CPSC 430/530 Assignment 6 Sample Solution

1 Problem 1

See attached `Prob1.v`

2 Problem 2

1. $e_0 : \forall t, t \to t$

   $e_0 = \Lambda t, \lambda(x : t), x$

2. $e_1 : \forall t_1, \forall t_2, \forall t_3, (t_1 \to t_2) \to (t_2 \to t_3) \to (t_1 \to t_3)$

   $e_1 = \Lambda t_1, \Lambda t_2, \Lambda t_3, \lambda(f_1 : t_1 \to t_2), \lambda(f_2 : t_2 \to t_3), \lambda x, f_2(f_1 x)$

3. $\lambda(x : \forall t, t \to t) x[\tau_0](x[\tau_1]) : \tau_2$

   $\tau_0 = \mathsf{nat} \to \mathsf{nat}, \tau_1 = \mathsf{nat}, \tau_2 = (\forall t, t \to t) \to \mathsf{nat} \to \mathsf{nat}$

4. $\lambda(x : \forall t, t \to t) x[\tau_3] x : \tau_4$

   $\tau_3 = \forall t, t \to t, \tau_4 = (\forall t, t \to t) \to (\forall t, t \to t)$

5. $\lambda(x : \forall t, t \to t) \Lambda t, x[\tau_5](x[t]) : \tau_6$

   $\tau_5 = t \to t, \tau_6 = (\forall t, t \to t) \to \forall t, t \to t$

6. $\lambda(m : \mathsf{nat}), \lambda(n : \mathsf{nat}), \Lambda t, n[t \to t](m[t]) : \tau_7$

   $\tau_7 = \mathsf{nat} \to \mathsf{nat} \to \mathsf{nat}$

3 Problem 3

See attached `Prob3.v`

4 Problem 4

4.1 The Formal Rules

For reader's convenience, we first write down all formal rules of the relevant systems.

Expressions

$$
\begin{aligned}
\mathsf{Typ}\ \tau &::= \mathsf{nat} \\
&\quad\ \mathsf{parr}(\tau_1;\tau_2) \\
\mathsf{Exp}\ e &::= x \\
&\quad\ \mathtt{z} \\
&\quad\ \mathtt{s}(e) \\
&\quad\ \mathtt{ifz}\{e_0; x.e_1\}(e) \\
&\quad\ \mathtt{lam}\{\tau\}(x.e) \\
&\quad\ \mathtt{ap}(e_1;e_2) \\
&\quad\ \mathtt{fix}\{\tau\}(x.e) \\
\mathsf{Val}\ v &::= \mathtt{z} \\
&\quad\ \mathtt{s}(v) \\
&\quad\ \mathtt{lam}\{\tau\}(x.e)
\end{aligned}
$$

Typing

$$\frac{}{\Gamma, x : \tau \vdash x : \tau}$$

$$\frac{}{\Gamma \vdash \mathtt{z} : \mathsf{nat}}$$

$$\frac{\Gamma \vdash e : \mathsf{nat}}{\Gamma \vdash \mathtt{s}(e) : \mathsf{nat}}$$

$$\frac{\Gamma \vdash e : \mathsf{nat} \quad \Gamma \vdash e_0 : \tau \quad \Gamma, x : \mathsf{nat} \vdash e_1 : \tau}{\Gamma \vdash \mathtt{ifz}\{e_0; x.e_1\}(e) : \tau}$$

$$\frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \mathtt{lam}\{\tau_1\}(x.e) : \mathsf{parr}(\tau_1;\tau_2)}$$

$$\frac{\Gamma \vdash e_1 : \mathsf{parr}(\tau_2;\tau) \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash \mathsf{ap}(e_1;e_2) : \tau}$$

$$\frac{\Gamma, x : \tau \vdash e : \tau}{\Gamma \vdash \mathtt{fix}\{\tau\}(x.e) : \tau}$$

Control machine frames

$$\frac{}{\mathtt{s}(\star)\ \mathsf{frame}}$$

$$\frac{}{\mathtt{ifz}\{e_0; x.e_1\}(\star)\ \mathsf{frame}}$$

$$\frac{}{\mathsf{ap}(\star; e_2)\ \mathsf{frame}}$$

Control machine stacks

$$\frac{}{\epsilon\ \mathsf{stack}}$$

$$\frac{f\ \mathsf{frame} \quad k\ \mathsf{stack}}{k; f \quad \mathsf{stack}}$$

Control machine semantics The state of the control machine is always represented by $k \vartriangle e$ where $\vartriangle$ stands for either $\triangleright$ or $\triangleleft$. When $\vartriangle = \triangleleft$, we shall always assume that e Val. We will not repeat this assumption every time we need it.

When the input expression is e, The initial state is $\epsilon \triangleright e$. States of the form $\epsilon \triangleleft e'$ are final states.

$$\overline{k \triangleright \mathtt{z} \mapsto k \triangleleft \mathtt{z}}$$

$$\overline{k \triangleright \mathtt{s}(e) \mapsto k; \mathtt{s}(\star) \triangleright e}$$

$$\overline{k; \mathtt{s}(\star) \triangleleft e \mapsto k \triangleleft \mathtt{s}(e)}$$

$$\overline{k \triangleright \mathtt{ifz}\{e_0; x.e_1\}(e) \mapsto k; \mathtt{ifz}\{e_0; x.e_1\}(\star) \triangleright e}$$

$$\overline{k; \mathtt{ifz}\{e_0; x.e_1\}(\star) \triangleleft \mathtt{z} \mapsto k \triangleright e_0}$$

$$\overline{k; \mathtt{ifz}\{e_0; x.e_1\}(\star) \triangleleft \mathtt{s}(e) \mapsto k \triangleright [e/x]e_1}$$

$$\overline{k \triangleright \mathtt{lam}\{\tau\}(x.e) \mapsto k \triangleleft \mathtt{lam}\{\tau\}(x.e)}$$

$$\overline{k \triangleright \mathtt{ap}(e_1; e_2) \mapsto k; \mathtt{ap}(\star; e_2) \triangleright e_1}$$

$$\overline{k; \mathtt{ap}(\star; e_2) \triangleleft \mathtt{lam}\{\tau\}(x.e) \mapsto k \triangleright [e_2/x]e}$$

$$\overline{k \triangleright \mathtt{fix}\{\tau\}(x.e) \mapsto k \triangleright [\mathtt{fix}\{\tau\}(x.e)/x]e}$$

Contexts

$$\begin{aligned}\mathcal{E} ::= {} & \circ \\ & \mathtt{s}(\mathcal{E}) \\ & \mathtt{ifz}\{e_0; x.e_1\}(\mathcal{E}) \\ & \mathtt{ap}(\mathcal{E}; e_2)\end{aligned}$$

Context application If $\mathcal{E}_1, \mathcal{E}_2$ are contexts and e is an expression, we define $\mathcal{E}_1[\mathcal{E}_2]$ and $\mathcal{E}_1[e]$ as follows.

$$\overline{\circ[e] = e}$$

$$\overline{\circ[\mathcal{E}_2] = \mathcal{E}_2}$$

$$\overline{\mathtt{s}(\mathcal{E}')[e] = \mathtt{s}(\mathcal{E}'[e])}$$

$$\overline{\mathtt{s}(\mathcal{E}')[\mathcal{E}_2] = \mathtt{s}(\mathcal{E}'[\mathcal{E}_2])}$$

$$\overline{\mathtt{ifz}\{e_0; x.e_1\}(\mathcal{E}')[e] = \mathtt{ifz}\{e_0; x.e_1\}(\mathcal{E}'[e])}$$

$$\overline{\mathtt{ifz}\{e_0; x.e_1\}(\mathcal{E}')[\mathcal{E}_2] = \mathtt{ifz}\{e_0; x.e_1\}(\mathcal{E}'[\mathcal{E}_2])}$$

$$\overline{\mathtt{ap}(\mathcal{E}'; e_2)[e] = \mathtt{ap}(\mathcal{E}'[e]; e_2)}$$

$$\overline{\mathtt{ap}(\mathcal{E}'; e_2)[\mathcal{E}_2] = \mathtt{ap}(\mathcal{E}'[\mathcal{E}_2]; e_2)}$$

Contextual evaluation semantics

$$\frac{e \to e'}{\mathcal{E}[e] \to \mathcal{E}[e']}$$

$$\overline{\mathtt{ifz}\{e_0; x.e_1\}(\mathtt{z}) \to e_0}$$

$$\frac{v\ \mathsf{Val}}{\mathtt{ifz}\{e_0; x.e_1\}(\mathtt{s}(v)) \to [v/x]e_1}$$

$$\overline{\texttt{ap}(\texttt{lam}\{\tau\}(x.e); e') \rightarrow [e'/x]e}$$

$$\overline{\texttt{fix}\{\tau\}(x.e) \rightarrow [\texttt{fix}\{\tau\}(x.e)/x]e}$$

4.2 The Proofs

The statements we need to prove are:

- Completeness: If $e \mapsto^* e'$ and e' Val then $\epsilon \triangleright e \mapsto^* \epsilon \triangleleft e'$.
- Soundness: If $\epsilon \triangleright e \mapsto^* \epsilon \triangleleft e'$, then $e \mapsto^* e'$ and e' Val.

We now give an outline of the proof. Note that $\vartriangle$ is a placeholder for either $\triangleleft$ or $\triangleright$. Also remember that whenever the control machine state is $k \triangleleft e$, we assume e Val.

Lemma 1 (Control machine always returns values). If $k \vartriangle e \mapsto^* k' \triangleleft e'$, then e' Val.

Proof: by induction on control machine semantics.

Lemma 2 (Associativity of context application). If $\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3$ are contexts and e is an expression then $\mathcal{E}_1[\mathcal{E}_2[\mathcal{E}_3]] = \mathcal{E}_1[\mathcal{E}_2][\mathcal{E}_3]$, and $\mathcal{E}_1[\mathcal{E}_2[e]] = \mathcal{E}_1[\mathcal{E}_2][e]$.

Proof: by induction on the structure of $\mathcal{E}_1$.

Converting stacks to contexts For each stack k we define its corrsponding context $\mathsf{Ctx}(k)$ as follows:

$$\begin{aligned}
\mathsf{Ctx}(\epsilon) &= \circ \\
\mathsf{Ctx}(k; \texttt{s}(\star)) &= \mathsf{Ctx}(k)[\texttt{s}(\circ)] \\
\mathsf{Ctx}(k; \texttt{ifz}\{e_0; x.e_1\}(\star)) &= \mathsf{Ctx}(k)[\texttt{ifz}\{e_0; x.e_1\}(\circ)] \\
\mathsf{Ctx}(k; \texttt{ap}(\star; e_2)) &= \mathsf{Ctx}(k)[\texttt{ap}(\circ; e_2)]
\end{aligned}$$

Lemma 3. If $k \vartriangle e \mapsto k' \vartriangle' e'$, then either $\mathsf{Ctx}(k')[e'] = \mathsf{Ctx}(k)[e]$, or $\mathsf{Ctx}(k)[e] \mapsto \mathsf{Ctx}(k')[e']$.

Proof: by case analysis on the control machine steps.

Thus if $\epsilon \triangleright e \mapsto^* \epsilon \triangleleft e'$, then $e \mapsto^* e'$. Combining this with Lemma 1 above gives us soundness.

The remaining lemmas are for proving completeness. Some of them require non-trivial induction arguments over expressions and contexts. To make things convenient, we first define the *depth* of an expression:

$$\begin{aligned}
\mathsf{depth}(x) &= 0 \\
\mathsf{depth}(\texttt{z}) &= 0 \\
\mathsf{depth}(\texttt{s}(e)) &= \mathsf{depth}(e) + 1 \\
\mathsf{depth}(\texttt{ifz}\{e_0; x.e_1\}(e)) &= \mathsf{depth}(e) + 1 \\
\mathsf{depth}(\texttt{lam}\{\tau\}(x.e)) &= 0 \\
\mathsf{depth}(\texttt{ap}(e_1; e_2)) &= \mathsf{depth}(e_1) + 1 \\
\mathsf{depth}(\texttt{fix}\{\tau\}(x.e)) &= 0
\end{aligned}$$

The depth of a context can be defined similarly, by setting $\mathsf{depth}(\circ) = 0$.

Lemma 4. If $\mathcal{E}[e]$ Val, then e Val.

Proof: by induction on the structure of $\mathcal{E}$.

Lemma 5. If $\mathcal{E}_1[e_1] = \mathcal{E}_2[e_2]$, then there exists $\mathcal{E}'$ such that either $e_1 = \mathcal{E}'[e_2]$, or $e_2 = \mathcal{E}'[e_1]$.

Proof: if either one of $\mathcal{E}_1, \mathcal{E}_2$ is $\circ$ then the statement is trivial. Otherwise, $\mathcal{E}_1, \mathcal{E}_2$ must have the same constructor, i.e. there exists $\mathcal{E}_3, \mathcal{E}_4$ such that $\mathcal{E}_1 = \mathtt{s}(\mathcal{E}_3)$ and $\mathcal{E}_2 = \mathtt{s}(\mathcal{E}_4)$, or some other constructor. Now $\mathcal{E}_3, \mathcal{E}_4$ have smaller depth than $\mathcal{E}_1, \mathcal{E}_2$, so an induction on the depth of contexts proves the statement.

Lemma 6. If $e' = \mathcal{E}[e]$ then $\mathsf{depth}(e') = \mathsf{depth}(\mathcal{E}) + \mathsf{depth}(e)$. Similarly, if $\mathcal{E}' = \mathcal{E}_1[\mathcal{E}_2]$ then $\mathsf{depth}(\mathcal{E}') = \mathsf{depth}(\mathcal{E}_1) + \mathsf{depth}(\mathcal{E}_2)$.

Proof: by induction on the structure of $\mathcal{E}_1$.

From Lemma 6 we can easily prove that whenever $\mathcal{E}_1[e] = \mathcal{E}_2[e]$, we have $\mathcal{E}_1 = \mathcal{E}_2$. Note that $\mathcal{E}_1, \mathcal{E}_2$ must have the same depth, so an induction on the depth of $\mathcal{E}_1$ suffices.

Lemma 7 (Contextual evaluation step is deterministic). If $e \mapsto e'$ and $e \mapsto e''$, then $e' = e''$.

Proof: When proving this lemma it is convenient to work with the following modified contextual dynamics. We restrict the induction rule to

$$\frac{e \mapsto e'}{\mathcal{E}[e] \mapsto \mathcal{E}[e']} \quad (\mathcal{E} \neq \circ).$$

The base rules remain unchanged. It is easy to see that $e \mapsto e'$ under the modified dynamics iff $e \mapsto e'$ under the original dynamics.

Now suppose that $e \mapsto e'$ and $e \mapsto e''$. If both reductions follow base rules, it is easy to see that $e' = e''$. It is also easy to see that we cannot have the case where one reduction follows the induction rule and the other follows a base rule, by case analysis on expression structure.

The remaining case is where both reductions follow the induction rule. We prove this by induction on the depth of e. Here we have $e = \mathcal{E}_1[e_1] = \mathcal{E}_2[e_2]$, and $\mathcal{E}_1 \neq \circ, \mathcal{E}_2 \neq \circ$, hence $\mathsf{depth}(e_1) < \mathsf{depth}(e)$ and $\mathsf{depth}(e_2) < \mathsf{depth}(e)$. By Lemma 5 above we may assume (without loss of generality) that there exists $\mathcal{E}'$ with $e_1 = \mathcal{E}'[e_2]$. Hence by Lemma 2 and 6 above we have

$$\mathcal{E}_1[e_1] = \mathcal{E}_1[\mathcal{E}'[e_2]] = \mathcal{E}_1[\mathcal{E}'][e_2] = \mathcal{E}_2[e_2], \quad \mathcal{E}_1[\mathcal{E}'] = \mathcal{E}_2.$$

Suppose that $e_1 \mapsto e_1'$ and $e_2 \mapsto e_2'$. Since $e_1 = \mathcal{E}'[e_2]$, we see that $e_1 \mapsto \mathcal{E}'[e_2']$. By induction hypothesis we get $e_1' = \mathcal{E}'[e_2']$. Hence

$$\mathcal{E}_1[e_1'] = \mathcal{E}_1[\mathcal{E}'[e_2']] = \mathcal{E}_1[\mathcal{E}'][e_2'] = \mathcal{E}_2[e_2'].$$

Lemma 8. We say an expression e is *stuck* if e is not a value and there does not exist any e' such that $e \mapsto e'$. If e is stuck then $\mathcal{E}[e]$ is also stuck for any context $\mathcal{E}$.

Proof: by induction on the depth of $\mathcal{E}$.

Lemma 9 (Weak Liveness of control machine). For any given k, e, there exists k', e' such that $k \triangleright e \mapsto^* k' \vartriangle e'$, and one of the following is true:

1. There exists a step $k' \vartriangle e' \mapsto k'' \vartriangle e''$, such that $\mathsf{Ctx}(k')[e'] \mapsto \mathsf{Ctx}(k'')[e'']$, meaning the control machine will perform one step of contextual evaluation (instead of staying at the same expression, see Lemma 3);
2. $k = k'$ and $\vartriangle = \triangleleft$, meaning the control machine has finished evaluating the current stack-top expression;
3. $\mathsf{Ctx}(k')[e']$ is stuck.

Proof: by induction on the depth of e. If $\mathsf{depth}(e) = 0$ then $e = x$ or $e = \mathtt{z}$ or $e = \mathtt{lam}\{\tau\}(x.e)$ or $\mathtt{fix}\{\tau\}(x.e)$. The first case is stuck. The fourth case is a recursive expression. In the other two cases, the expression can be returned immediately. The inductive case is easy.

Lemma 10 (Liveness of control machine). For any control machine state $k \vartriangle e$, there exists k', e' such that $k \vartriangle e \mapsto^* k' \vartriangle e'$, and one of the following is true:

1. There exists a step $k' \vartriangle e' \mapsto k'' \vartriangle e''$, such that $\mathsf{Ctx}(k')[e'] \mapsto \mathsf{Ctx}(k'')[e'']$, meaning the control machine will perform one step of contextual evaluation, instead of staying at the same expression;
2. $k' = \epsilon$ and $\vartriangle = \vartriangleleft$, meaning the control machine has terminated;
3. $\mathsf{Ctx}(k')[e']$ is stuck.

Proof: by induction on the number of frames in k. Apply Lemma 9 above.

Lemma 11 (Completeness). For any control machine state $k \vartriangle e$, if $e_1 = \mathsf{Ctx}(k)[e]$, $e_1 \mapsto^* e_2$, and e_2 Val, then $k \vartriangle e \mapsto^* \epsilon \vartriangleleft e_2$.

Proof: We combine Lemma 7 and 10, and perform induction on the number of steps in the evaluation $e_1 \mapsto^* e_2$.

We start from the control machine state $k \vartriangle e$, and execute it until we encounter one of the three cases in Lemma 10. At this point, suppose that the control machine state is $k' \vartriangle e'$. Then by Lemma 3 we have $e_1 \mapsto^* \mathsf{Ctx}(k')[e']$. Since contextual evaluation is deterministic, we have $\mathsf{Ctx}(k')[e'] \mapsto e_2$. Thus we cannot encounter case 3 of Lemma 10.

If we encounter case 2 then we are done. Otherwise, we can execute the control machine by one more step, which corresponds to one step of contextual evaluation. By Lemma 7 this must be one of the steps in $e \mapsto^* e'$. Thus by induction hypothesis, the new control machine state $k'' \vartriangle e''$ should eventually reach a final state. Again by Lemma 7, this final state must be the same as $\epsilon \vartriangleleft e_2$.