Problem Set 5 Solutions

Problem 1 State

Do exercise 8.14 (a) and (b) on page 387 of the course text.

Solution:

a. Operational semantics for try.

i. The new configurations are of this form:

\(((\ast\text{trying}\ast\ E_1\ E_2\ S_1), S_2)\)

A \ast\text{trying}\ast\ configuration remembers the store \(S_1\) that was in effect before evaluation of \(E_1\) began, so that in case \(E_1\) evaluates to \#f, \(S_1\) can be restored before proceeding with \(E_2\).

ii. Rewrite rules for try and \ast\text{trying}\ast:\

\[
\begin{align*}
\langle (\text{try} E_1 E_2), S \rangle &\Rightarrow (\langle \ast\text{trying}\ast\ E_1 E_2 S \rangle, S) & \text{[try]} \\
\langle E_1, S \rangle &\Rightarrow (E'_1, S') & \text{[trying-progress]} \\
\langle \ast\text{trying}\ast\ E_1 E_2 S_{save}, S \rangle &\Rightarrow (\ast\text{trying}\ast\ E'_1 E_2 S_{save}, S') & \text{[trying-progress]} \\
\langle \ast\text{trying}\ast\ #f E_2 S_{save}, S \rangle &\Rightarrow (E_2, S_{save}) & \text{[trying-false]} \\
\langle \ast\text{trying}\ast\ #t E_2 S_{save}, S \rangle &\Rightarrow (#t, S) & \text{[trying-true]}
\end{align*}
\]

An alternate approach to handling try is to change the configurations to have the form \(\langle E, S^* \rangle\) rather than \(\langle E, S \rangle\). Then the \(S^*\) can be used as a stack of stores to appropriately handle nested try expressions.

In the past, some people suggested using configurations of the form \(\langle E, S_1, S_2 \rangle\). While these can handle a single try expression, they cannot handle the more general case of nested try expressions.

b. Here is a denotational semantics for try:

\[
\begin{align*}
\mathcal{E}[\langle \text{try} E_1 E_2 \rangle] \\
= \lambda es. \text{ with-boolean&store } (\mathcal{E}[E_1] e s) \\
(\lambda bs'. \text{ if } b \text{ then } ((\text{Value}\to\text{Expressible} (\text{Bool}\to\text{Value} \text{ true})), s') \\
\text{ else } (\mathcal{E}[E_2] e s))
\end{align*}
\]

Note that both \(E_1\) and \(E_2\) are evaluated with respect to the original store \(s\).
Problem 2 More State

a. Give a translation of call-by-value FLAVARK into call-by-value FLICK. You do not need to translate rec.

Solution:

\[
\begin{align*}
T[\lambda] &= \lambda \\
T[i] &= \text{prim } ^{\uparrow} i \\
T[(\text{set! } i e)] &= \text{prim } := i \ T[e] \\
T[(\text{lam } i e)] &= \text{lam } i \ T[e] \\
T[(\text{app } e_1 e_2)] &= \text{app } T[e_1] \ (\text{cell } T[e_2]) \\
T[(\text{cell } e)] &= \text{cell } T[e] \\
T[(\text{pair } e_1 e_2)] &= \text{pair } T[e_1] \ T[e_2] \\
T[(\text{begin } e_1 e_2)] &= \text{begin } T[e_1] \ T[e_2] \\
T[(\text{if } e_1 e_2 e_3)] &= \text{if } T[e_1] \ T[e_2] \ T[e_3] \\
T[(\text{prim } o \ e_1 \ldots \ e_n)] &= \text{prim } o \ T[e_1] \ldots \ T[e_n]
\end{align*}
\]

b. Give a translation of call-by-reference FLAVARK into call-by-value FLICK. You do not need to translate rec.

Solution:

Identical to the above, except that expressions of the form (app \ e i) are handled specially by the following rule:

\[
T[(\text{app } e i)] = \text{app } T[e] \ i
\]

Problem 3 Control

Do exercise 9.11 on page 449 of the course text.

Solution: An update with solutions to this problem will be posted later today.