Exercise 1 (Reduction Relation with Closures)

For the evaluation of lambda terms that is closer to evaluation of programs in functional programming languages, one usually replaces textual substitution $t[v/x]$ with a more lazy approach that records the binding $x \mapsto v$ in an environment. These bindings are used whenever we need the value of a variable $v$.

In this approach abstractions $\lambda x. t$ do not evaluate to themselves, but to a pair $(\lambda x. t)[e]$, where $e$ is the current environment. We call such pairs function closures.

a) Define a big-step reduction relation for the lambda calculus with function closures and environments.

b) Add explicit error handling for the case where the binding of a free variable $v$ cannot be found in the environment. Introduce an explicit value $\text{abort}$ to indicate such an exception in the relation.

Solution

a) First recall the standard $\Rightarrow_{cbv}$ relation:

$$
\lambda x. t \Rightarrow_{cbv} \lambda x. t
$$

$$
s \Rightarrow_{cbv} \lambda x. s' \quad t \Rightarrow_{cbv} v \quad s'[v/x] \Rightarrow_{cbv} w
$$

$$
s \Rightarrow_{cbv} w
$$

Note that there are no rules for variables since the reduction relation only considers closed terms. Now we define the relation for lambda calculus with closures:

$$
e(x) = v \quad e \vdash \lambda x. t \Rightarrow_{cbv} \lambda x. t[e]
$$

$$
e \vdash t_1 \Rightarrow_{cbv} (\lambda x. t)[e'] \quad e \vdash t_2 \Rightarrow_{cbv} v' \quad e' + (x \mapsto v') \vdash t \Rightarrow_{cbv} v
$$

$$
e \vdash t_1 t_2 \Rightarrow_{cbv} v
$$
In the following example empty closures for lambdas are omitted for better readability:

\[(\lambda x. \lambda y. x) \Rightarrow_{cbv} (\lambda x. \lambda y. x)\]

\[(\lambda u. u) \Rightarrow_{cbv} (\lambda u. u)\]

\[e(x) = (\lambda u. u)\]

b) We just need to add rules to propagate errors, and modify the existing rules to ensure that no subexpression evaluates to \texttt{abort}.

\[e(t_1) \Rightarrow_{cbv} \texttt{abort} \quad e(t_2) \Rightarrow_{cbv} v \quad v \neq \texttt{abort} \]

Exercise 2 (Better Translation Algorithm)

Give a variant of the translation algorithm that produces shorter terms. More specifically, define a variant of \(\lambda^* x. t\) that analyzes more precisely where \(x\) actually appears in \(t\).

Solution

\[
\begin{align*}
\lambda^* x. x &= I \\
\lambda^* x. X &= K X & \text{if } x \notin \text{FV}(X) \\
\lambda^* x. X &= X & \text{if } x \notin \text{FV}(X) \\
\lambda^* x. (X Y) &= B X (\lambda^* x. Y) & \text{if } x \notin \text{FV}(X) \land x \in \text{FV}(Y) \\
\lambda^* x. (Y X) &= C (\lambda^* x. Y) X & \text{if } x \notin \text{FV}(X) \land x \in \text{FV}(Y) \\
\lambda^* x. (X Y) &= S (\lambda^* x. X) (\lambda^* x. Y) & \text{if } x \in \text{FV}(X, Y)
\end{align*}
\]

where \(B := S (K S) K\) and \(C := S (B B S) (K K)\). \(B\) and \(C\) fulfill the following properties

\[
\begin{align*}
B X Y Z &\rightarrow^* X (Y Z) \\
C X Y Z &\rightarrow^* X Z Y
\end{align*}
\]
Homework 3 (Proofs with Small-steps and Big-steps)

Let \( \omega := \lambda x. x x \) and

\[ t := (\lambda x. (\lambda x y. x) z y) (\omega \omega ((\lambda x y. x) y)) \]

Prove the following:

a) \( t \Rightarrow_n z \)

b) \( t \rightarrow^3_{cbv} t \)

c) \( t \not\rightarrow^+_\text{cbn} t \)

Homework 4 (More Combinators)

Find combinators \( O \) and \( W \) such that:

\[
\begin{align*}
O & \to^+ O \\
W X Y & \to^* X Y Y
\end{align*}
\]

Homework 5 (Mocking Birds)

Consider a combinatory logic that only provides the basic combinators \( B \) and \( M \) (the “mocking bird”) where:

\[
\begin{align*}
B X Y Z & \to^* X (Y Z) \\
M X & \to^* X X
\end{align*}
\]

Prove the following properties of this logic:

a) For every combinator \( X \), there is a combinator \( Y \) such that \( Y \to^* X Y \).

b) For all combinators \( U \) and \( W \), there exist combinators \( X \) and \( Y \) such that \( Y \to^* U X \) and \( X \to^* W Y \).

Homework 6 (Correctness of the Translation Algorithm)

Show that the translation algorithm given in the tutorial is correct. That is, show that it fulfills the following property:

\( (\lambda^x. X) Y \to^* X[Y/x] \)