$ *seq ars* (**is** *?*$\varrho \in$ -) **using** *seq-chop*[*OF sigma2*] **by** *auto*
hence *alpha'*: $(s,$*fst* α*-step*, *snd* α*-step*$) \in$ *ars* **by** (*metis seq-tail1*(*2*))
have *beta*:$(s,[\beta\text{-}step]) \in$ *seq ars* (**is** *?*$\beta \in$ -)
and *upsilon*: (*lst* $(s,[\beta\text{-}step]),\upsilon\text{-}step) \in$ *seq ars* (**is** *?*$\upsilon \in$ -) **using** *seq-chop*[*OF tau2*] **by** *auto*

have *lp*:*local-peak ars* (*?*β,*?*α) **using** *alpha beta* **unfolding** *local-peak-def peak-def* **by** *auto*

from *this* **obtain** $\gamma 1\ \gamma 2\ \gamma 3\ \delta 1\ \delta 2\ \delta 3$ **where** *LT*: *LT ars r* (*?*β, *?*α, $\gamma 1$, $\gamma 2$, $\gamma 3$, $\delta 1$, $\delta 2$, $\delta 3$) **using** *assms*(*3*) **apply** *auto* **by** *metis*
have *P*:$P = ((s,[\beta\text{-}step]@\upsilon\text{-}step),(s,[\alpha\text{-}step]@\varrho\text{-}step))$ (**is** $P =$ *?P*) **using** *decompose* **unfolding** *tau-dec sigma-dec* **by** *auto*
obtain κ μ **where** *D*:*DD ars r* (*?*β,*?*α,κ,μ) **using** *LT-imp-DD*[*OF t i 1*(*1*) *LT*] **unfolding** *P* **by** *fast*

hence *kappa*: $\kappa \in$*seq ars* **and** *mu*: $\mu \in$*seq ars* **unfolding** *DD-def diagram-def* **by** *auto*
have *P-IH1*: *peak ars* (*?*υ,κ) **using** *upsilon kappa D* **unfolding** *DD-def*

diagram-def peak-def **by** *auto*
have *beta-ne*: *labels ?β* ≠ [] **unfolding** *labels-def* **by** *auto*
have *dec*: *D r* (*labels ?β*) (*labels ?α*) (*labels κ*) (*labels μ*) **using** *D* **unfolding** *DD-def D2-def* **by** *auto*
have *x1*:((*?υ*,*κ*), (*τ*,*?α*)) ∈ *pex r* **using** *lemma3-6*[*OF assms*(*1*) *beta-ne dec*]
unfolding *pex-def measure-def decompose labels-def tau-dec* **apply** (*auto simp*: *add-mset-commute*) **using** *union-commute* **by** *metis*
have (*lexmax r* (*labels τ*) + *lexmax r* (*labels* (*?α*)), *lexmax r* (*labels τ*) + *lexmax r* (*labels σ*)) ∈ *mul-eq r* (**is** (*?l*,*?r*) ∈ -)
unfolding *sigma-dec labels-def snd-conv list.map lexmax.simps diff-from-empty* **by** (*simp add*: *lemma2-6-2-a t*)
hence ((*?υ*,*κ*),*P*) ∈ *pex r* **using** *x1* **unfolding** *sigma-s pex-def measure-def decompose* **using** *mul-and-mul-eq-imp-mul*[*OF assms*(*1*)] **by** *auto*
from *this* **obtain** *κ′ υ′* **where** *IH1*: *DD ars r* (*?υ*,*κ*,*κ′*,*υ′*) **using** *1*(*1*)[*OF* - *P-IH1*] **unfolding** *decompose* **by** *auto*
hence *kappa′*:*κ′*∈*seq ars* **and** *upsilon′*: *υ′*∈*seq ars* **using** *D* **unfolding** *DD-def diagram-def* **by** *auto*
have *tau′*: (*fst μ*,*snd μ*@(*snd υ′*)) ∈ *seq ars* (**is** *?τ′* ∈ -) **using** *seq-concat*(*1*)[*OF mu upsilon′*] *D IH1* **unfolding** *DD-def diagram-def* **by** *auto*
have *DIH1*: *DD ars r* (*τ*,*?α*,*κ′*,*?τ′*) **using** *lemma3-5-DD*[*OF assms*(*1*) *i D IH1*] *tau-dec* **by** *auto*
hence *dec-dih1*: *D r* (*labels τ*) (*labels ?α*) (*labels κ′*) (*labels ?τ′*) **using** *DIH1* **unfolding** *DD-def D2-def* **by** *simp*

have *P-IH2*: *peak ars* (*?τ′*,*?ϱ*) **using** *tau′ rho D* **unfolding** *DD-def diagram-def peak-def* **by** *auto*
have *alpha-ne*: *labels ?α* ≠ [] **unfolding** *labels-def* **by** *simp*
have ((*?τ′*,*?ϱ*),*P*) ∈ *pex r* **using** *lemma3-6-v*[*OF assms*(*1*) *i alpha-ne dec-dih1*] **unfolding** *pex-def measure-def decompose labels-def sigma-dec* **by** *auto*
from *this* **obtain** *ϱ′ τ″* **where** *IH2*: *DD ars r* (*?τ′*,*?ϱ*,*ϱ′*,*τ″*) **using** *1*(*1*)[*OF* - *P-IH2*] **by** *auto*
show *?thesis* **using** *lemma3-5-DD-v*[*OF assms*(*1*) *i DIH1 IH2*] **unfolding** *decompose fst-conv snd-conv sigma-dec* **by** *fast*
qed
qed
qed
qed

definition *LD-conv* :: *'b set* ⇒ *'a rel* ⇒ *bool*
where *LD-conv L ars* = (∃ (*r*:: (*'b rel*)) (*lrs*::(*'a*,*'b*) *lars*). (*ars* = *unlabel lrs*) ∧ *trans r* ∧ *wf r* ∧ (∀ *p*. (*local-peak lrs p* ⟶ (∃ *γ1 γ2 γ3 δ1 δ2 δ3*. *LT lrs r* (*fst p*,*snd p*,*γ1*,*γ2*,*γ3*,*δ1*,*δ2*,*δ3*)))))

lemma *sound-conv*: **assumes** *LD-conv L ars* **shows** *CR ars*
using *assms LT-imp-D D-imp-CR* **unfolding** *LD-conv-def* **by** *metis*

hide-const (**open**) *D*
hide-const (**open**) *seq*
hide-const (**open**) *measure*

hide-fact (**open**) *split*

end

References

[1] V. van Oostrom. Confluence by decreasing diagrams. *Theoretical Computer Science*, 126(2):259–280, 1994.

[2] V. van Oostrom. Confluence by decreasing diagrams – converted. In *Proc. 19th International Conference on Rewriting Techniques and Applications*, volume 5117 of *Lecture Notes in Computer Science*, pages 306–320, 2008.

[3] H. Zankl. Confluence by decreasing diagrams – formalized. In *Proc. 24th International Conference on Rewriting Techniques and Applications*, number 21 in Leibniz International Proceedings in Informatics, pages 352–367, 2013.