Transformation Into Simple HASKELL-Programs

(1) Transformation of function declarations into pattern declarations

$$\frac{\underline{\mathsf{var}}\ \underline{\mathsf{pat}}_1^1\ \ldots\ \underline{\mathsf{pat}}_n^1\ =\ \underline{\mathsf{exp}}^1;\ \ldots;\ \underline{\mathsf{var}}\ \underline{\mathsf{pat}}_1^k\ \ldots\ \underline{\mathsf{pat}}_n^k\ =\ \underline{\mathsf{exp}}^k}{\begin{array}{l}\underline{\mathsf{var}}\ =\ \backslash\ \mathtt{x}_1\ \ldots\ \mathtt{x}_n\ \texttt{->}\ \texttt{case}\ (\mathtt{x}_1,\ldots,\mathtt{x}_n)\ \texttt{of}\ \{(\underline{\mathsf{pat}}_1^1,\ldots,\underline{\mathsf{pat}}_n^1)\ \texttt{->}\ \underline{\mathsf{exp}}^1;\\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\vdots\\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (\underline{\mathsf{pat}}_1^k,\ldots,\underline{\mathsf{pat}}_n^k)\ \texttt{->}\ \underline{\mathsf{exp}}^k\}\end{array}}$$

where $\mathtt{x}_1,\ldots,\mathtt{x}_n$ must be new variables and $n > 0$

(2) Transforming sequences of declarations into a single declaration

$$\frac{\{\underline{\mathsf{var}}_1\ =\ \underline{\mathsf{exp}}_1;\ \ldots;\ \underline{\mathsf{var}}_n\ =\ \underline{\mathsf{exp}}_n\}}{(\underline{\mathsf{var}}_1,\ldots,\underline{\mathsf{var}}_n)\ =\ (\underline{\mathsf{exp}}_1,\ldots,\underline{\mathsf{exp}}_n)}\quad \text{where } n \geq 2$$

(3) Transforming lambda-expressions with several patterns

$$\frac{\backslash\ \underline{\mathsf{pat}}_1\ \ldots\ \underline{\mathsf{pat}}_n\ \texttt{->}\ \underline{\mathsf{exp}}}{\backslash\ \underline{\mathsf{pat}}_1\ \texttt{->}\ (\backslash\ \underline{\mathsf{pat}}_2\ \texttt{->}\ \ldots\ (\backslash\ \underline{\mathsf{pat}}_n\ \texttt{->}\ \underline{\mathsf{exp}})\ldots)}\quad \text{where } n \geq 2$$

(4) Transforming lambda-patterns into `case`

$$\frac{\backslash\ \underline{\mathsf{pat}}\ \texttt{->}\ \underline{\mathsf{exp}}}{\backslash\ \underline{\mathsf{var}}\ \texttt{->}\ \texttt{case}\ \underline{\mathsf{var}}\ \texttt{of}\ \underline{\mathsf{pat}}\ \texttt{->}\ \underline{\mathsf{exp}}}\quad \text{where } \underline{\mathsf{pat}} \text{ is no variable, } \underline{\mathsf{var}} \text{ is new}$$

(5) Transforming `case` into `match`

$$\frac{\texttt{case}\ \underline{\mathsf{exp}}\ \texttt{of}\ \{\underline{\mathsf{pat}}_1\ \texttt{->}\ \underline{\mathsf{exp}}_1;\ldots\underline{\mathsf{pat}}_n\ \texttt{->}\ \underline{\mathsf{exp}}_n\}}{\begin{array}{l}\texttt{match}\ \underline{\mathsf{pat}}_1\ \underline{\mathsf{exp}}\ \underline{\mathsf{exp}}_1\\ \qquad (\texttt{match}\ \underline{\mathsf{pat}}_2\ \underline{\mathsf{exp}}\ \underline{\mathsf{exp}}_2\\ \qquad\qquad \ldots\\ \qquad\qquad\quad (\texttt{match}\ \underline{\mathsf{pat}}_n\ \underline{\mathsf{exp}}\ \underline{\mathsf{exp}}_n\ \ \texttt{bot})\ldots)\end{array}}$$

(6) `match` of variables

$$\frac{\texttt{match}\ \underline{\mathsf{var}}\ \underline{\mathsf{exp}}\ \underline{\mathsf{exp}}_1\ \underline{\mathsf{exp}}_2}{(\backslash\ \underline{\mathsf{var}}\ \texttt{->}\ \underline{\mathsf{exp}}_1)\ \underline{\mathsf{exp}}}$$

(7) `match` of joker pattern

$$\frac{\texttt{match}\ _\ \underline{\mathsf{exp}}\ \underline{\mathsf{exp}}_1\ \underline{\mathsf{exp}}_2}{\underline{\mathsf{exp}}_1}$$

(8) `match` of constructors

$$\frac{\texttt{match}\ (\underline{\mathsf{constr}}\ \underline{\mathsf{pat}}_1\ \ldots\ \underline{\mathsf{pat}}_n)\ \underline{\mathsf{exp}}\ \underline{\mathsf{exp}}_1\ \underline{\mathsf{exp}}_2}{\begin{array}{l}\texttt{if}\ (\mathsf{isa}_{\underline{\mathsf{constr}}}\ \underline{\mathsf{exp}})\ \texttt{then}\ (\texttt{match}\ \ (\underline{\mathsf{pat}}_1,\ldots,\underline{\mathsf{pat}}_n)\\ \qquad\qquad\qquad\qquad\qquad\qquad (\mathsf{argof}_{\underline{\mathsf{constr}}}\ \underline{\mathsf{exp}})\ \underline{\mathsf{exp}}_1\ \underline{\mathsf{exp}}_2)\ \texttt{else}\ \underline{\mathsf{exp}}_2\end{array}}$$

(9) `match` of empty tuple

$$\frac{\texttt{match}\ ()\ \underline{\mathsf{exp}}\ \underline{\mathsf{exp}}_1\ \underline{\mathsf{exp}}_2}{\texttt{if}\ (\mathsf{isa}_{()}\ \underline{\mathsf{exp}})\ \texttt{then}\ \underline{\mathsf{exp}}_1\ \texttt{else}\ \underline{\mathsf{exp}}_2}$$

(10) `match` of non-empty tuple

$$\frac{\texttt{match}\ (\underline{\mathsf{pat}}_1,\ldots,\underline{\mathsf{pat}}_n)\ \underline{\mathsf{exp}}\ \underline{\mathsf{exp}}_1\ \underline{\mathsf{exp}}_2}{\begin{array}{l}\texttt{match}\ \underline{\mathsf{pat}}_1\ (\mathtt{sel}_{n,1}\ \underline{\mathsf{exp}})\\ \qquad (\texttt{match}\ \underline{\mathsf{pat}}_2\ (\mathtt{sel}_{n,2}\ \underline{\mathsf{exp}})\\ \qquad\qquad \ldots\ (\texttt{match}\ \underline{\mathsf{pat}}_n\ (\mathtt{sel}_{n,n}\ \underline{\mathsf{exp}})\ \underline{\mathsf{exp}}_1\ \underline{\mathsf{exp}}_2)\ \ldots\\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \underline{\mathsf{exp}}_2)\ \ \underline{\mathsf{exp}}_2\end{array}}\quad \text{where } n \geq 2$$

(11) Declaration of several variables

$$\frac{\texttt{let}\ (\underline{\mathsf{var}}_1,\ldots,\underline{\mathsf{var}}_n) = \underline{\mathsf{exp}}\ \texttt{in}\ \underline{\mathsf{exp}}'}{\texttt{let}\ \underline{\mathsf{var}} = \texttt{match}\ (\underline{\mathsf{var}},\ldots,\underline{\mathsf{var}}_n)\ \underline{\mathsf{var}}\ \underline{\mathsf{exp}}\ \texttt{bot}\quad \texttt{in}\quad \texttt{match}\ (\underline{\mathsf{var}},\ldots,\underline{\mathsf{var}}_n)\ \underline{\mathsf{var}}\ \underline{\mathsf{exp}}'\ \texttt{bot}}$$

where $n \geq 2$ and $\underline{\mathsf{var}}$ is a new variable