CSE505 HW03 Sample Solution

1. Extending $ST\lambda C$

   (a) Small step operational semantics

$$\frac{e_1 \to e_1'}{\textsf{cons}\ (e_1, e_2) \to \textsf{cons}\ (e_1', e_2)} \qquad \frac{e_2 \to e_2'}{\textsf{cons}\ (v_1, e_2) \to \textsf{cons}\ (v_1, e_2')} \qquad \frac{e_1 \to e_1'}{\textsf{fold}\ (e_1, e_2, e_3) \to \textsf{fold}\ (e_1', e_2, e_3)}$$

$$\frac{e_2 \to e_2'}{\textsf{fold}\ (v_1, e_2, e_3) \to \textsf{fold}\ (v_1, e_2', e_3)} \qquad \frac{e_3 \to e_3'}{\textsf{fold}\ (v_1, v_2, e_3) \to \textsf{fold}\ (v_1, v_2, e_3')}$$

$$\frac{}{\textsf{fold}\ (v_1, v_2, \textsf{empty}) \to v_2} \qquad \frac{}{\textsf{fold}\ (v_1, v_2, \textsf{cons}\ (v_4, v_5)) \to \textsf{fold}\ (v_1, v_1\ v_2\ v_4, v_5)}$$

   (b) New stuck states: $\textsf{empty}\ x$, $\textsf{empty} v$, $\textsf{cons}\ (v, v')\ x$, $\textsf{cons}\ (v, v')\ v$, $\textsf{fold}\ (v_1, v_2, v_3)$ when v_3 is not $\textsf{empty}$ or $\textsf{cons}\ (v, v')$. New nested stuck states: $\textsf{cons}\ (e, e')$ if e is stuck, $\textsf{cons}\ (v, e')$ if e' is stuck, $\textsf{fold}\ (e_1, e_2, e_3)$ if e_1 stuck, $\textsf{fold}\ (v, e_2, e_3)$ if e_2 is stuck and $\textsf{fold}\ (v, v', e_3)$ if e_3 is stuck.

   (c) Typing rules

$$\frac{}{\Gamma \vdash \textsf{empty} : \tau\ \textsf{list}} \qquad \frac{\Gamma \vdash e_1 : \tau \qquad \Gamma \vdash e_2 : \tau\ \textsf{list}}{\Gamma \vdash \textsf{cons}\ (e_1, e_2) : \tau\ \textsf{list}}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \to (\tau_2 \to \tau_1) \qquad \Gamma \vdash e_2 : \tau_1 \qquad \Gamma \vdash e_3 : \tau_2\ \textsf{list}}{\Gamma \vdash \textsf{fold}\ (e_1, e_2, e_3) : \tau_1}$$

   (d) Preservation, Progress, and Substitution extensions.

Preservation Lemma If $\cdot \vdash e : \tau$ and $e \to e'$, then $\cdot \vdash e' : \tau$.

Proof. By induction on the height of the derivation of $\cdot \vdash e_1 : \tau$. We must add new cases for the bottom rule of the derivation.

- $\cdot \vdash \textsf{empty} : \tau\ \textsf{list}$. Then $e \to e'$ is impossible, so lemma holds vacuously.
- $\cdot \vdash \textsf{cons}\ (e_1, e_2) : \tau\ \textsf{list}$. Then we know $\cdot \vdash e_1 : \tau$ and $\cdot \vdash e_2 : \tau\ \textsf{list}$. There are 2 ways to derive $\textsf{cons}\ (e_1, e_2) \to e'$:
  - e' is $\textsf{cons}\ (e_1', e_2)$ and $e_1 \to e_1'$. By induction $\cdot \vdash e_1' : \tau$, and we can derive that $\cdot \vdash \textsf{cons}\ (e_1', e_2) : \tau\ \textsf{list}$.
  - e' is $\textsf{cons}\ (e_1, e_2')$ and $e_2 \to e_2'$. By induction $\cdot \vdash e_2' : \tau\ \textsf{list}$, and we can derive that $\cdot \vdash \textsf{cons}\ (e_1, e_2') : \tau\ \textsf{list}$.
- $\cdot \vdash \textsf{fold}\ (e_1, e_2, e_3) : \tau_1$. Then we know $\cdot \vdash e_1 : \tau_1 \to (\tau_2 \to \tau_1)$, $\cdot \vdash e_2 : \tau_1$, and $\cdot \vdash e_3 : \tau_2\ \textsf{list}$. There are 5 ways to derive $\textsf{fold}\ (e_1, e_2, e_3) \to e'$:
  - e' is $\textsf{fold}\ (e_1', e_2, e_3)$ and $e_1 \to e_1'$. By induction $\cdot \vdash e_1' : \tau_1 \to (\tau_2 \to \tau_1)$, and we can derive that $\cdot \vdash \textsf{fold}\ (e_1', e_2, e_3) : \tau_1$.
  - e' is $\textsf{fold}\ (e_1, e_2', e_3)$ and $e_2 \to e_2'$. By induction $\cdot \vdash e_2' : \tau_1$, and we can derive that $\cdot \vdash \textsf{fold}\ (e_1, e_2', e_3) : \tau_1$.
  - e' is $\textsf{fold}\ (e_1, e_2, e_3')$ and $e_3 \to e_3'$. By induction $\cdot \vdash e_3' : \tau_2\ \textsf{list}$, and we can derive that $\cdot \vdash \textsf{fold}\ (e_1, e_2, e_3') : \tau_1$.
  - e_3 is $\textsf{empty}$, e_2 is some v, and e' is also v. Then since $\cdot \vdash e_2 : \tau_1$, we have that $\cdot \vdash e' : \tau_1$.
  - e_1 is some v_1, e_2 is some v_2, e_3 is some $\textsf{cons}\ (v_4, v_5)$, and e' is $v_1\ v_2\ v_4$. Since $\cdot \vdash e_3 \equiv \textsf{cons}\ (v_4, v_5) : \tau_2\ \textsf{list}$, we must have $\cdot \vdash v_4 : \tau_2$. Since we have $\cdot \vdash v_1 : \tau_1 \to (\tau_2 \to \tau_1)$, $\cdot \vdash v_2 : \tau_1$, and $\cdot \vdash v_4 : \tau_2$, the application typing rule gives us $\cdot \vdash v_1\ v_2\ v_4 : \tau_1$ and from this we can derive $\cdot \vdash \textsf{fold}\ (v_1, v_1\ v_2\ v_4, v_5) : \tau$.

$\square$

Canonical Forms Lemma Extend the lemma with: If $\cdot \vdash v : \tau\ \mathsf{list}$ then v has the form empty or $\mathsf{cons}\ (v_1, v_2)$

Proof. By inspection of the typing rules. □

Progress Lemma If $\cdot \vdash e : \tau$, then e is a value or there exists an e' such that $e \to e'$.

Proof. By structural induction (syntax height) on e. The structure of e may now also have one of the following forms:

- empty. Then e is a value.
- $\mathsf{cons}\ (e_1, e_2)$. By induction either e_1 is some v_1 or can become some e_1'. If it becomes e_1', then $\mathsf{cons}\ (e_1, e_2) \to \mathsf{cons}\ (e_1', e_2)$. Else by induction either e_2 is some v_2 or can become some e_2'. If it becomes e_2', then $\mathsf{cons}\ (v_1, e_2) \to \mathsf{cons}\ (v_1, e_2')$. Else e is $\mathsf{cons}\ (v_1, v_2)$, a value.
- $\mathsf{fold}\ (e_1, e_2, e_3)$. By induction either e_1 is some v_1 or can become some e_1'. If it becomes e_1', then $\mathsf{fold}\ (e_1, e_2, e_3) \to \mathsf{fold}\ (e_1', e_2, e_3)$. Else $e_1 \equiv v_1$ and by induction either e_2 is some v_2 or can become some e_2'. If it becomes e_2', then $\mathsf{fold}\ (v_1, e_2, e_3) \to \mathsf{fold}\ (v_1, e_2', e_3)$. Else $e_2 \equiv v_2$ and by induction either e_3 is some v_3 or can become some e_3'. If it becomes e_3', then $\mathsf{fold}\ (v_1, v_2, e_3) \to \mathsf{fold}\ (v_1, v_2, e_3')$. Otherwise e is $\mathsf{fold}\ (v_1, v_2, v_3)$; since it typechecks, inverting the assumed typing derivation gives $\cdot \vdash v_3 : \tau_2\ \mathsf{list}$. By Canonical Forms, either v_3 is empty, in which case $\mathsf{fold}\ (v_1, v_2, v_3) \to v_2$, or v_3 is some $\mathsf{cons}\ (v_4, v_5)$, in which case $\mathsf{fold}\ (v_1, v_2, v_3) \to \mathsf{fold}\ (v_1, v_1\ v_2\ v_4, v_5)$.

□

Extend Substitution

$$\frac{}{\mathsf{empty}\,[e/x] = \mathsf{empty}} \qquad \frac{e_1\,[e/x] = e_1' \qquad e_2\,[e/x] = e_2'}{\mathsf{cons}\ (e_1, e_2)\,[e/x] = \mathsf{cons}\ (e_1', e_2')}$$

$$\frac{e_1\,[e/x] = e_1' \qquad e_2\,[e/x] = e_2' \qquad e_3\,[e/x] = e_3'}{\mathsf{fold}\ (e_1, e_2, e_3)\,[e/x] = \mathsf{fold}\ (e_1', e_2', e_3')}$$

Substitution Lemma If $\Gamma, x : \tau' \vdash e_1 : \tau$ and $\Gamma \vdash e_2 : \tau'$, then $\Gamma \vdash e_1\,[e_2/x] : \tau$.

Proof. By induction on derivation of $\Gamma, x : \tau' \vdash e_1 : \tau$. The bottom rule could conclude:

- $\Gamma, x : \tau' \vdash \mathsf{empty} : \tau\ \mathsf{list}$. Then $\mathsf{empty}\,[e_2/x] = \mathsf{empty}$ and $\Gamma \vdash \mathsf{empty} : \tau\ \mathsf{list}$.
- $\Gamma, x : \tau' \vdash \mathsf{cons}\ (e_3, e_4) : \tau\ \mathsf{list}$. Then $\Gamma, x : \tau' \vdash e_3 : \tau$ and $\Gamma, x : \tau' \vdash e_4 : \tau\ \mathsf{list}$. By induction, we have that $\Gamma \vdash e_3\,[e_2/x] = e_3' : \tau$ and $\Gamma \vdash e_4\,[e_2/x] = e_4' : \tau\ \mathsf{list}$, giving us that $\Gamma \vdash \mathsf{cons}\ (e_3, e_4)\,[e/x] = \mathsf{cons}\ (e_3', e_4') : \tau\ \mathsf{list}$.
- $\Gamma, x : \tau' \vdash \mathsf{fold}\ (e_3, e_4, e_5) : \tau_1$. Then $\Gamma, x : \tau' \vdash e_3 : \tau_1 \to (\tau_2 \to \tau_1)$, $\Gamma, x : \tau' \vdash e_4 : \tau_1$, and $\Gamma, x : \tau' \vdash e_5 : \tau_2\ \mathsf{list}$. By induction, we have that $\Gamma \vdash e_3\,[e_2/x] = e_3' : \tau_1 \to (\tau_2 \to \tau_1)$, $\Gamma \vdash e_4\,[e_2/x] = e_4' : \tau_1$, and $\Gamma \vdash e_5\,[e_2/x] = e_5' : \tau_2\ \mathsf{list}$, giving us that
$\Gamma \vdash \mathsf{fold}\ (e_3, e_4, e_5)\,[e/x] = \mathsf{fold}\ (e_3', e_4', e_5') : \tau_1$.

□

(e) Implementation: See file `hw3.ml`.

2. For all cases see file `main.ml`.