CIS 670 Homework 4, Due Nov 6, 2002

1. Show the dynamic correctness of Leroy's boxing translation (for terms of base type, such as *int* and *float*).

2. Compare *typecase* with type *Dynamic* to *typerec* in $\lambda_i^{ML}$ in terms of expressiveness. Give some examples. Make some observations.

   Recall the formation rule for *typecase* is:

$$\frac{\begin{array}{l}\Delta\Gamma \vdash e : \mathit{Dynamic} \\ \alpha_{i1}\ldots\alpha_{im}\Gamma, x_i : \tau_i \vdash e_i : \tau \qquad (\alpha_{i1}\ldots\alpha_{im} \notin \tau, 1 \leq i \leq n) \\ \Delta\Gamma \vdash e' : \tau\end{array}}{\Delta\Gamma \vdash \mathit{typecase}\ \ e\ \mathit{of}\, (x_1 : \tau_1) \Rightarrow e_n \ldots (x_n : \tau_n) \Rightarrow e_1 \text{ else } e_{else} : \tau}$$

   While the formation rule for *typerec* is:

$$\frac{\begin{array}{l}\Delta, \alpha : \star \vdash \sigma \\ \Delta \vdash \tau : \star \\ \Delta\Gamma \vdash e_{int} : \sigma[\mathit{int}\,/\alpha] \\ \Delta\Gamma \vdash e_{\rightarrow} : \forall\beta : \star.\forall\gamma : \star.\sigma[\beta/\alpha] \rightarrow \sigma[\gamma/\alpha] \rightarrow \sigma[\beta \rightarrow \gamma/\alpha] \\ \Delta\Gamma \vdash e_{\times} : \forall\beta : \star.\forall\gamma : \star.\sigma[\beta/\alpha] \rightarrow \sigma[\gamma/\alpha] \rightarrow \sigma[\beta \times \gamma/\alpha]\end{array}}{\Delta\Gamma \vdash \mathit{typerec}[\alpha.\sigma]\ \tau\ (e_{int}, e_{\rightarrow}, e_{\times}) : \sigma[\tau/\alpha]}$$

3. Say we were to add array types to $\lambda_i^{ML}$. How would you change the static and dynamic semantics of *typerec*?

4. What would be different if the typing rule for $\lambda_i^{ML}$ *typerec* were:

$$\frac{\begin{array}{l}\Delta \vdash \tau' : \star \rightarrow \star \\ \Delta \vdash \tau : \star \\ \Delta\Gamma \vdash e_{int} : \tau'\ \mathit{int} \\ \Delta\Gamma \vdash e_{\rightarrow} : \forall\beta : \star.\forall\gamma : \star.\forall\gamma : \star.\tau'\alpha \rightarrow \tau'\gamma \rightarrow \tau'(\beta \rightarrow \gamma) \\ \Delta\Gamma \vdash e_{\times} : \forall\beta : \star.\forall\gamma : \star.\forall\gamma : \star.\tau'\alpha \rightarrow \tau'\gamma \rightarrow \tau'(\beta \times \gamma)\end{array}}{\Delta\Gamma \vdash \mathit{typerec}[\tau']\ \tau\ (e_{int}, e_{\rightarrow}, e_{\times}) : \tau'\tau}$$

5. Implement a version of $J$ in $\lambda_i^{ML}$ of type $\forall\gamma : \star \rightarrow \star.\forall\alpha : \star.\forall\beta : Type.(\gamma\alpha) \rightarrow (\gamma\beta)$.

6. Even though $\lambda_i^{ML}$ can encode $J$, because it is a predicative calculus, $J$ is not as powerful in this language. Writing a loop with $J$ requires that it be instantiated with a polymorphic type. Show that we cannot write such a loop in $\lambda_i^{ML}$, by showing that the evaluation of any expression terminates. For this problem, consider the following (slightly) reduced version of $\lambda_i^{ML}$.

$$\begin{array}{lcl}\kappa & ::= & \star \mid \kappa_1 \rightarrow \kappa_2 \\ \tau & ::= & \alpha \mid \mathit{int} \mid \tau_1 \rightarrow \tau_2 \\ \sigma & ::= & \forall\alpha : \kappa.\sigma \mid \tau \\ e & ::= & i \mid x \mid \lambda x : \sigma.e \mid e_1 e_2 \mid \Lambda\alpha : \kappa.e \mid e[\tau] \\ & & \mathit{typerec}[\alpha.\sigma]\ \tau\ (e_{int}, e_{\rightarrow})\end{array}$$

You may assume that the normalization of $\lambda_i^{ML}$ types always terminates. Use the following unary logical relation to describe terminating terms. $\mathcal{C}[\![\sigma]\!]$ is defined over all closed polytypes, while $\mathcal{V}[\![\sigma]\!]$ is only defined over the closed, normal forms of monotypes and polytypes.

$$\begin{array}{lcl}
\mathcal{C}[\![\sigma]\!] & = & \{e \mid\ \vdash e : \sigma \text{ and } \sigma \text{ normalizes to } \sigma' \text{ and } e \to^* v \in \mathcal{V}[\![\sigma']\!]\} \\
 & & \\
\mathcal{V}[\![\mathit{int}]\!] & = & \{i\} \\
\mathcal{V}[\![\tau_1 \to \tau_2]\!] & = & \{v \mid\ \vdash v : \tau_1 \to \tau_2, \text{ and for all } v' \in \mathcal{V}[\![\tau_1]\!], vv' \in \mathcal{C}[\![\tau_2]\!]\} \\
\mathcal{V}[\![\forall\alpha:\kappa.\sigma]\!] & = & \{v \mid\ \vdash v : \forall\alpha:\kappa.\sigma, \text{ and for all} \vdash \tau' : \kappa, v[\tau'] \in \mathcal{C}[\![\sigma[\tau'/\alpha]]\!]\}
\end{array}$$

7. Where does the termination proof break down if we make the following change to $\lambda_i^{ML}$?

   Add universal types (but only quantify over kind type, why?) to the monotypes.

$$\begin{array}{l}
\tau ::= \ldots \mid \forall\alpha:\star.\tau \\
e ::= \ldots \mid \mathit{typerec}[\alpha.\sigma]\ \tau\ (e_{\mathit{int}}, e_{\to}, e_{\forall})
\end{array}$$

   Add a branch for universal types to *typerec*

$$\frac{\begin{array}{l}
\Delta, \alpha:\star \vdash \sigma \\
\Delta \vdash \tau : \star \\
\Delta\Gamma \vdash e_{\mathit{int}} : \sigma[\mathit{int}/\alpha] \\
\Delta\Gamma \vdash e_{\to} : \forall\beta:\star.\forall\gamma:\star.\sigma[\beta/\alpha] \to \sigma[\gamma/\alpha] \to \sigma[\beta\to\gamma/\alpha] \\
\Delta\Gamma \vdash e_{\forall} : \forall\beta:\star\to\star.(\forall\gamma:\star.\sigma[\gamma/\alpha] \to \sigma[\beta\gamma/\alpha]) \to \sigma[\forall\gamma:\star.\beta\gamma/\alpha]
\end{array}}{\Delta\Gamma \vdash \mathit{typerec}[\alpha.\sigma]\ \tau\ (e_{\mathit{int}}, e_{\to}, e_{\forall}) : \sigma[\tau/\alpha]}$$

   New evaluation rule for universal types:

$$\begin{array}{l}
\mathit{typerec}[\alpha.\sigma]\ (\forall\gamma:\star.\tau)\ (e_{\mathit{int}}, e_{\to}, e_{\forall}) \mapsto \\
\quad e_{\forall}[\lambda\gamma:\star.\tau](\Lambda\gamma:\star.\ \mathit{typerec}[\alpha.\sigma]\ (\tau)\ (e_{\mathit{int}}, e_{\to}, e_{\forall}))
\end{array}$$

8. (Extra Credit) The fact that primitive recursion over types does not lead to infinite loops means that type checking $\lambda_i^{ML}$ is decidable. An important part of this process is reducing constructors to their normal forms.

   Show strong normalization for constructor reduction in $\lambda_i^{ML}$. Let $SN(\tau)$ mean that $\tau$ is strongly normalizing. Define the predicates:

$$\begin{array}{rcl}
[\![\star]\!] & = & \{\tau \mid \Delta \vdash \tau : \star \quad SN(\tau)\} \\
[\![\kappa_1 \to \kappa_2]\!] & = & \{\tau \mid \Delta \vdash \tau : \kappa_1 \to \kappa_2 \quad \forall\tau_1 \in [\![\kappa_1]\!], \tau\tau_1 \in [\![\kappa_2]\!]\}
\end{array}$$

   Like all logical relations proofs, we must first show that this unary relation implies the property we want, i.e. that if $\tau \in [\![\kappa]\!]$ then $\tau$ is $SN$. We must then show that all well-typed terms are in this relation, that if $\Delta \vdash \tau : \kappa$ then $\tau \in [\![\kappa]\!]$.

However, unlike the termination proofs, this proof is complicated by the fact that reduction may occur in terms with free variables. Therefore, to prove these two results, we need a number of additional lemmas. In particular, we must also show that:

(a) $\alpha\tau_1\tau_2\ldots\tau_n$ is $SN$ iff each $\tau_i$ is $SN$.

(b) If $\Delta \vdash \alpha : \kappa_1 \to \ldots \to \kappa_n \to \kappa$, $\Delta \vdash \tau_i : \kappa_i$ and $\tau_i$ is $SN$, then $\alpha\tau_1\ldots\tau_n \in [\![\kappa]\!]$.

(c) Say $\Delta, \alpha : \kappa \vdash \tau : \kappa_1 \to \ldots \kappa_n \to \kappa$ and $\tau_i \in [\![\kappa_i]\!]$. If $(\tau[\tau_0/\alpha])\tau_1\ldots\tau_n \in [\![\kappa]\!]$, then $(\lambda\alpha : \kappa_0.\tau)\tau_0\tau_1\ldots\tau_n \in [\![\kappa]\!]$.

(d) Let $\kappa' = \kappa_1 \to \ldots \to \kappa_n \to \kappa$. Say $\Delta \vdash \mathit{Typerec}\,\tau(\tau_{int}, \tau_{\to}) : \kappa'$, $\tau_j \in [\![\kappa_j]\!]$, $\tau \in [\![\kappa]\!]$ and each branch of *Typerec* in the appropriate relation. Then $\mathit{Typerec}\,\tau\ (\tau_{int}, \tau_{\to}) \in [\![\kappa']\!]$.

Instead of writing out the proof each of these lemmas, write an explanation of where each lemma is used (probably in the proof of another lemma).