Commutation

TRSs

Julian Nagele Vincent van Oostrom

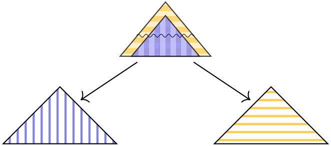
Computational Logic, University of Innsbruck

9th International School on Rewriting July 6 & 7, 2017

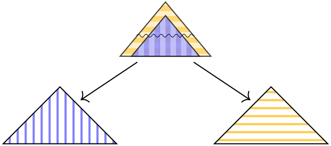
Outline

- Closing Critical Pairs
- Decreasing Diagrams
- Application: Program Transformations

Critical Pairs



Critical Pairs

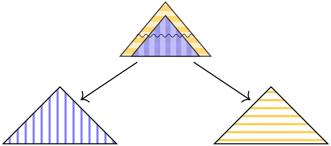


Definition

- $\ell_1 \to r_1 \in \mathcal{R}$ and $\ell_2 \to r_2 \in \mathcal{S}$
- $\mathcal{V}\mathsf{ar}(\ell_1 \to r_1) \cap \mathcal{V}\mathsf{ar}(\ell_2 \to r_2) = \varnothing$ (rename apart)
- $p \in \mathcal{P}\mathsf{os}_{\mathcal{F}}(\ell_2)$
- $\mathsf{mgu}(\ell_2|_p, \ell_1) = \sigma$

then $\ell_2\sigma[r_1\sigma]_p \;{}_{\mathcal{R}}\!\!\leftarrow \ell_2\sigma[\ell_1\sigma]_p \to_{\mathcal{S}} r_2\sigma$

Critical Pairs



Definition

- $\ell_1 \to r_1 \in \mathcal{R}$ and $\ell_2 \to r_2 \in \mathcal{S}$
- $\mathcal{V}\mathrm{ar}(\ell_1 \to r_1) \cap \mathcal{V}\mathrm{ar}(\ell_2 \to r_2) = \varnothing$ (rename apart)
- $p \in \mathcal{P}\mathrm{os}_{\mathcal{F}}(\ell_2)$
- $\mathrm{mgu}(\ell_2|_p, \ell_1) = \sigma$

then $\ell_2\sigma[r_1\sigma]_p \;{}_{\mathcal{R}}{\leftarrow}\!\rtimes\!{\rightarrow}_{\mathcal{S}}\; r_2\sigma$ is critical pair of $\mathcal{R}$ on $\mathcal{S}$

Critical Pair Lemma

Lemma (Knuth, Bendix, Huet)

TRS $\mathcal{R}$ is locally confluent iff

$$\leftarrow\!\rtimes\!\rightarrow \;\subseteq\; \rightarrow^* \cdot {}^*\!\leftarrow$$

Critical Pair Lemma

Lemma ()

TRSs $\mathcal{R}$ and $\mathcal{S}$ locally commute iff

$$({}_{\mathcal{R}}\!\leftarrow \rtimes \rightarrow_{\mathcal{S}}) \cup ({}_{\mathcal{R}}\!\leftarrow \ltimes \rightarrow_{\mathcal{S}}) \subseteq \rightarrow_{\mathcal{S}}^{*} \cdot {}_{\mathcal{R}}^{*}\!\leftarrow$$

Critical Pair Lemma

Lemma ()

TRSs $\mathcal{R}$ and $\mathcal{S}$ locally commute iff

$$({}_{\mathcal{R}}\!\leftarrow \rtimes \rightarrow_{\mathcal{S}}) \cup ({}_{\mathcal{R}}\!\leftarrow \ltimes \rightarrow_{\mathcal{S}}) \subseteq \rightarrow^*_{\mathcal{S}} \cdot {}^{\;*}_{\mathcal{R}}\!\leftarrow$$

Example

- TRSs

$$\mathcal{S}: \; \mathsf{f}(x, x) \to \mathsf{c} \qquad\qquad \mathcal{R}: \; \mathsf{a} \to \mathsf{b}$$

- $({}_{\mathcal{R}}\!\leftarrow \rtimes \rightarrow_{\mathcal{S}}) \cup ({}_{\mathcal{R}}\!\leftarrow \ltimes \rightarrow_{\mathcal{S}}) = \varnothing$
- $\mathsf{f}(\mathsf{a}, \mathsf{b}) \; {}_{\mathcal{R}}\!\leftarrow \mathsf{f}(\mathsf{a}, \mathsf{a}) \rightarrow_{\mathcal{S}} \mathsf{c}$
- $\mathsf{f}(\mathsf{a}, \mathsf{b}) \in \mathsf{NF}(\mathcal{S})$ and $\mathsf{c} \in \mathsf{NF}(\mathcal{R})$

Critical Pair Lemma

Lemma (Folklore)

Left-linear TRSs $\mathcal{R}$ and $\mathcal{S}$ locally commute iff

$$({}_{\mathcal{R}}\!\leftarrow \rtimes \rightarrow_{\mathcal{S}}) \cup ({}_{\mathcal{R}}\!\leftarrow \ltimes \rightarrow_{\mathcal{S}}) \subseteq \rightarrow_{\mathcal{S}}^{*} \cdot {}_{\mathcal{R}}^{*}\!\leftarrow$$

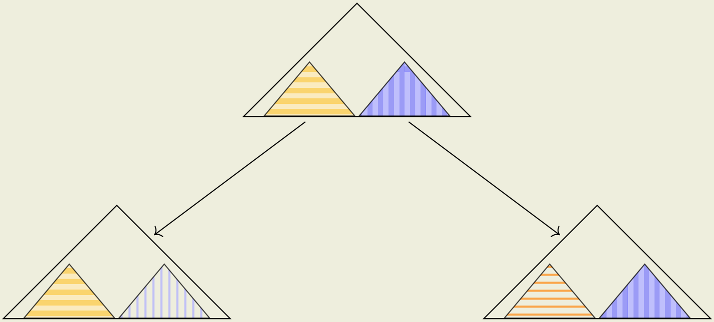
Example

- TRSs

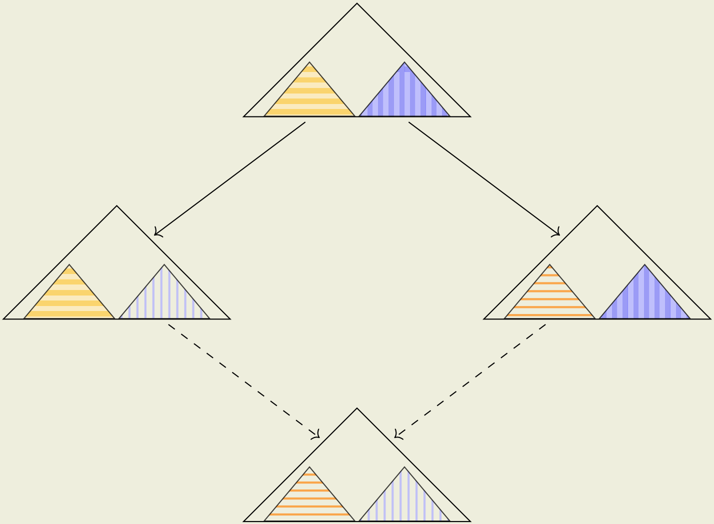
$$\mathcal{S}:\ \mathsf{f}(x,x) \to \mathsf{c} \qquad \mathcal{R}:\ \mathsf{a} \to \mathsf{b}$$

- $({}_{\mathcal{R}}\!\leftarrow \rtimes \rightarrow_{\mathcal{S}}) \cup ({}_{\mathcal{R}}\!\leftarrow \ltimes \rightarrow_{\mathcal{S}}) = \varnothing$
- $\mathsf{f}(\mathsf{a},\mathsf{b})\ {}_{\mathcal{R}}\!\leftarrow \mathsf{f}(\mathsf{a},\mathsf{a}) \rightarrow_{\mathcal{S}} \mathsf{c}$
- $\mathsf{f}(\mathsf{a},\mathsf{b}) \in \mathsf{NF}(\mathcal{S})$ and $\mathsf{c} \in \mathsf{NF}(\mathcal{R})$

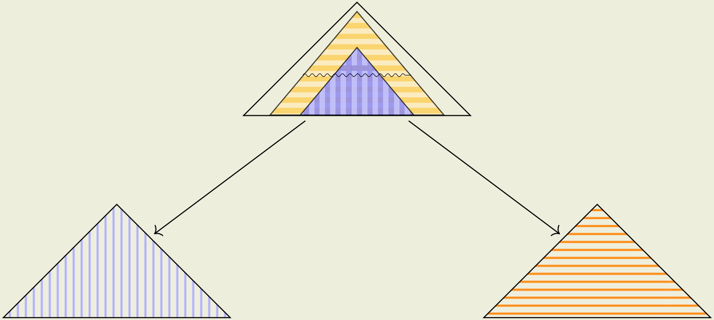
Proof



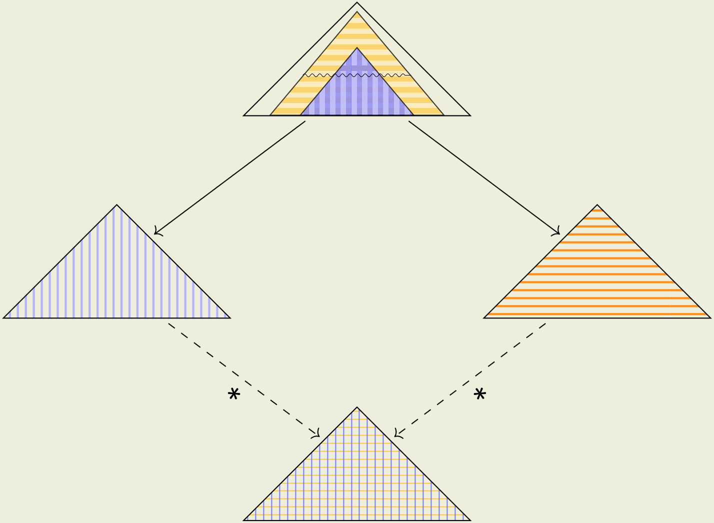
Proof



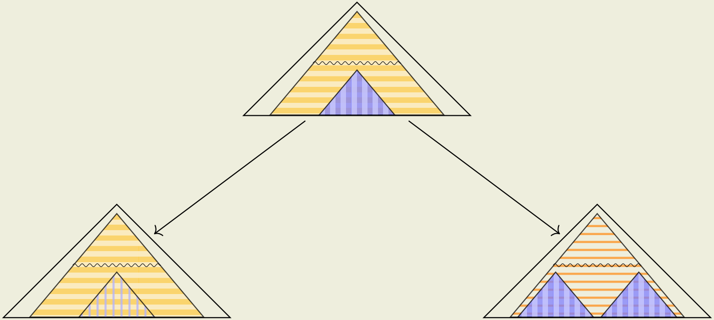
Proof



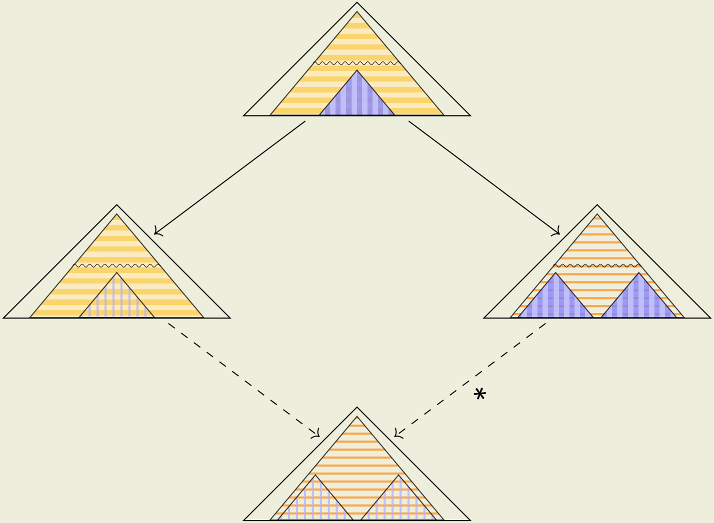
Proof



Proof



Proof



Exercises

- Do the following two TRSs commute?

$$\mathcal{R}: \quad 0 + y \to y \qquad s(x) + y \to s(x + y)$$
$$\mathcal{S}: \quad x + 0 \to x \qquad x + s(y) \to s(x + y)$$

- Show by picture: Linear TRSs $\mathcal{R}$ and $\mathcal{S}$ strongly commute if

$$({}_{\mathcal{R}}\!\leftarrow\rtimes\!\to_{\mathcal{S}}) \cup ({}_{\mathcal{R}}\!\leftarrow\ltimes\!\to_{\mathcal{S}}) \subseteq \to_{\mathcal{S}}^{*} \cdot \; {}_{\mathcal{R}}^{=}\!\!\leftarrow$$

Recall

- TRSs $\mathcal{R}$ and $\mathcal{S}$ strongly commute if

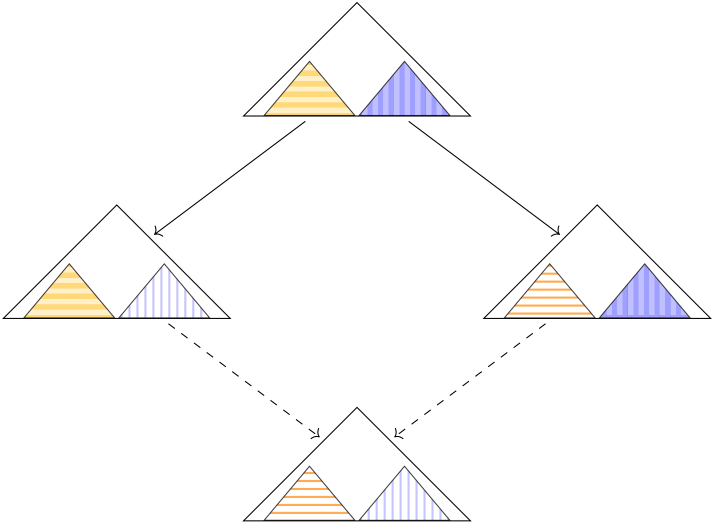
$${}_{\mathcal{R}}\!\leftarrow \cdot \to_{\mathcal{S}} \subseteq \to_{\mathcal{S}}^{*} \cdot \; {}_{\mathcal{R}}^{=}\!\!\leftarrow$$

Solutions

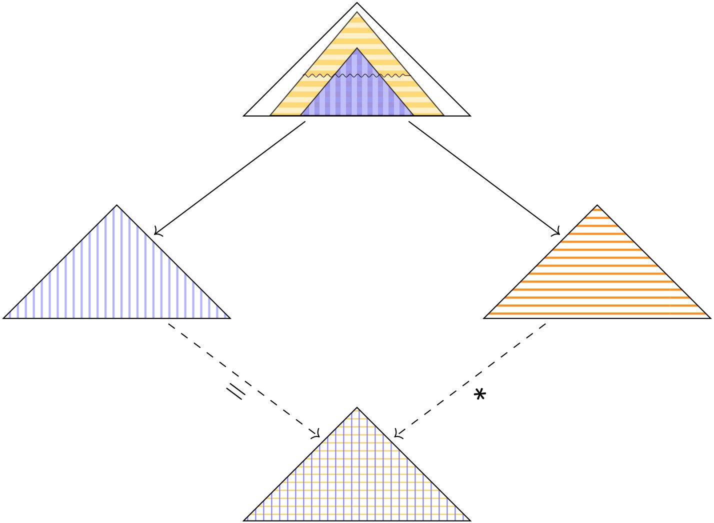
- $\mathcal{R} \cup \mathcal{S}$ is terminating
- critical pairs are joinable

$$
\begin{aligned}
0 \;{}_{\mathcal{R}}{\leftarrow} \rtimes {\rightarrow}_{\mathcal{S}}\; 0 &\qquad& 0 \;{}_{\mathcal{R}}{\leftarrow} \ltimes {\rightarrow}_{\mathcal{S}}\; 0 \\
s(x+0) \;{}_{\mathcal{R}}{\leftarrow} \rtimes {\rightarrow}_{\mathcal{S}}\; s(x) &\qquad& s(x+0) \;{}_{\mathcal{R}}{\leftarrow} \ltimes {\rightarrow}_{\mathcal{S}}\; s(x) \\
s(y) \;{}_{\mathcal{R}}{\leftarrow} \rtimes {\rightarrow}_{\mathcal{S}}\; s(0+y) &\qquad& s(y) \;{}_{\mathcal{R}}{\leftarrow} \ltimes {\rightarrow}_{\mathcal{S}}\; s(0+y) \\
s(x+s(y)) \;{}_{\mathcal{R}}{\leftarrow} \rtimes {\rightarrow}_{\mathcal{S}}\; s(s(x)+y) &\qquad& s(x+s(y)) \;{}_{\mathcal{R}}{\leftarrow} \ltimes {\rightarrow}_{\mathcal{S}}\; s(s(x)+y)
\end{aligned}
$$

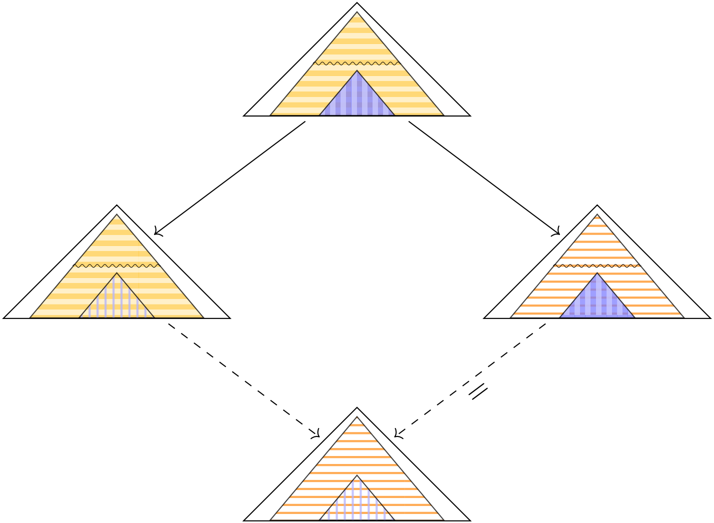
Solutions



Solutions



Solutions



Rule Labeling

$$_{\mathcal{R}}\xleftarrow{\alpha} \cdot \xrightarrow{\beta}_{\mathcal{S}} \subseteq \xrightarrow{\curlyvee\alpha}{}^{*}_{\mathcal{S}} \cdot \xrightarrow{\beta}{}^{=}_{\mathcal{S}} \cdot \xrightarrow{\curlyvee\alpha\beta}{}^{*}_{\mathcal{S}} \cdot {}_{\mathcal{R}}^{*}\xleftarrow{\curlyvee\alpha\beta} \cdot {}_{\mathcal{R}}^{=}\xleftarrow{\alpha} \cdot {}_{\mathcal{R}}^{*}\xleftarrow{\curlyvee\beta}$$

Rule Labeling

$$ {}_{\mathcal{R}}\xleftarrow{\alpha} \cdot \xrightarrow{\beta}_{\mathcal{S}} \subseteq \xrightarrow{\curlyvee\alpha}{}^{*}_{\mathcal{S}} \cdot \xrightarrow{\beta}{}^{=}_{\mathcal{S}} \cdot \xrightarrow{\curlyvee\alpha\beta}{}^{*}_{\mathcal{S}} \cdot {}^{*}_{\mathcal{R}}\xleftarrow{\curlyvee\alpha\beta} \cdot {}^{=}_{\mathcal{R}}\xleftarrow{\alpha} \cdot {}^{*}_{\mathcal{R}}\xleftarrow{\curlyvee\beta} $$

Definition

- rule labeling for TRS $\mathcal{R}$ is function $\phi : \mathcal{R} \to \mathbb{N}$
- $s \xrightarrow{\alpha} t$ if $s \to_{\ell\to r,p,\sigma} t$ and $\phi(\ell \to r) = \alpha$

Theorem

Linear TRS $\mathcal{R}$ and $\mathcal{S}$ commute if there is a rule labeling ϕ for $\mathcal{R} \cup \mathcal{S}$ such that

$$ ({}_{\mathcal{R}}\xleftarrow{\alpha} \rtimes \xrightarrow{\beta}_{\mathcal{S}}) \cup ({}_{\mathcal{R}}\xleftarrow{\alpha} \ltimes \xrightarrow{\beta}_{\mathcal{S}}) \subseteq \xrightarrow{\curlyvee\alpha}{}^{*}_{\mathcal{S}} \cdot \xrightarrow{\beta}{}^{=}_{\mathcal{S}} \cdot \xrightarrow{\curlyvee\alpha\beta}{}^{*}_{\mathcal{S}} \cdot {}^{*}_{\mathcal{R}}\xleftarrow{\curlyvee\alpha\beta} \cdot {}^{=}_{\mathcal{R}}\xleftarrow{\alpha} \cdot {}^{*}_{\mathcal{R}}\xleftarrow{\curlyvee\beta} $$

Rule Labeling

$$\mathrel{_{\mathcal{R}}\xleftarrow{\alpha}} \cdot \xrightarrow{\beta}_{\mathcal{S}} \subseteq \xrightarrow{\curlyvee\alpha}{}^{*}_{\mathcal{S}} \cdot \xrightarrow{\beta}{}^{=}_{\mathcal{S}} \cdot \xrightarrow{\curlyvee\alpha\beta}{}^{*}_{\mathcal{S}} \cdot {}_{\mathcal{R}}^{*}\xleftarrow{\curlyvee\alpha\beta} \cdot {}_{\mathcal{R}}^{=}\xleftarrow{\alpha} \cdot {}_{\mathcal{R}}^{*}\xleftarrow{\curlyvee\beta}$$

Definition

- rule labeling for TRS $\mathcal{R}$ is function $\phi : \mathcal{R} \to \mathbb{N}$
- $s \xrightarrow{\alpha} t$ if $s \to_{\ell\to r,p,\sigma} t$ and $\phi(\ell \to r) = \alpha$

Theorem

Linear TRS $\mathcal{R}$ and $\mathcal{S}$ commute if there is a rule labeling ϕ for $\mathcal{R} \cup \mathcal{S}$ such that

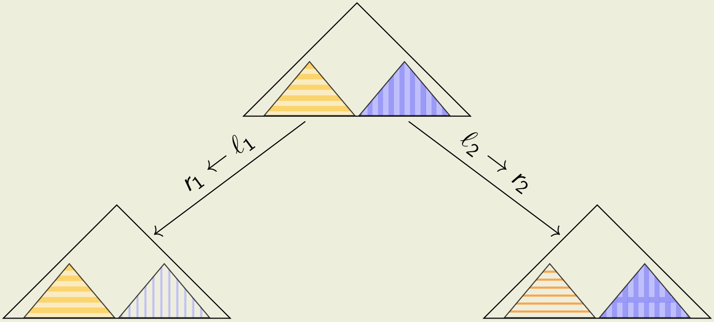
$$({}_{\mathcal{R}}\xleftarrow{\alpha} \rtimes \xrightarrow{\beta}_{\mathcal{S}}) \cup ({}_{\mathcal{R}}\xleftarrow{\alpha} \ltimes \xrightarrow{\beta}_{\mathcal{S}}) \subseteq \xrightarrow{\curlyvee\alpha}{}^{*}_{\mathcal{S}} \cdot \xrightarrow{\beta}{}^{=}_{\mathcal{S}} \cdot \xrightarrow{\curlyvee\alpha\beta}{}^{*}_{\mathcal{S}} \cdot {}_{\mathcal{R}}^{*}\xleftarrow{\curlyvee\alpha\beta} \cdot {}_{\mathcal{R}}^{=}\xleftarrow{\alpha} \cdot {}_{\mathcal{R}}^{*}\xleftarrow{\curlyvee\beta}$$

Question

- How to implement?

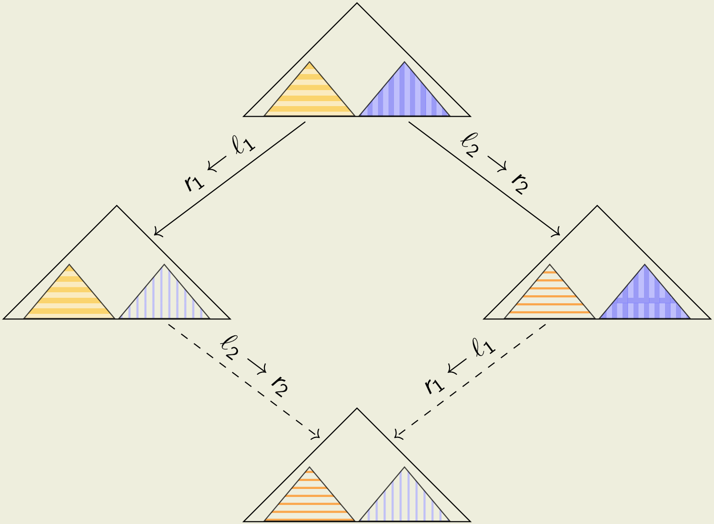
$$\mathrel{{}_{\mathcal{R}}\!\overset{\alpha}{\longleftarrow}} \cdot \mathrel{\overset{\beta}{\longrightarrow}_{\mathcal{S}}} \subseteq \mathrel{\overset{\curlyvee\alpha}{\longrightarrow}{}^{*}_{\mathcal{S}}} \cdot \mathrel{\overset{\beta}{\longrightarrow}{}^{=}_{\mathcal{S}}} \cdot \mathrel{\overset{\curlyvee\alpha\beta}{\longrightarrow}{}^{*}_{\mathcal{S}}} \cdot \mathrel{{}^{*}_{\mathcal{R}}\!\overset{\curlyvee\alpha\beta}{\longleftarrow}} \cdot \mathrel{{}^{=}_{\mathcal{R}}\!\overset{\alpha}{\longleftarrow}} \cdot \mathrel{{}^{*}_{\mathcal{R}}\!\overset{\curlyvee\beta}{\longleftarrow}}$$

Proof



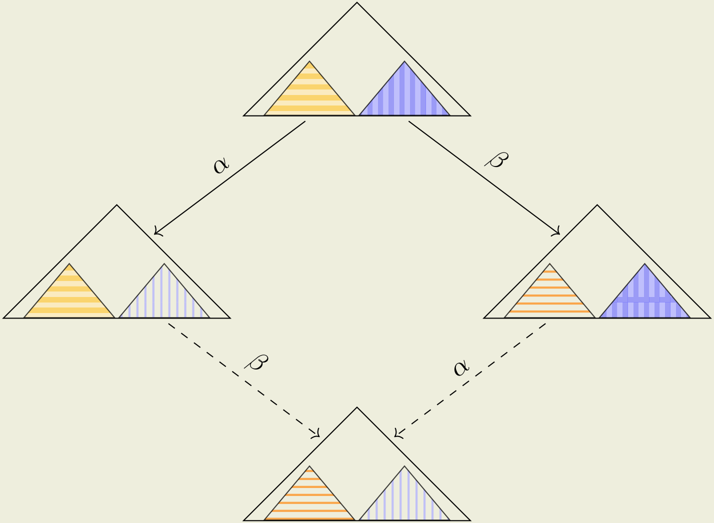
$$\mathcal{R}\xleftarrow{\alpha} \cdot \xrightarrow{\beta}_{\mathcal{S}} \subseteq \xrightarrow{\curlyvee\alpha}{}^{*}_{\mathcal{S}} \cdot \xrightarrow{\beta}{}^{=}_{\mathcal{S}} \cdot \xrightarrow{\curlyvee\alpha\beta}{}^{*}_{\mathcal{S}} \cdot {}^{*}_{\mathcal{R}}\xleftarrow{\curlyvee\alpha\beta} \cdot {}^{=}_{\mathcal{R}}\xleftarrow{\alpha} \cdot {}^{*}_{\mathcal{R}}\xleftarrow{\curlyvee\beta}$$

Proof



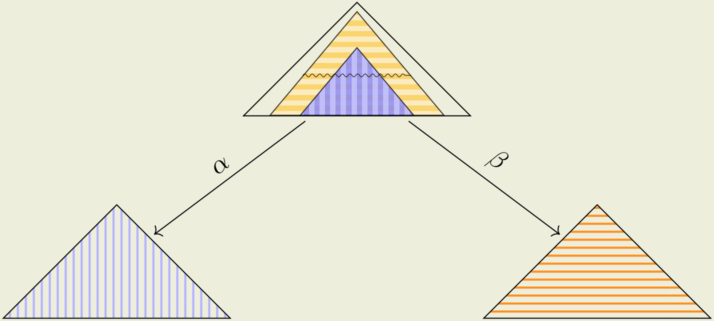
$$\mathrel{{}_{\mathcal{R}}\xleftarrow{\alpha}} \cdot \xrightarrow{\beta}_{\mathcal{S}} \subseteq \xrightarrow{\curlyvee\alpha}{}^{*}_{\mathcal{S}} \cdot \xrightarrow{\beta}{}^{=}_{\mathcal{S}} \cdot \xrightarrow{\curlyvee\alpha\beta}{}^{*}_{\mathcal{S}} \cdot {}^{*}_{\mathcal{R}}\xleftarrow{\curlyvee\alpha\beta} \cdot {}^{=}_{\mathcal{R}}\xleftarrow{\alpha} \cdot {}^{*}_{\mathcal{R}}\xleftarrow{\curlyvee\beta}$$

Proof



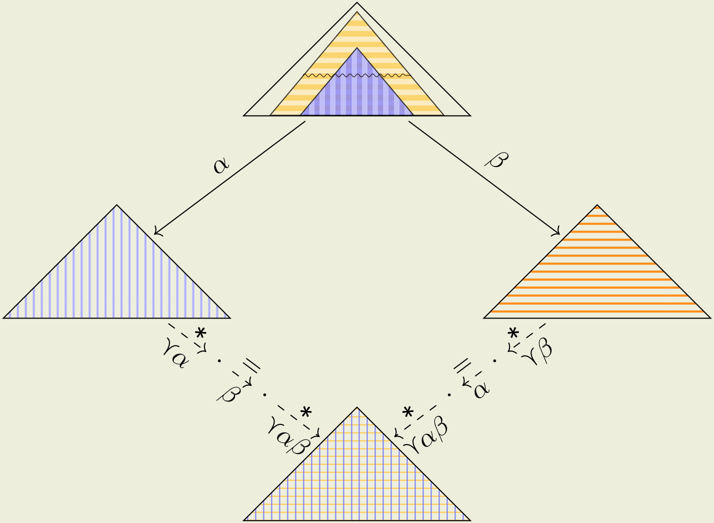
$$\mathcal{R}\xleftarrow{\alpha} \cdot \xrightarrow{\beta}_{\mathcal{S}} \subseteq \xrightarrow{\curlyvee\alpha}{}^{*}_{\mathcal{S}} \cdot \xrightarrow{\beta}{}^{=}_{\mathcal{S}} \cdot \xrightarrow{\curlyvee\alpha\beta}{}^{*}_{\mathcal{S}} \cdot {}^{*}_{\mathcal{R}}\xleftarrow{\curlyvee\alpha\beta} \cdot {}^{=}_{\mathcal{R}}\xleftarrow{\alpha} \cdot {}^{*}_{\mathcal{R}}\xleftarrow{\curlyvee\beta}$$

Proof



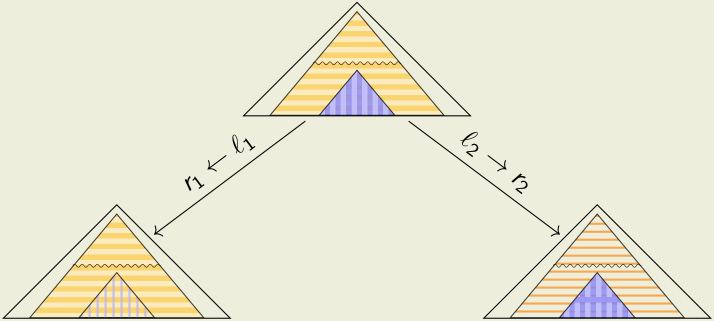
$$\mathcal{R}\overset{\alpha}{\longleftarrow} \cdot \overset{\beta}{\longrightarrow}_{\mathcal{S}} \subseteq \overset{\curlyvee\alpha}{\longrightarrow}{}^{*}_{\mathcal{S}} \cdot \overset{\beta}{\longrightarrow}{}^{=}_{\mathcal{S}} \cdot \overset{\curlyvee\alpha\beta}{\longrightarrow}{}^{*}_{\mathcal{S}} \cdot {}^{*}_{\mathcal{R}}\overset{\curlyvee\alpha\beta}{\longleftarrow} \cdot {}^{=}_{\mathcal{R}}\overset{\alpha}{\longleftarrow} \cdot {}^{*}_{\mathcal{R}}\overset{\curlyvee\beta}{\longleftarrow}$$

Proof



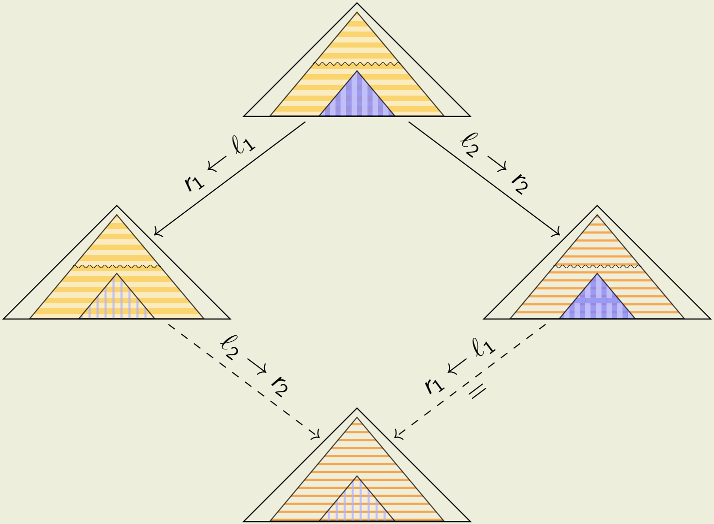
$$\underset{\mathcal{R}}{\xleftarrow{\alpha}} \cdot \xrightarrow[\mathcal{S}]{\beta} \subseteq \xrightarrow[\mathcal{S}]{\curlyvee\alpha}{}^{*} \cdot \xrightarrow[\mathcal{S}]{\beta}{}^{=} \cdot \xrightarrow[\mathcal{S}]{\curlyvee\alpha\beta}{}^{*} \cdot {}^{*}\underset{\mathcal{R}}{\xleftarrow{\curlyvee\alpha\beta}} \cdot {}^{=}\underset{\mathcal{R}}{\xleftarrow{\alpha}} \cdot {}^{*}\underset{\mathcal{R}}{\xleftarrow{\curlyvee\beta}}$$

Proof



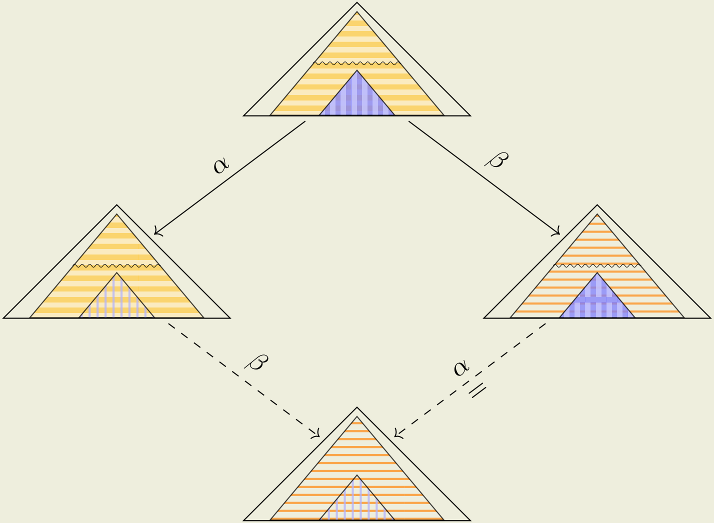
$$\underset{\mathcal{R}}{\overset{\alpha}{\longleftarrow}} \cdot \xrightarrow[\mathcal{S}]{\beta} \subseteq \xrightarrow[\mathcal{S}]{\curlyvee\alpha}{}^{*} \cdot \xrightarrow[\mathcal{S}]{\beta}{}^{=} \cdot \xrightarrow[\mathcal{S}]{\curlyvee\alpha\beta}{}^{*} \cdot {}^{*}\underset{\mathcal{R}}{\overset{\curlyvee\alpha\beta}{\longleftarrow}} \cdot {}^{=}\underset{\mathcal{R}}{\overset{\alpha}{\longleftarrow}} \cdot {}^{*}\underset{\mathcal{R}}{\overset{\curlyvee\beta}{\longleftarrow}}$$

Proof



$$\mathrel{{}_{\mathcal{R}}{\xleftarrow{\alpha}}} \cdot \xrightarrow{\beta}_{\mathcal{S}} \subseteq \xrightarrow{\curlyvee\alpha}{}^{*}_{\mathcal{S}} \cdot \xrightarrow{\beta}{}^{=}_{\mathcal{S}} \cdot \xrightarrow{\curlyvee\alpha\beta}{}^{*}_{\mathcal{S}} \cdot {}_{\mathcal{R}}^{*}\xleftarrow{\curlyvee\alpha\beta} \cdot {}_{\mathcal{R}}^{=}\xleftarrow{\alpha} \cdot {}_{\mathcal{R}}^{*}\xleftarrow{\curlyvee\beta}$$

Proof



Exercise

- Show that the following TRS commutes with itself

$$\begin{aligned} \mathsf{hd}(x : y) &\to x & \mathsf{tl}(x : y) &\to y \\ \mathsf{nats} &\to 0 : \mathsf{inc}(\mathsf{nats}) & \mathsf{inc}(x : y) &\to s(x) : \mathsf{inc}(y) \\ \mathsf{inc}(\mathsf{tl}(\mathsf{nats})) &\to \mathsf{tl}(\mathsf{inc}(\mathsf{nats})) \end{aligned}$$

Exercise & Solution

- Show that the following TRS commutes with itself

$$\mathsf{hd}(x : y) \xrightarrow{0} x \qquad \mathsf{tl}(x : y) \xrightarrow{0} y$$

$$\mathsf{nats} \xrightarrow{0} 0 : \mathsf{inc}(\mathsf{nats}) \qquad \mathsf{inc}(x : y) \xrightarrow{0} s(x) : \mathsf{inc}(y)$$

$$\mathsf{inc}(\mathsf{tl}(\mathsf{nats})) \xrightarrow{1} \mathsf{tl}(\mathsf{inc}(\mathsf{nats}))$$

- only one CP $\mathsf{tl}(\mathsf{inc}(\mathsf{nats})) \xleftarrow{1} \mathsf{inc}(\mathsf{tl}(\mathsf{nats})) \xrightarrow{0} \mathsf{inc}(\mathsf{tl}(0 : \mathsf{inc}(\mathsf{nats})))$
- can be joined as

$$\begin{aligned} \mathsf{tl}(\mathsf{inc}(\mathsf{nats})) &\xrightarrow{0} \mathsf{tl}(\mathsf{inc}(0 : \mathsf{inc}(\mathsf{nats}))) \\ &\xrightarrow{0} \mathsf{tl}(\mathsf{s}(0) : \mathsf{inc}(\mathsf{inc}(\mathsf{nats}))) \\ &\xrightarrow{0} \mathsf{inc}(\mathsf{inc}(\mathsf{nats})) \\ &\xleftarrow{0} \mathsf{inc}(\mathsf{tl}(0 : \mathsf{inc}(\mathsf{nats}))) \end{aligned}$$

Critical Peak Steps

- What about non-terminating non-right-linear systems?

Critical Peak Steps

- What about non-terminating non-right-linear systems?

Definition

Set of critical peak steps of $\mathcal{S}$ for $\mathcal{R}$ is defined as

$$\mathrm{CPS}_{\mathcal{R}}(\mathcal{S}) = \{s \to u \mid t \;{}_{\mathcal{R}}\!\leftarrow s \to_{\mathcal{S}} u \text{ is a critical peak}\}$$

Critical Peak Steps

- What about non-terminating non-right-linear systems?

Definition

Set of critical peak steps of $\mathcal{S}$ for $\mathcal{R}$ is defined as

$$\mathsf{CPS}_{\mathcal{R}}(\mathcal{S}) = \{ s \to u \mid t \;{}_{\mathcal{R}}\!\leftarrow s \to_{\mathcal{S}} u \text{ is a critical peak} \}$$

Theorem (Hirokawa, Middeldorp)

Left-linear locally commuting TRSs $\mathcal{R}$ and $\mathcal{S}$ commute if $\mathsf{CPS}_{\mathcal{S}}(\mathcal{R}) \cup \mathsf{CPS}_{\mathcal{R}}(\mathcal{S})$ *is relatively terminating over* $\mathcal{R} \cup \mathcal{S}$

Critical Peak Steps

- What about non-terminating non-right-linear systems?

Definition

Set of critical peak steps of $\mathcal{S}$ for $\mathcal{R}$ is defined as

$$\mathrm{CPS}_{\mathcal{R}}(\mathcal{S}) = \{s \to u \mid t \;{}_{\mathcal{R}}\!\leftarrow s \to_{\mathcal{S}} u \text{ is a critical peak}\}$$

Theorem (Hirokawa, Middeldorp)

Left-linear locally commuting TRSs $\mathcal{R}$ and $\mathcal{S}$ commute if $\mathrm{CPS}_{\mathcal{S}}(\mathcal{R}) \cup \mathrm{CPS}_{\mathcal{R}}(\mathcal{S})$ is relatively terminating over $\mathcal{R} \cup \mathcal{S}$

Lemma

Let $\mathcal{R}$ and $\mathcal{S}$ be left-linear TRSs. If $t \;{}_{\mathcal{R}}\!\leftarrow\!\!\!\!\!\ominus\; s \;\ominus\!\!\!\!\!\rightarrow_{\mathcal{S}} u$ then $t \;\ominus\!\!\!\!\!\rightarrow_{\mathcal{S}} \cdot \;{}_{\mathcal{R}}\!\leftarrow\!\!\!\!\!\ominus\; u$ or $t \;{}_{\mathcal{R}}\!\leftarrow\!\!\!\!\!\ominus\; \cdot \;{}_{\mathrm{CPS}_{\mathcal{S}}(\mathcal{R})}\!\leftarrow s \to_{\mathrm{CPS}_{\mathcal{R}}(\mathcal{S})} \cdot \;\ominus\!\!\!\!\!\rightarrow_{\mathcal{S}} u$

Proof

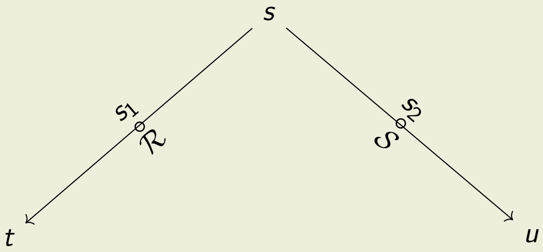
- use predecessor labeling: define $t \overset{s}{\leftrightarrow}_{\mathcal{R}} u$ if $s \to^{*}_{\mathcal{R} \cup \mathcal{S}} t \leftrightarrow_{\mathcal{R}} u$ (and for $\mathcal{S}$)

Proof

- use predecessor labeling: define $t \overset{s}{\nrightarrow}_{\mathcal{R}} u$ if $s \to^*_{\mathcal{R}\cup\mathcal{S}} t \nrightarrow_{\mathcal{R}} u$ (and for $\mathcal{S}$)
- labels are compared using $\succ = \to^+_{(\mathsf{CPS}_{\mathcal{R}}(\mathcal{S})\cup\mathsf{CPS}_{\mathcal{S}}(\mathcal{R}))/(\mathcal{R}\cup\mathcal{S})}$

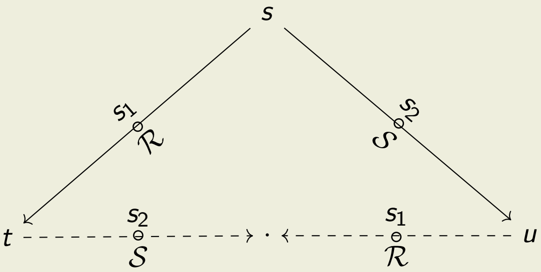
Proof

- use predecessor labeling: define $t \overset{s}{\rightarrow\!\!\!\!\circ}_{\mathcal{R}} u$ if $s \to^*_{\mathcal{R}\cup\mathcal{S}} t \rightarrow\!\!\!\!\circ_{\mathcal{R}} u$ (and for $\mathcal{S}$)
- labels are compared using $\succ = \to^+_{(\mathsf{CPS}_{\mathcal{R}}(\mathcal{S})\cup\mathsf{CPS}_{\mathcal{S}}(\mathcal{R}))/(\mathcal{R}\cup\mathcal{S})}$
- show decreasingness of $\rightarrow\!\!\!\!\circ$: assume $t \;{}_{\mathcal{R}}\overset{s_1}{\leftarrow\!\!\!\!\circ}\; s \overset{s_2}{\rightarrow\!\!\!\!\circ}_{\mathcal{S}} u$



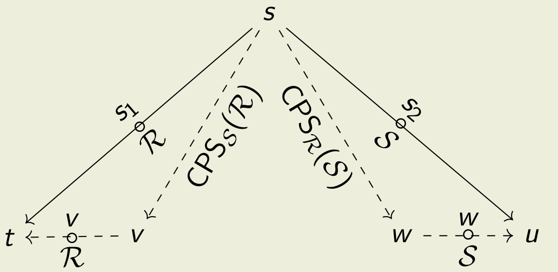
Proof

- use predecessor labeling: define $t \xrightarrow{s}_{\mathcal{R}} u$ if $s \to^{*}_{\mathcal{R}\cup\mathcal{S}} t \to_{\mathcal{R}} u$ (and for $\mathcal{S}$)
- labels are compared using $\succ \; = \; \to^{+}_{(\mathsf{CPS}_{\mathcal{R}}(\mathcal{S})\cup\mathsf{CPS}_{\mathcal{S}}(\mathcal{R}))/(\mathcal{R}\cup\mathcal{S})}$
- show decreasingness of $\to$: assume $t \;{}_{\mathcal{R}}\!\xleftarrow{s_1} s \xrightarrow{s_2}_{\mathcal{S}} u$



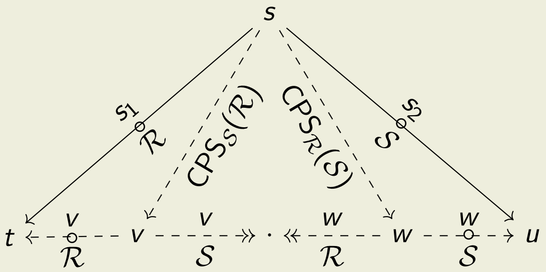
Proof

- use predecessor labeling: define $t \overset{s}{\circ\!\!\!\to}_{\mathcal{R}} u$ if $s \to^*_{\mathcal{R}\cup\mathcal{S}} t \circ\!\!\!\to_{\mathcal{R}} u$ (and for $\mathcal{S}$)
- labels are compared using $\succ \; = \; \to^+_{(\mathsf{CPS}_{\mathcal{R}}(\mathcal{S})\cup\mathsf{CPS}_{\mathcal{S}}(\mathcal{R}))/(\mathcal{R}\cup\mathcal{S})}$
- show decreasingness of $\circ\!\!\!\to$: assume $t \; {}_{\mathcal{R}}\overset{s_1}{\leftarrow\!\!\!\circ} \; s \; \overset{s_2}{\circ\!\!\!\to}_{\mathcal{S}} \; u$



Proof

- use predecessor labeling: define $t \overset{s}{-\!\!\circ\!\!\to}_{\mathcal{R}} u$ if $s \to^*_{\mathcal{R} \cup \mathcal{S}} t -\!\!\circ\!\!\to_{\mathcal{R}} u$ (and for $\mathcal{S}$)
- labels are compared using $\succ \,=\, \to^+_{(\mathsf{CPS}_{\mathcal{R}}(\mathcal{S}) \cup \mathsf{CPS}_{\mathcal{S}}(\mathcal{R}))/(\mathcal{R} \cup \mathcal{S})}$
- show decreasingness of $-\!\!\circ\!\!\to$: assume $t \;{}_{\mathcal{R}}\!\overset{s_1}{\leftarrow\!\!\circ\!\!-}\; s \overset{s_2}{-\!\!\circ\!\!\to}_{\mathcal{S}} u$



Program Transformations

- refactoring
- optimizing compilers
- ...

Program Transformations

- refactoring
- optimizing compilers
- ...

```
if (Value *LHSV = dyn_castNegVal(LHS)) {
  if (!isa<Constant>(RHS))
    if (Value *RHSV = dyn_castNegVal(RHS)) {
      Value *NewAdd = Builder->CreateAdd(LHSV, RHSV, "sum");
      return BinaryOperator::CreateNeg(NewAdd);
    }

  return BinaryOperator::CreateSub(RHS, LHSV);
}
```

Program Transformations

- refactoring
- optimizing compilers
- ...

```
%z = sub 0, %x
%r = add %z, %y
   =>
%r = sub %y, %x
```

```
%z = sub 0, %x
%w = sub 0, %y
%r = add %z, %w
   =>
%v = add %x, %y
%r = sub 0, %v
```

Program Transformations

- refactoring
- optimizing compilers
- ...

$$(0 - x) + y \to y - x \qquad (0 - x) + (0 - y) \to 0 - (x + y)$$

Meaning Preservation

- when are two programs equivalent – when does a transformation preserve semantics?

Meaning Preservation

- when are two programs equivalent – when does a transformation preserve semantics?
- assume a rewriting semantics given by complete TRS $\mathcal{S}$
- transformations $\mathcal{T}$ preserve meaning of program t if for all ground contexts C and ground substitutions σ

$$C[t\sigma]$$

Meaning Preservation

- when are two programs equivalent – when does a transformation preserve semantics?
- assume a rewriting semantics given by complete TRS $\mathcal{S}$
- transformations $\mathcal{T}$ preserve meaning of program t if for all ground contexts C and ground substitutions σ

$$C[t\sigma] \xrightarrow[\mathcal{S}]{!} v$$

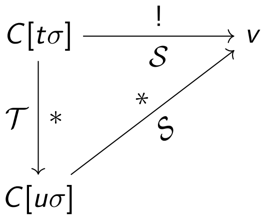
Meaning Preservation

- when are two programs equivalent – when does a transformation preserve semantics?
- assume a rewriting semantics given by complete TRS $\mathcal{S}$
- transformations $\mathcal{T}$ preserve meaning of program t if for all ground contexts C and ground substitutions σ

$$\begin{array}{ccc} C[t\sigma] & \xrightarrow[\mathcal{S}]{!} & v \\ \mathcal{T}\Big\downarrow * & & \\ C[u\sigma] & & \end{array}$$

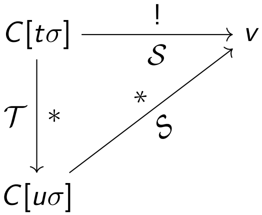
Meaning Preservation

- when are two programs equivalent – when does a transformation preserve semantics?
- assume a rewriting semantics given by complete TRS $\mathcal{S}$
- transformations $\mathcal{T}$ preserve meaning of program t if for all ground contexts C and ground substitutions σ



Meaning Preservation

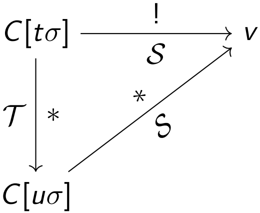
- when are two programs equivalent – when does a transformation preserve semantics?
- assume a rewriting semantics given by complete TRS $\mathcal{S}$
- transformations $\mathcal{T}$ preserve meaning of program t if for all ground contexts C and ground substitutions σ



- show that $\mathcal{S}$ and $\mathcal{T}$ commute and ground $\mathcal{S}$-normal-forms are $\mathcal{T}$-normal-forms

Meaning Preservation

- when are two programs equivalent – when does a transformation preserve semantics?
- assume a rewriting semantics given by complete TRS $\mathcal{S}$
- transformations $\mathcal{T}$ preserve meaning of program t if for all ground contexts C and ground substitutions σ



- show that $\mathcal{S}$ and $\mathcal{T} \cup \mathcal{S}$ commute and ground $\mathcal{S}$-normal-forms are $\mathcal{T}$-normal-forms

Example

- Semantics $\mathcal{S}$

$$
\begin{array}{ccc}
\mathsf{s}(\mathsf{p}(x)) \to x & \mathsf{p}(\mathsf{s}(x)) \to x & x - 0 \to x \\
x - \mathsf{s}(y) \to \mathsf{p}(x - y) & x - \mathsf{p}(y) \to \mathsf{s}(x - y) & \mathsf{p}(x) - y \to \mathsf{p}(x - y) \\
\mathsf{s}(x) - y \to \mathsf{s}(x - y) & x + 0 \to x & 0 + x \to x \\
x + \mathsf{s}(y) \to \mathsf{s}(x + y) & x + \mathsf{p}(y) \to \mathsf{p}(x + y) & \mathsf{s}(x) + y \to \mathsf{s}(x + y) \\
\mathsf{p}(x) + y \to \mathsf{p}(x + y) & &
\end{array}
$$

- Transformation $\mathcal{T}$

$$(0 - x) + (0 - y) \to 0 - (x + y)$$

Example

- Semantics $\mathcal{S}$

$$
\begin{array}{ccc}
\mathsf{s}(\mathsf{p}(x)) \to x & \mathsf{p}(\mathsf{s}(x)) \to x & x - 0 \to x \\
x - \mathsf{s}(y) \to \mathsf{p}(x - y) & x - \mathsf{p}(y) \to \mathsf{s}(x - y) & \mathsf{p}(x) - y \to \mathsf{p}(x - y) \\
\mathsf{s}(x) - y \to \mathsf{s}(x - y) & x + 0 \to x & 0 + x \to x \\
x + \mathsf{s}(y) \to \mathsf{s}(x + y) & x + \mathsf{p}(y) \to \mathsf{p}(x + y) & \mathsf{s}(x) + y \to \mathsf{s}(x + y) \\
\mathsf{p}(x) + y \to \mathsf{p}(x + y) & &
\end{array}
$$

- Transformation $\mathcal{T}$

$$(0 - x) + y \to y - x$$

GHC's Rewrite Rules

Haskell

```
map f [] = []
map f (h:t) = f h : map f t
```

GHC's Rewrite Rules

Haskell

```
map f [] = []
map f (h:t) = f h : map f t

{-# RULES
  "map/map" forall f g xs.
     map f (map g xs) = map (f . g) xs
  #-}
```

GHC's Rewrite Rules

Haskell

```
map f [] = []
map f (h:t) = f h : map f t

{-# RULES
  "map/map" forall f g xs.
     map f (map g xs) = map (f . g) xs
  #-}
```

- optimization of Haskell programs using rewrite rules
- library authors can use rules to express domain-specific optimizations that the compiler cannot discover for itself
- simple, but effective in optimizing real programs

As Higher-Order Rewrite System

- Semantics $\mathcal{S}$

$$\begin{aligned} \mathsf{map}\ (\lambda x.\ F\ x)\ \mathsf{nil} &\to \mathsf{nil} \\ \mathsf{map}\ (\lambda x.\ F\ x)\ (\mathsf{cons}\ h\ t) &\to \mathsf{cons}\ (F\ h)\ (\mathsf{map}\ (\lambda x.\ F\ x)\ t) \end{aligned}$$

- Transformation $\mathcal{T}$

$$\mathsf{map}\ (\lambda x.\ F\ x)\ (\mathsf{map}\ (\lambda x.\ G\ x)\ xs) \to \mathsf{map}\ (\lambda x.\ F\ (G\ x))\ xs$$

As Higher-Order Rewrite System

- Semantics $\mathcal{S}$

$$\mathsf{map}\ (\lambda x.\, F\ x)\ \mathsf{nil} \to \mathsf{nil}$$
$$\mathsf{map}\ (\lambda x.\, F\ x)\ (\mathsf{cons}\ h\ t) \to \mathsf{cons}\ (F\ h)\ (\mathsf{map}\ (\lambda x.\, F\ x)\ t)$$

- Transformation $\mathcal{T}$

$$\mathsf{map}\ (\lambda x.\, F\ x)\ (\mathsf{map}\ (\lambda x.\, G\ x)\ xs) \to \mathsf{map}\ (\lambda x.\, F\ (G\ x))\ xs$$

- Critical Pairs

$$\mathsf{map}\ (\lambda x.\, F\ x)\ \mathsf{nil}\ {}_{\mathcal{S}}{\leftarrow}\!\rtimes\!{\rightarrow}_{\mathcal{T}}\ \mathsf{map}\ (\lambda x.\, F\ (G\ x))\ \mathsf{nil}$$
$$\mathsf{map}\ (\lambda x.\, F\ x)\ (\mathsf{cons}\ (G\ h)\ (\mathsf{map}\ (\lambda x.\, G\ x)\ t))\ {}_{\mathcal{S}}{\leftarrow}\!\rtimes\!{\rightarrow}_{\mathcal{T}}$$
$$\mathsf{map}\ (\lambda x.\, F\ (G\ x))\ (\mathsf{cons}\ h\ t)$$

Exercises

Show that the following transformations are meaning preserving

- $x - (0 - y) \to x + y$
- $\mathrm{and}(\mathrm{eq}(x, 0), \mathrm{eq}(y, 0)) \to \mathrm{eq}(\mathrm{or}(x, y), 0)$ – on Booleans, make up your own semantics

Haskell

```
  foldr f n [] = n
  foldr f n (x : xs) = f x (foldr f n xs)
  sum [] = 0
  sum (x : xs) = x + sum xs

{-# RULES
  "sum/foldr" sum xs  = foldr (+) 0 xs
  #-}
```