

Delta-Rules

A set of rules δ of the form $c t_1 \dots t_n \rightarrow r$

with $c \in \mathcal{C}$, $t_1, \dots, t_n, r \in \Lambda$ is a *set of Delta-rules* iff

- t_1, \dots, t_n, r are closed lambda-terms
- all t_i are in \rightarrow_{β} -normal form and they do not contain the left-hand side of a rule from δ
- δ does not contain rules $c t_1 \dots t_n \rightarrow r$ and $c t_1 \dots t_m \rightarrow r'$ with $m \geq n$

δ -Reduction

- $l \rightarrow_{\delta} r$, if $l \rightarrow r \in \delta$
- $t_1 \rightarrow_{\delta} t_2$ implies $(t_1 r) \rightarrow_{\delta} (t_2 r)$, $(r t_1) \rightarrow_{\delta} (r t_2)$, $\lambda y. t_1 \rightarrow_{\delta} \lambda y. t_2$

We define $\rightarrow_{\beta\delta} = \rightarrow_{\beta} \cup \rightarrow_{\delta}$.

Non-Terminating Reductions

- Even β -reduction does not terminate:

$$(\lambda x. x x) (\lambda x. x x) \rightarrow_{\beta} (\lambda x. x x) (\lambda x. x x) \rightarrow_{\beta} \dots$$

- In general, termination depends on the reduction strategy.

– Leftmost outermost reduction:

$$(\lambda x. y) ((\lambda x. x x) (\lambda x. x x)) \rightarrow_{\beta} y,$$

– Leftmost innermost reduction:

$$(\lambda x. y) ((\lambda x. x x) (\lambda x. x x)) \rightarrow_{\beta} (\lambda x. y) ((\lambda x. x x) (\lambda x. x x)) \rightarrow_{\beta} \dots$$

Translation of Simple HASKELL into Lambda Terms

\mathcal{C}_0 = pre-defined function symbols (+, not, sqrt, etc.)

Con = constructor symbols (including Int, Float, Char)

$\mathcal{C} = \mathcal{C}_0 \cup \text{Con} \cup \{\text{tuple}_n \mid n \in \{0, 2, 3, \dots\}\} \cup \{\text{if}, \text{fix}\}$

$\mathcal{L}am(\underline{\text{var}})$	= $\underline{\text{var}}$
$\mathcal{L}am(c)$	= c , where $c \in \mathcal{C}_0 \cup \text{Con}$
$\mathcal{L}am((\underline{\text{exp}}_1, \dots, \underline{\text{exp}}_n))$	= $\text{tuple}_n \mathcal{L}am(\underline{\text{exp}}_1) \dots \mathcal{L}am(\underline{\text{exp}}_n)$, where $n \in \{0, 2, 3, \dots\}$
$\mathcal{L}am((\underline{\text{exp}}))$	= $\mathcal{L}am(\underline{\text{exp}})$
$\mathcal{L}am((\underline{\text{exp}}_1 \underline{\text{exp}}_2))$	= $(\mathcal{L}am(\underline{\text{exp}}_1) \mathcal{L}am(\underline{\text{exp}}_2))$
$\mathcal{L}am(\text{if } \underline{\text{exp}}_1 \text{ then } \underline{\text{exp}}_2 \text{ else } \underline{\text{exp}}_3)$	= $\text{if } \mathcal{L}am(\underline{\text{exp}}_1) \mathcal{L}am(\underline{\text{exp}}_2) \mathcal{L}am(\underline{\text{exp}}_3)$
$\mathcal{L}am(\text{let } \underline{\text{var}} = \underline{\text{exp}} \text{ in } \underline{\text{exp}}')$	= $\mathcal{L}am(\underline{\text{exp}}') [\underline{\text{var}} / (\text{fix } (\lambda \underline{\text{var}}. \mathcal{L}am(\underline{\text{exp}})))]$
$\mathcal{L}am(\backslash \underline{\text{var}} \rightarrow \underline{\text{exp}})$	= $\lambda \underline{\text{var}}. \mathcal{L}am(\underline{\text{exp}})$

δ -Rules for HASKELL-Programs

Con_n = constructor symbols of arity n

δ_0 = rules for pre-defined symbols in HASKELL

$$\begin{aligned} \delta = & \delta_0 \cup \\ & \{\text{bot} \rightarrow \text{bot}, \\ & \text{isa}_{()} \text{tuple}_0 \rightarrow \text{True}, \\ & \text{if True} \rightarrow \lambda xy.x, \\ & \text{if False} \rightarrow \lambda xy.y, \\ & \text{fix} \rightarrow \lambda f.f(\text{fix } f)\} \cup \\ & \{\text{isa}_{\text{constr}}(\text{constr } t_1 \dots t_n) \rightarrow \text{True} \mid \text{constr} \in \text{Con}_n, t_j \text{ closed}\} \cup \\ & \{\text{isa}_{\text{constr}}(\text{constr}' t_1 \dots t_m) \rightarrow \text{False} \mid \text{constr} \neq \text{constr}' \in \text{Con}_m, t_j \text{ closed}\} \cup \\ & \{\text{argof}_{\text{constr}}(\text{constr } t_1 \dots t_n) \rightarrow \text{tuple}_n t_1 \dots t_n \mid \text{constr} \in \text{Con}_n, \\ & \qquad \qquad \qquad n \in \{0, 2, 3, \dots\}, t_j \text{ closed}\} \cup \\ & \{\text{argof}_{\text{constr}}(\text{constr } t) \rightarrow t \mid \text{constr} \in \text{Con}_1, t \text{ closed}\} \cup \\ & \{\text{sel}_{n,i}(\text{tuple}_n t_1 \dots t_n) \rightarrow t_i \mid n \geq 2, 1 \leq i \leq n, t_j \text{ closed}\} \end{aligned}$$