Delta-Rules

A set of rules δ of the form $c\,t_1 \ldots t_n \;\rightarrow\; r$
with $c \in \mathcal{C},\;\; t_1, \ldots, t_n,\;\; r \in \Lambda$ is a *set of Delta-rules* iff

- $t_1, \ldots, t_n, r$ are closed lambda-terms
- all t_i are in $\rightarrow_\beta$-normal form and they do not contain the left-hand side of a rule from δ
- δ does not contain rules $c\,t_1 \ldots t_n \;\rightarrow\; r$ and $c\,t_1 \ldots t_m \;\rightarrow\; r'$ 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:

$$(\lambda x.x\,x)\,(\lambda x.x\,x) \ \rightarrow_\beta \ (\lambda x.x\,x)\,(\lambda x.x\,x) \ \rightarrow_\beta \ \ldots$$

- 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:

$$(\lambda x.y)((\lambda x.x\,x)\,(\lambda x.x\,x)) \ \rightarrow_\beta \ (\lambda x.y)((\lambda x.x\,x)\,(\lambda x.x\,x)) \ \rightarrow_\beta \ \ldots$$

Translation of Simple HASKELL into Lambda Terms

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

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

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

$$
\begin{array}{lcl}
\mathcal{L}am(\underline{\mathsf{var}}) & = & \underline{\mathsf{var}} \\
\mathcal{L}am(c) & = & c, \quad \text{where } c \in \mathcal{C}_0 \cup \mathsf{Con} \\
\mathcal{L}am((\underline{\mathsf{exp}}_1, \dots, \underline{\mathsf{exp}}_n)) & = & \mathtt{tuple}_n\, \mathcal{L}am(\underline{\mathsf{exp}}_1) \dots \mathcal{L}am(\underline{\mathsf{exp}}_n), \\
 & & \qquad \text{where } n \in \{0, 2, 3, \dots\} \\
\mathcal{L}am((\underline{\mathsf{exp}})) & = & \mathcal{L}am(\underline{\mathsf{exp}}) \\
\mathcal{L}am((\underline{\mathsf{exp}}_1\ \underline{\mathsf{exp}}_2)) & = & (\mathcal{L}am(\underline{\mathsf{exp}}_1)\ \mathcal{L}am(\underline{\mathsf{exp}}_2)) \\
\mathcal{L}am(\mathtt{if}\ \underline{\mathsf{exp}}_1\ \mathtt{then}\ \underline{\mathsf{exp}}_2\ \mathtt{else}\ \underline{\mathsf{exp}}_3) & = & \mathtt{if}\ \mathcal{L}am(\underline{\mathsf{exp}}_1)\ \mathcal{L}am(\underline{\mathsf{exp}}_2)\ \mathcal{L}am(\underline{\mathsf{exp}}_3) \\
\mathcal{L}am(\mathtt{let}\ \underline{\mathsf{var}} = \underline{\mathsf{exp}}\ \mathtt{in}\ \underline{\mathsf{exp}}') & = & \mathcal{L}am(\underline{\mathsf{exp}}')\ [\underline{\mathsf{var}} \,/\, (\mathtt{fix}\ (\lambda \underline{\mathsf{var}}.\mathcal{L}am(\underline{\mathsf{exp}})))] \\
\mathcal{L}am(\backslash \underline{\mathsf{var}}\ \mathtt{->}\ \underline{\mathsf{exp}}) & = & \lambda \underline{\mathsf{var}}.\mathcal{L}am(\underline{\mathsf{exp}})
\end{array}
$$

δ-Rules for HASKELL-Programs

Con_n = constructor symbols of arity n

δ_0 = rules for pre-defined symbols in HASKELL

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