Decreasing-Diagrams

Harald Zankl

March 17, 2025

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)
⟨*proof*⟩

lemma *set-mset-diff* [*simp*]:
set-mset (M −s A) = set-mset M − A
⟨*proof*⟩

lemma *diff-eq-singleton-imp*:
M −s A = {#a#} ⟹ a ∈ (set-mset M − A)
⟨*proof*⟩

lemma *count-intersect* [*simp*]:
count (M ∩s A) a = count M a ∗ of-bool (a ∈ A)
⟨*proof*⟩

lemma *set-mset-intersect* [*simp*]:
set-mset (M ∩s A) = set-mset M ∩ A
⟨*proof*⟩

lemma *diff-from-empty*: *{#}−s S = {#}* ⟨*proof*⟩

lemma *diff-empty*: *M −s {} = M* ⟨*proof*⟩

lemma *submultiset-implies-subset*: **assumes** *M ⊆# N* **shows** *set-mset M ⊆ set-mset N*
⟨*proof*⟩

lemma *subset-implies-remove-empty*: **assumes** *set-mset M ⊆ S* **shows** *M −s S = {#}*
⟨*proof*⟩

lemma *remove-empty-implies-subset*: **assumes** *M −s S = {#}* **shows** *set-mset M ⊆ S* ⟨*proof*⟩

lemma *lemmaA-3-8*: *(M + N) −s S = (M −s S) + (N −s S)* ⟨*proof*⟩
lemma *lemmaA-3-9*: *(M −s S) −s T = M −s (S ∪ T)* ⟨*proof*⟩
lemma *lemmaA-3-10*: *M = (M ∩s S) + (M −s S)* ⟨*proof*⟩
lemma *lemmaA-3-11*: *(M −s T) ∩s S = (M ∩s S) −s T* ⟨*proof*⟩

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*⟩

from PhD thesis of van Oostrom

lemma *ds-monotone*: **assumes** *S* ⊆ *T* **shows** *ds r S* ⊆ *ds r T* ⟨*proof*⟩

lemma *subset-imp-ds-subset*: **assumes** *trans r* **and** *S* ⊆ *ds r T* **shows** *ds r S* ⊆ *ds r T*
⟨*proof*⟩

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*
⟨*proof*⟩

lemma *downset-intro*:
assumes ∀ *k*∈*set-mset K*.∃ *j*∈*set-mset J*.(*k*,*j*)∈*r* **shows** *set-mset K* ⊆ *dm r J*
⟨*proof*⟩

lemma *downset-elim*:
assumes *set-mset K* $\subseteq$ *dm r J* **shows** $\forall k{\in}$*set-mset K*.$\exists j{\in}$*set-mset J*.$(k,j){\in}r$
⟨*proof*⟩

to closure-free representation

lemma *mult-eq-implies-one-or-zero-step*:
assumes *trans r* **and** $(M,N) \in$ *mult-eq r* **shows** $\exists I\ J\ K.\ N = I + J \wedge M = I + K \wedge$ *set-mset K* $\subseteq$ *dm r J*
⟨*proof*⟩

from closure-free representation

lemma *one-step-implies-mult-eq*: **assumes** *trans r* **and** *set-mset K* $\subseteq$ *dm r J* **shows** $(I+K,I+J){\in}$*mult-eq r*
⟨*proof*⟩

lemma *mult-is-mul*: **assumes** *trans r* **shows** *mult r* = *mul r* ⟨*proof*⟩

lemma *mult-eq-is-mul-eq*: **assumes** *trans r* **shows** *mult-eq r* = *mul-eq r* ⟨*proof*⟩

lemma *mul-eq r* = $(mul\ r)^=$ ⟨*proof*⟩

useful properties on multisets

lemma *mul-eq-reflexive*: $(M,M) \in$ *mul-eq r* ⟨*proof*⟩

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*
⟨*proof*⟩

lemma *mul-eq-singleton*: **assumes** $(M, \{\#\alpha\#\}) \in$ *mul-eq r* **shows** $M = \{\#\alpha\#\} \vee$ *set-mset M* $\subseteq$ *dm r* $\{\#\alpha\#\}$ ⟨*proof*⟩

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*
⟨*proof*⟩

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*
⟨*proof*⟩

lemma *wf-mul*: **assumes** *trans r* **and** *wf r* **shows** *wf* (*mul r*)
⟨*proof*⟩

lemma *remove-is-empty-imp-mul*: **assumes** *M* $-s$ *dm r* $\{\#\alpha\#\} = \{\#\}$ **shows** $(M,\{\#\alpha\#\}) \in$ *mul r* ⟨*proof*⟩

Lemma 2.6

lemma *lemma2-6-1-set*: *ds r* $(S \cup T)$ = *ds r S* $\cup$ *ds r T*
⟨*proof*⟩

lemma *lemma2-6-1-list*: $dl\ r\ (\sigma@\tau) = dl\ r\ \sigma \cup dl\ r\ \tau$
⟨*proof*⟩

lemma *lemma2-6-1-multiset*: $dm\ r\ (M + N) = dm\ r\ M \cup dm\ r\ N$
⟨*proof*⟩

lemma *lemma2-6-1-diff*: $(dm\ r\ M) - ds\ r\ S \subseteq dm\ r\ (M - s\ S)$
⟨*proof*⟩

missing but useful

lemma *dl-monotone*: $dl\ r\ (\sigma@\tau) \subseteq dl\ r\ (\sigma@\tau'@\tau)$ ⟨*proof*⟩

Lemma 2.6.2

lemma *lemma2-6-2-a*: **assumes** *t*: *trans r* **and** $M \subseteq\# N$ **shows** $(M,N) \in$ *mul-eq r* ⟨*proof*⟩

lemma *mul-eq-not-equal-imp-elt*:
assumes $(M,N) \in$*mul-eq r* **and** $y \in$*set-mset M* $-$ *set-mset N* **shows** $\exists z \in$*set-mset N*.$(y,z) \in r$ ⟨*proof*⟩

lemma *lemma2-6-2-b*: **assumes** *trans r* **and** $(M,N) \in$ *mul-eq r* **shows** *dm r M* $\subseteq$ *dm r N* ⟨*proof*⟩

Lemma 2.6.3

lemma *ds-trans-contrapos*: **assumes** *t*: *trans r* **and** $x \notin ds\ r\ S$ **and** $(x,y) \in r$ **shows** $y \notin ds\ r\ S$
⟨*proof*⟩

lemma *dm-max-elt*: **assumes** *i*: *irrefl 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*⟩

lemma *dm-subset*: **assumes** *i*:*irrefl r* **and** *t*: *trans r* **shows** $dm\ r\ M \subseteq dm\ r\ (M - s\ dm\ r\ M)$
⟨*proof*⟩

lemma *dm-eq*: **assumes** *i*:*irrefl r* **and** *t*: *trans r* **shows** $dm\ r\ M = dm\ r\ (M - s\ dm\ r\ M)$
⟨*proof*⟩

lemma *lemma2-6-3*: **assumes** *t*:*trans r* **and** *i*:*irrefl r* **and** $(M,N) \in$ *mul-eq r*
shows $\exists\ I'\ J'\ K'.\ N = I' + J' \wedge M = I' + K' \wedge J' \cap\# K' = \{\#\} \wedge$ *set-mset* $K' \subseteq dm\ r\ J'$
⟨*proof*⟩

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*⟩

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*⟩

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*
⟨*proof*⟩

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*⟩

lemma *add-left-one*:
assumes $\exists$ *I J K*. *add-mset q N* = *I* + *J* $\wedge$ *add-mset q M* = *I* + *K* $\wedge$ (*J*$\cap$#*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*⟩

lemma *lemma2-6-6-b-one* :
assumes *trans r* **and** *irrefl r* **and** (*add-mset q M*, *add-mset q N*) $\in$ *mul-eq r*
shows (*M*,*N*) $\in$ *mul-eq r*
⟨*proof*⟩

lemma *lemma2-6-6-b′* : **assumes** *trans r* **and** *i*: *irrefl r* **and** (*Q* + *M*, *Q* + *N*) $\in$ *mul-eq r*
shows (*M*,*N*) $\in$ *mul-eq r* ⟨*proof*⟩

lemma *lemma2-6-9*: **assumes** *t*: *trans r* **and** (*M*,*N*) $\in$ *mul r* **shows** (*Q*+*M*,*Q*+*N*) $\in$ *mul r* ⟨*proof*⟩

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*⟩

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*⟩

1.1.3 Lexicographic maximum measure

Def 3.1: lexicographic maximum measure

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

notation

lexmax (‹-|-|›) [*1000*] *1000*)

lemma *lexmax-singleton*: $r|[\alpha]| = \{\#\alpha\#\}$ ⟨*proof*⟩

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*⟩

Lemma 3.2(2)

lemma *lemma3-2-2*: $r|\sigma@\tau| = r|\sigma| + (r|\tau| -s\ r \downarrow l\ \sigma)$ ⟨*proof*⟩

Definition 3.3

definition *D* :: *'a rel* ⇒ *'a list* ⇒ *'a list* ⇒ *'a list* ⇒ *'a list* ⇒ *bool* **where**
D r τ σ σ' τ' = ((*r*|*σ*@*τ'*|, *r*|*τ*| + *r*|*σ*|) ∈ *mul-eq r*
∧ (*r*|*τ*@*σ'*|, *r*|*τ*| + *r*|*σ*|) ∈ *mul-eq r*)

lemma *D-eq*: **assumes** *trans r* **and** *irrefl r* **and** *D r τ σ σ' τ'*
shows (*r*|*τ'*| −*s dl r σ*,*r*|*τ*|) ∈ *mul-eq r* **and** (*r*|*σ'*| −*s dl r τ*,*r*|*σ*|) ∈ *mul-eq r*
⟨*proof*⟩

lemma *D-inv*: **assumes** *trans r* **and** *irrefl r* **and** (*r*|*τ'*| −*s dl r σ*,*r*|*τ*|) ∈ *mul-eq r*
and (*r*|*σ'*| −*s dl r τ*,*r*|*σ*|) ∈ *mul-eq r*
shows *D r τ σ σ' τ'*
⟨*proof*⟩

lemma *D*: **assumes** *trans r* **and** *irrefl r*
shows *D r τ σ σ' τ'* = ((*r*|*τ'*| −*s dl r σ*,*r*|*τ*|) ∈ *mul-eq r*
∧ (*r*|*σ'*| −*s dl r τ*,*r*|*σ*|) ∈ *mul-eq r*)
⟨*proof*⟩

lemma *mirror-D*: **assumes** *trans r* **and** *irrefl r* **and** *D r τ σ σ' τ'* **shows** *D r σ τ τ' σ'*
⟨*proof*⟩

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*⟩

lemma *split*: **assumes** *LD-1'* $r\ \beta\ \alpha\ \sigma 1\ \sigma 2\ \sigma 3$ **shows** $\sigma 2 = [] \vee \sigma 2 = [\alpha]$
⟨*proof*⟩

lemma *proposition3-4-step*: **assumes** *trans* r **and** *irrefl* r **and** *LD-1'* $r\ \beta\ \alpha\ \sigma 1\ \sigma 2\ \sigma 3$
shows $(r|\sigma 1@\sigma 2@\sigma 3| -s\ (dm\ r\ \{\#\beta\#\}),\ r|[\alpha]|\ ) \in$ *mul-eq* r ⟨*proof*⟩

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 $D\ r\ [\beta]\ [\alpha]\ (\sigma 1@\sigma 2@\sigma 3)\ (\tau 1@\tau 2@\tau 3)$
⟨*proof*⟩

lemma *lexmax-decompose*: **assumes** $\alpha \in\# r|\sigma|$ **shows** $\exists \sigma 1\ \sigma 3.\ (\sigma = \sigma 1@[\alpha]@\sigma 3 \wedge \alpha \notin dl\ r\ \sigma 1)$
⟨*proof*⟩

lemma *lexmax-elt*: **assumes** *trans* r **and** $x \in (set\ \sigma)$ **and** $x \notin$ *set-mset* $r|\sigma|$
shows $\exists y.\ (x,y) \in r \wedge y \in$ *set-mset* $r|\sigma|$ ⟨*proof*⟩

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*⟩

lemma *drop-left-mult-eq*:
assumes *trans* r **and** *irrefl* r **and** $(N+M,M) \in$ *mul-eq* r **shows** $N = \{\#\}$ ⟨*proof*⟩

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** $(?M,\text{-}) \in \text{-}$)
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*⟩

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-eq* r (**is** $(?M,\text{-}) \in \text{-}$)
shows $\exists\ \sigma 1\ \sigma 2\ \sigma 3.\ ((\sigma = \sigma 1@\sigma 2@\sigma 3) \wedge LD\text{-}1'\ r\ \beta\ \alpha\ \sigma 1\ \sigma 2\ \sigma 3)$
⟨*proof*⟩

lemma *proposition3-4-inv*:
assumes *t*: *trans* r **and** *i*: *irrefl* r **and** $D\ r\ [\beta]\ [\alpha]\ \sigma\ \tau$
shows $\exists\ \sigma 1\ \sigma 2\ \sigma 3\ \tau 1\ \tau 2\ \tau 3.\ (\sigma = \sigma 1@\sigma 2@\sigma 3 \wedge \tau = \tau 1@\tau 2@\tau 3 \wedge LD'\ r\ \beta\ \alpha\ \sigma 1\ \sigma 2\ \sigma 3\ \tau 1\ \tau 2\ \tau 3)$
⟨*proof*⟩

Lemma 3.5

lemma *lemma3-5-1*:

assumes t: *trans* r **and** *irrefl* r **and** $D\ r\ \tau\ \sigma\ \sigma'\ \tau'$ **and** $D\ r\ \upsilon\ \sigma'\ \sigma''\ \upsilon'$
shows (*lexmax* r (τ @ υ @ σ''), *lexmax* r (τ @ υ) + *lexmax* r σ) $\in$ *mul-eq* r ⟨*proof*⟩

lemma *claim1*: **assumes** t: *trans* r **and** $D\ r\ \tau\ \sigma\ \sigma'\ \tau'$
shows ($r|\sigma@\tau'|$ + (($r|\upsilon'|$ −*s* r ↓*l* ($\sigma@\tau'$)) ∩*s* r ↓*l* τ),$r|\sigma|$ + $r|\tau|$) $\in$ *mul-eq* r (**is** (*?F*+*?H*,*?G*) $\in$ -)
⟨*proof*⟩

lemma *step3*: **assumes** t:*trans* r **and** $D\ r\ \tau\ \sigma\ \sigma'\ \tau'$
shows r ↓*l* ($\sigma@\tau$) ⊇ (r ↓*m* ($r|\sigma'|$ + $r|\tau|$)) ⟨*proof*⟩

lemma *claim2*: **assumes** t: *trans* r **and** $D\ r\ \tau\ \sigma\ \sigma'\ \tau'$
shows (($r|\upsilon'|$ −*s* r ↓*l* ($\sigma@\tau'$)) −*s* r ↓*l* τ, ($r|\upsilon'|$ −*s* r ↓*l* σ') −*s* r ↓*l* τ) $\in$ *mul-eq* r (**is** (*?L*,*?R*) $\in$ -)
⟨*proof*⟩

lemma *lemma3-5-2*: **assumes** *trans* r **and** *irrefl* r **and** $D\ r\ \tau\ \sigma\ \sigma'\ \tau'$ **and** $D\ r\ \upsilon\ \sigma'\ \sigma''\ \upsilon'$
shows ($r|(\sigma$ @ τ' @ $\upsilon')|$, $r|\sigma|$ + $r|(\tau@\upsilon)|$) $\in$ *mul-eq* r
⟨*proof*⟩

lemma *lemma3-5*: **assumes** *trans* r **and** *irrefl* r **and** $D\ r\ \tau\ \sigma\ \sigma'\ \tau'$ **and** $D\ r\ \upsilon\ \sigma'\ \sigma''\ \upsilon'$
shows $D\ r$ ($\tau@\upsilon$) σ σ'' ($\tau'@\upsilon'$)
⟨*proof*⟩

lemma *step2*: **assumes** *trans* r **and** $\tau \neq []$ **shows** (M ∩*s* *dl* r τ,*lexmax* r τ) $\in$ *mul* r ⟨*proof*⟩

Lemma 3.6

lemma *lemma3-6*: **assumes** t: *trans* r **and** *ne*: $\tau \neq []$ **and** D: $D\ r\ \tau\ \sigma\ \sigma'\ \tau'$
shows ($r|\sigma'|$ + $r|\upsilon|$, $r|\sigma|$ + $r|\tau@\upsilon|$) $\in$ *mul* r (**is** (*?L*,*?R*) $\in$ -) ⟨*proof*⟩

lemma *lemma3-6-v*: **assumes** *trans* r **and** *irrefl* r **and** $\sigma \neq []$ **and** $D\ r\ \tau\ \sigma\ \sigma'\ \tau'$
shows ($r|\tau'|$ + $r|\upsilon|$, $r|\tau|$ + $r|\sigma@\upsilon|$) $\in$ *mul* r
⟨*proof*⟩

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*: $(s,x\#xs) \in seq\ lars$
shows $(snd\ x,xs) \in seq\ lars$ **and** $(s,fst\ x,snd\ x) \in lars$ **and** $lst\ (s,x\#xs) = lst\ (snd\ x,xs)$
⟨*proof*⟩

lemma *seq-chop*: **assumes** $(s,ss@ts) \in seq\ ars$ **shows** $(s,ss) \in seq\ ars\ (lst(s,ss),ts) \in seq\ ars$ ⟨*proof*⟩

lemma *seq-concat-helper*:
assumes $(s,ls) \in seq\ ars$ **and** $ss2 \in seq\ ars$ **and** $lst\ (s,ls) = fst\ ss2$
shows $(s,ls@snd\ ss2) \in seq\ ars \wedge (lst\ (s,ls@snd\ ss2) = lst\ ss2)$
⟨*proof*⟩

lemma *seq-concat*:
assumes $ss1 \in seq\ ars$ **and** $ss2 \in seq\ ars$ **and** $lst\ ss1 = fst\ ss2$
shows $(fst\ ss1,snd\ ss1@snd\ ss2) \in seq\ ars$ **and** $(lst\ (fst\ ss1,snd\ ss1@snd\ ss2) = lst\ ss2)$
⟨*proof*⟩

diagrams

definition *diagram* :: $('a,'b)\ lars \Rightarrow ('a,'b)\ seq \times ('a,'b)\ seq \times ('a,'b)\ seq \times ('a,'b)\ seq \Rightarrow bool$
where $diagram\ ars\ d = (let\ (\tau,\sigma,\sigma',\tau') = d\ in\ \{\sigma,\tau,\sigma',\tau'\} \subseteq seq\ ars \wedge fst\ \sigma = fst\ \tau \wedge lst\ \sigma = fst\ \tau' \wedge lst\ \tau = fst\ \sigma' \wedge lst\ \sigma' = lst\ \tau')$

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

definition *D2* :: $'b\ rel \Rightarrow ('a,'b)\ seq \times ('a,'b)\ seq \times ('a,'b)\ seq \times ('a,'b)\ seq \Rightarrow bool$
where $D2\ r\ d = (let\ (\tau,\sigma,\sigma',\tau') = d\ in\ D\ r\ (labels\ \tau)\ (labels\ \sigma)\ (labels\ \sigma')\ (labels\ \tau'))$

lemma *lemma3-5-d*: **assumes** $diagram\ ars\ (\tau,\sigma,\sigma',\tau')$ **and** $diagram\ ars\ (\upsilon,\sigma',\sigma'',\upsilon')$
shows $diagram\ ars\ ((fst\ \tau,snd\ \tau@snd\ \upsilon),\sigma,\sigma'',(fst\ \tau'),snd\ \tau'@snd\ \upsilon')$ ⟨*proof*⟩

lemma *lemma3-5-d-v*: **assumes** $diagram\ ars\ (\tau,\sigma,\sigma',\tau')$ **and** $diagram\ ars\ (\tau',\upsilon,\upsilon',\tau'')$
shows $diagram\ ars\ (\tau,(fst\ \sigma,snd\ \sigma@snd\ \upsilon),(fst\ \sigma',snd\ \sigma'@snd\ \upsilon'),\tau'')$ ⟨*proof*⟩

lemma *lemma3-5'*: **assumes** *trans r* **and** *irrefl r* **and** $D2\ r\ (\tau,\sigma,\sigma',\tau')$ **and** $D2\ r\ (\upsilon,\sigma',\sigma'',\upsilon')$
shows $D2\ r\ ((fst\ \tau,snd\ \tau@snd\ \upsilon),\sigma,\sigma'',(fst\ \tau'),snd\ \tau'@snd\ \upsilon')$
⟨*proof*⟩

lemma *lemma3-5'-v*: **assumes** *trans r* **and** *irrefl r* **and** $D2\ r\ (\tau,\sigma,\sigma',\tau')$ **and** $D2\ r\ (\tau',\upsilon,\upsilon',\tau'')$
shows $D2\ r\ (\tau,\ (fst\ \sigma,snd\ \sigma@snd\ \upsilon),(fst\ \sigma',snd\ \sigma'@snd\ \upsilon'),\tau'')$ ⟨*proof*⟩

lemma *trivial-diagram*: **assumes** $\sigma \in$ *seq ars* **shows** *diagram ars* (σ,(*fst* σ,[]),(*lst* σ,[]),σ)
⟨*proof*⟩

lemma *trivial-D2*: **assumes** $\sigma \in$ *seq ars* **shows** *D2 r* (σ,(*fst* σ,[]),(*lst* σ,[]),σ)
⟨*proof*⟩

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')
⟨*proof*⟩

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'')
⟨*proof*⟩

lemma *trivial-DD*: **assumes** $\sigma \in$ *seq ars* **shows** *DD ars r* (σ,(*fst* σ,[]),(*lst* σ,[]),σ)
⟨*proof*⟩

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')
⟨*proof*⟩

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*⟩

lemma *wf*: **assumes** *trans r* **and** *wf r* **shows** *wf* (*pex r*) ⟨*proof*⟩

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** $\forall P.$ (*local-peak ars P* $\longrightarrow$ ($\exists$ σ' τ'. *DD ars r* (*fst P*,*snd P*,σ',τ')))
and *peak ars P* **shows** ($\exists$ σ' τ'. *DD ars r* (*fst P*,*snd P*,σ',τ')) ⟨*proof*⟩

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 ∃ *ss* ∈ *seq ars*. *fst ss* = *a* ∧ *lst ss* = *b* ⟨*proof*⟩

lemma *steps-imp-seq*: **assumes** (*a*,*b*) ∈ (*unlabel ars*)^*
shows ∃ *ss* ∈ *seq ars*. *fst ss* = *a* ∧ *lst ss* = *b* ⟨*proof*⟩

lemma *step-imp-unlabeled-step*: **assumes** (*a*,*b*,*c*) ∈ *ars* **shows** (*a*,*c*) ∈ (*unlabel ars*)
⟨*proof*⟩

lemma *seq-imp-steps*:
assumes *ss* ∈ *seq ars* **and** *fst ss* = *a* **and** *lst ss* = *b* **shows** (*a*,*b*) ∈ (*unlabel ars*)^*
⟨*proof*⟩

lemma *seq-vs-steps*: **shows** (*a*,*b*) ∈ (*unlabel ars*)^* = (∃ *ss*. *fst ss* = *a* ∧ *lst ss* = *b* ∧ *ss* ∈ *seq ars*)
⟨*proof*⟩

lemma *D-imp-CR*: **assumes** $\forall P.$ (*peak ars P* $\longrightarrow$ ($\exists$ σ' τ'. *DD ars r* (*fst P*,*snd P*,σ',τ'))) **shows** *CR* (*unlabel ars*) ⟨*proof*⟩

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* ∧ ($\forall P.$ (*local-peak lrs P* $\longrightarrow$ ($\exists$ σ' τ'. (*DD lrs r* (*fst P*,*snd P*,σ',τ')))))))

lemma *sound*: **assumes** *LD L ars* **shows** *CR ars*
⟨*proof*⟩

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*^+)^-1) {α} ⟨*proof*⟩

lemma *newman*: **assumes** *WCR ars* **and** *SN ars* **shows** *CR ars* ⟨*proof*⟩

1.2 Conversion Version

This section follows [2].

auxiliary results on multisets

lemma *mul-eq-add-right*: $(M,M+P) \in$ *mul-eq r* ⟨*proof*⟩

lemma *mul-add-right*: **assumes** $(M,N) \in$ *mul r* **shows** $(M,N+P) \in$ *mul r* ⟨*proof*⟩

lemma *mul-eq-and-ds-imp-ds*:
assumes *t*: *trans r* **and** $(M,N) \in$ *mul-eq r* **and** *set-mset N* $\subseteq$ *ds r S*
shows *set-mset M* $\subseteq$ *ds r S* ⟨*proof*⟩

lemma *lemma2-6-2-set*: **assumes** $S \subseteq T$ **shows** *ds r S* $\subseteq$ *ds r T* ⟨*proof*⟩

lemma *leq-imp-subseteq*: **assumes** $M \subseteq\# N$ **shows** *set-mset M* $\subseteq$ *set-mset N* ⟨*proof*⟩

lemma *mul-add-mul-eq-imp-mul*: **assumes** $(M,N) \in$ *mul r* **and** $(P,Q) \in$ *mul-eq r* **shows** $(M+P,N+Q) \in$ *mul r* ⟨*proof*⟩

labeled conversion

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

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

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

definition *measure-conv* :: $'b$ *rel* ⇒ $('a,'b)$ *conv* ⇒ $'b$ *multiset*
where *measure-conv r c* = *lexmax r* (*labels-conv c*)

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

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

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

definition *LDD* :: $('a,'b)$ *lars* ⇒ $'b$ *rel* ⇒ $('a,'b)$ *seq* × $('a,'b)$ *seq* × $('a,'b)$ *seq* × $('a,'b)$ *seq* × $('a,'b)$ *seq* × $('a,'b)$ *seq* × $('a,'b)$ *seq* ⇒ *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*⟩

lemma *conv-chop*: **assumes** (*s*,*ss1*@*ss2*) ∈ *conv ars* **shows** (*s*,*ss1*) ∈ *conv ars* (*lst-conv* (*s*,*ss1*),*ss2*) ∈ *conv ars* ⟨*proof*⟩

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*)
⟨*proof*⟩

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*⟩

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* ⟨*proof*⟩

lemma *seq-decompose*:
assumes *σ* ∈ *seq ars* **and** *labels σ* = *σ1'*@*σ2'*
shows ∃ *σ1 σ2*. ({*σ1*,*σ2*} ⊆ *seq ars* ∧ *σ* = (*fst σ1*,*snd σ1*@*snd σ2*) ∧ *lst σ1* = *fst σ2* ∧ *lst σ2* = *lst σ* ∧ *labels σ1* = *σ1'* ∧ *labels σ2* = *σ2'*) ⟨*proof*⟩

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*)
⟨*proof*⟩

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*⟩

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*⟩

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*⟩

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 δ*1* ∈ *conv ars*
and *set-mset* (*measure-conv r* δ*1*) ⊆ *dm r M*
and (*M*,{#*fst* α-*step*,*fst* β-*step*#}) ∈ *mul-eq r*
shows ∃ σ τ. ({σ,τ} ⊆ *seq ars* ∧ *fst* σ = *fst* δ*1* ∧ *fst* τ = *lst-conv* δ*1* ∧ *lst* σ = *lst* τ ∧ *set-mset* (*measure r* (σ,τ)) ⊆ *dm r M*) ⟨*proof*⟩

lemma *labels-multiset*: **assumes** *length* (*labels* σ) ≤ *1* **and** *set* (*labels* σ) ⊆ {α}
shows (*r*|*labels* σ|, {#α#}) ∈ *mul-eq r* ⟨*proof*⟩

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 ∃σ*1* σ*2* σ*3*. (σ'=(*fst* σ*1*,*snd* σ*1*@*snd* σ*2*@*snd* σ*3*) ∧ *lst* σ*1*=*fst* σ*2* ∧ *lst* σ*2* = *fst* σ*3* ∧ *lst* σ*3* = *lst* σ'
∧ *LD-1'* *r* β α (*labels* σ*1*) (*labels* σ*2*) (*labels* σ*3*))
set (*labels* τ') ⊆ *ds r* ({α,β})
⟨*proof*⟩

lemma *local-decreasing-extended-imp-decreasing*:

assumes *LT1 ars r* (*s*,[β-*step*]) (*s*,[α-*step*]) $\gamma 1$ $\gamma 2$ $\gamma 3$
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 ∃ $\sigma 1$ $\sigma 2$ $\sigma 3'$ $\gamma 1'''$. ({$\sigma 1$,$\sigma 2$,$\sigma 3'$,$\gamma 1'''$} ⊆ *seq ars* ∧
set (*labels* $\sigma 1$) ⊆ *ds r* {*fst* β-*step*} ∧ *length* (*labels* $\sigma 2$) ≤ 1 ∧ *set* (*labels* $\sigma 2$) ⊆ {*fst* α-*step*} ∧ *set* (*labels* $\sigma 3'$) ⊆ *ds r* {*fst* α-*step*,*fst* β-*step*}) ∧
set (*labels* $\gamma 1'''$) ⊆ *ds r* {*fst* α-*step*,*fst* β-*step*} ∧
snd β-*step* = *fst* $\sigma 1$ ∧ *lst* $\sigma 1$ = *fst* $\sigma 2$ ∧ *lst* $\sigma 2$ = *fst* $\sigma 3'$ ∧ *lst* $\sigma 3'$ = *lst* $\gamma 1'''$ ∧ *fst* $\gamma 1'''$ = *fst* $\gamma 3$
⟨*proof*⟩

lemma *LDD-imp-DD*:
assumes *t*:*trans r* **and** *i*:*irrefl r* **and** *LDD ars r* (τ,σ,$\sigma 1$,$\sigma 2$,$\sigma 3$,$\tau 1$,$\tau 2$,$\tau 3$)
shows ∃ σ' τ'. *DD ars r* (τ,σ,σ',τ') ⟨*proof*⟩

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*]),$\gamma 1$,$\gamma 2$,$\gamma 3$,$\delta 1$,$\delta 2$,$\delta 3$)
shows ∃ κ μ. *DD ars r* ((*s*,[β-*step*]),(*s*,[α-*step*]),κ,μ)
⟨*proof*⟩

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*⟩

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* ⟶ (∃ $\gamma 1$ $\gamma 2$ $\gamma 3$ $\delta 1$ $\delta 2$ $\delta 3$. *LT lrs r* (*fst p*,*snd p*,$\gamma 1$,$\gamma 2$,$\gamma 3$,$\delta 1$,$\delta 2$,$\delta 3$)))))

lemma *sound-conv*: **assumes** *LD-conv L ars* **shows** *CR ars*
⟨*proof*⟩

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.