6.883: FOUNDATIONS OF PROGRAM ANALYSIS
PROBLEM SET 1

JEAN YANG

Problem 1

Please refer to Problem 2 for definition of equals, and.

$$
\begin{aligned}
\text{fix} &:= \lambda f.(\lambda x.f\ (x\ x))\ (\lambda x.f\ (x\ x)) \\
\text{ifthenelse} &:= \lambda e.\lambda t.\lambda f.e\ t\ f \\
0 &:= \lambda f.\lambda s.s \\
1 &:= \lambda f.\lambda s.f\ s \\
2 &:= \lambda f.\lambda s.f\ (f\ s) \\
\text{add} &:= \lambda m.\lambda n.\lambda s.n\ f\ (m\ f\ s) \\
\text{pred} &:= \lambda n.\lambda f.\lambda s.n\ (\lambda g.\lambda h.h\ (g\ f))\ (\lambda x.s)\ (\lambda x.x) \\
\text{sub} &:= \lambda m.\lambda n.\lambda f.\lambda s.m\ \text{pred}\ (n\ f\ s) \\
\text{fibf} &:= \lambda f.\lambda n.\text{ifthenelse}\ (\text{equals}\ n\ 0)\ 1\ (\text{ifthenelse}\ (\text{equals}\ n\ 1)\ 1\ (\text{add}\ (f\ (\text{sub}\ n\ 1))\ (f\ (\text{sub}\ n\ 2))) \\
\text{fib} &:= \text{fix fibf}
\end{aligned}
$$

Problem 2

2a.

$$
\begin{aligned}
\text{or} &:= \lambda e1.\lambda e2.\lambda f.\lambda g.e1\ f\ (e2\ f\ g) \\
\text{and} &:= \lambda e1.\lambda e2.\lambda f.\lambda g.e1\ (e1\ f\ g)\ g
\end{aligned}
$$

2b.

$$
\begin{aligned}
\text{true} &:= \lambda f.\lambda g.f \\
\text{false} &:= \lambda f.\lambda g.g \\
\text{isZero} &:= \lambda n.n\ (\lambda x.\text{false})\ \text{true}
\end{aligned}
$$

2c.

$$\text{equals} := \lambda m.\lambda n.\text{and}\ (\text{isZero}\ (\text{sub}\ m\ n))\ (\text{isZero}\ (\text{sub}\ n\ m))$$

Problem 3

3a.

$$\frac{\langle c, \sigma\rangle \to \sigma' \quad \langle e, \sigma'\rangle \to \text{false}}{\langle \text{trans } c \text{ check } e, \sigma\rangle \to \sigma}$$

$$\frac{\langle c, \sigma\rangle \to \sigma' \quad \langle e, \sigma'\rangle \to \text{true}}{\langle \text{trans } c \text{ check } e, \sigma\rangle \to \sigma'}$$

3b.

$$
\dfrac{
\dfrac{\sigma_0 = \sigma[x\backslash 0]}{\langle x := 0, \sigma\rangle \to \sigma_0}
\quad
\dfrac{
\dfrac{
\dfrac{
\dfrac{\dfrac{\sigma_0(x) = 1}{\langle x+1, \sigma_0\rangle \to 0+1} \quad \sigma_1 = \sigma_0[x\backslash x']}{\langle x := x+1, \sigma_0\rangle \to \sigma_1}
\quad
\dfrac{\sigma(x_1) \neq 2}{\langle x = 2, \sigma_1\rangle \to \text{false}}
}{\langle \text{trans } x := x+1 \text{ check } x = 2)\rangle \to \sigma_0}
\quad
\dfrac{\dfrac{\vdots}{\langle x+1, \sigma_0\rangle \to 1} \quad \sigma_2 = \sigma_1[x\backslash 1]}{\langle x := x+1, \sigma_0\rangle \to \sigma_2}
}{\langle \text{trans } x := x+1 \text{ check } x = 2);\ x := x+1\rangle \to \sigma_2}
\quad
\dfrac{\sigma_2(x) = 0}{\langle x = 1, \sigma_2\rangle \to \text{true}}
}{\text{trans (trans } x := x+1 \text{ check } x = 2);\ x := x+1 \text{ check } x = 1, \sigma\rangle \to \sigma_2}
}{\langle x := 0;\ \text{trans (trans } x := x+1 \text{ check } x = 2);\ x := x+1 \text{ check } x = 1, \sigma\rangle \to \sigma_2}
$$

Date: Wednesday, September 30, 2009.
Collaborator: Joseph ♡ Near.

Problem 4

4a. We introduce the AST term trans' in order to distinguish a fully unevaluated trans command from a partially evaluated one.

$$\langle \text{trans } c \text{ check } e, \sigma, s\rangle \rightarrow \langle \text{trans}' \; c \text{ check } e, \sigma, \sigma.s\rangle$$
$$\langle \text{trans}' \text{ skip check true}, \sigma, s\rangle \rightarrow \langle v, \sigma', s\rangle$$
$$\langle \text{trans}' \text{ skip check false}, \sigma, s\rangle \rightarrow \langle v, \sigma, s\rangle$$

4b. Here we use v as the convention for the result of evaluating a value.

$$\begin{aligned} H ::= \;& 0 \mid n + H \mid H + e \mid x := H \\ & \mid \text{if } H \text{ then } C1 \text{ else } C2 \mid H;\; C \\ & \mid \text{trans } H \text{ check } e \\ & \mid \text{trans } v \text{ check } H \end{aligned}$$

Problem 5

Claim. *With the constant evaluation rule*

$$\frac{}{\langle N, \sigma\rangle \rightarrow 2 * n},$$

if the initial state of a program contains only even values the final state will contain only even values.

Proof. By induction on the derivation tree.

First we can show that evaluation of expressions always results in an even integer.

- **Evaluation of numbers.** We have the rule

$$\langle N, \sigma\rangle \rightarrow 2 * n,$$

  so the result is always even.
- **Evaluation of locations.** The rule

$$\langle X, \sigma\rangle \rightarrow \sigma(X)$$

  yields a value from the store. Since the state contains only even integer values, the result must be even.
- **Evaluation of sums, subtractions, and products.** We can show this by structural induction on terms. We have the rules

$$\frac{\langle a_0, \sigma\rangle \rightarrow n_0 \quad \langle a_1, \sigma\rangle \rightarrow n_1 \quad n = n_0 + n_1}{\langle a_0 + a_1, \sigma\rangle \rightarrow n}$$
$$\frac{\langle a_0, \sigma\rangle \rightarrow n_0 \quad \langle a_1, \sigma\rangle \rightarrow n_1 \quad n = n_0 - n_1}{\langle a_0 - a_1, \sigma\rangle \rightarrow n}$$
$$\frac{\langle a_0, \sigma\rangle \rightarrow n_0 \quad \langle a_1, \sigma\rangle \rightarrow n_1 \quad n = n_0 * n_1}{\langle a_0 * a_1, \sigma\rangle \rightarrow n}$$

  When a_0 and a_1 are numbers or locations, the results n_0 and n_1 must be even for all three arithmetic rules. When a_0 or a_1 is an arithmetic expression, we can break the expression down to a binary operation between integers, and so we can show that it must evaluate to an integer.

Now we use well-founded induction on the $\prec$ relationship between proper subderivations that the derivation of commands only adds even integers to σ.

The skip command does not change the state, so it preserves the desired property:

$$\langle \text{skip}, \sigma\rangle \rightarrow \sigma.$$

Consider the assignment command:

$$\frac{\langle a, \sigma\rangle \rightarrow m}{\langle X := a, \sigma\rangle \rightarrow \sigma[m \backslash X]}.$$

Since we have shown m must be an even integer, assignment can only add even integers to σ.

For command sequencing, let

$$d = \cfrac{\langle c_0, \sigma\rangle \to \sigma'' \quad \langle c_1, \sigma''\rangle \to \sigma'}{\langle c_0; c_1, \sigma\rangle \to \sigma'},$$

$$d_0 = \cfrac{\vdots}{\langle c_0, \sigma\rangle \to \sigma''},$$

$$d_1 = \cfrac{\vdots}{\langle c_1, \sigma''\rangle \to \sigma'}.$$

We have $d_0 \prec d$ and $d_1 \prec d$. Given that derivations d_0 and d_1 yield environments containing only even integers from environments containing only even integers, we know σ' and σ'' contain only even integers. Then the σ'' resulting from d contains only even integers.

Now consider conditionals:

$$d = \cfrac{\langle b, \sigma\rangle \to \text{true} \quad \langle c_0, \sigma\rangle \to \sigma'}{\langle \text{if } b \text{ then } c_0 \text{ else } c_1, \sigma\rangle \to \sigma'},$$

$$d_0 = \cfrac{\vdots}{\langle c_0, \sigma\rangle \to \sigma'}.$$

Evaluation of $\langle b, \sigma\rangle$ does not affect σ. We have $d_0 \prec d$, so the resulting environment σ' contains only even integers. The argument is analogous for the false case.

The base case for while loops does not affect σ:

$$\cfrac{\langle b, \sigma\rangle \to \text{false}}{\langle \text{while } b \text{ do } c, \sigma\rangle \to \sigma}.$$

Now consider the loop case:

$$d = \cfrac{\langle b, \sigma\rangle \to \text{true} \quad \langle c, \sigma\rangle \to \sigma'' \quad \langle \text{while } b \text{ do } c, \sigma''\rangle \to \sigma'}{\langle \text{while } b \text{ do } c, \sigma\rangle \to \sigma'},$$

$$d_0 = \cfrac{\vdots}{\langle c, \sigma\rangle \to \sigma''},$$

$$d_1 = \cfrac{\vdots}{\langle \text{while } b \text{ do } c, \sigma''\rangle \to \sigma'}.$$

We have $d_0 \prec d$ and $d_1 \prec d$, so σ'' and σ' contain only even integers and we know d results in an environment σ' containing only even integers. □

Problem 6

Claim. *For an arbitrary program P, let $\langle P, \sigma_1\rangle \to \sigma_1'$ and $\langle P, \sigma_2\rangle \to \sigma_2'$. Then*

$$\forall x.((\sigma_1(x) = n^p \iff \sigma_2(x) = n^p) \Rightarrow (\sigma_1'(x) = m^p \Rightarrow \sigma_2'(x) = m^p)).$$

Proof. Because structure of the derivation rules are the same whether variables are public or private, the derivation trees for $\langle P, \sigma_1\rangle$ and $\langle P, \sigma_2\rangle$ must have the same structure. Thus we can prove our claim using induction on the derivation tree for $\langle P, \sigma_1\rangle$.

We can first show using structural induction that derivation of expressions preserves the property that $\sigma_1(x) = n^p \iff \sigma_2(x) = n^p$ for all x implies $\langle e, \sigma_1\rangle \to n^p \iff \langle e, \sigma_2\rangle \to n^p$.

- This is trivially true for the rules with no premises:

$$\cfrac{}{\langle N, \sigma\rangle \to n^p}$$

$$\cfrac{}{\langle x, \sigma\rangle \to \sigma(x)}$$

- For binary relations on expressions e_1 and e_2, if both e_1 and e_2 evaluate to public values, then all of their components must be public and therefore agreed upon by σ_1 and σ_2, so $\langle e, \sigma_1\rangle \to n^p \Rightarrow \langle e, \sigma_2\rangle \to n^p$.

$$\cfrac{\langle e_1, \sigma\rangle \to n_1^p \quad \langle e_2, \sigma\rangle \to n_2^p \quad n_1 + n_2 = n}{\langle e_1 + e_2, \sigma\rangle \to n^p}$$

$$\cfrac{\langle e_1, \sigma\rangle \to n_1^p \quad \langle e_2, \sigma\rangle \to n_2^p \quad n_1 = n_2 = b}{\langle e_1 = e_2, \sigma\rangle \to b^p}$$

- For binary relations on expressions e_1 and e_2, if e_1 or e_2 evaluate to private values, then there must be some component of e_1 or $e_2$that is secret. Then the result of the evaluation in both σ_1 and σ_2 will result in a secret value, so we don't care whether the results agree.

$$\frac{\langle e_1, \sigma\rangle \to n_1^q \quad \langle e_2, \sigma\rangle \to n_2^q \quad n_1 + n_2 = n \quad (n_1 \text{ or } n_2 \text{ secret})}{\langle e_1 + e_2, \sigma\rangle \to n^s}$$

$$\frac{\langle e_1, \sigma\rangle \to n_1^q \quad \langle e_2, \sigma\rangle \to n_2^q \quad n_1 = n_2 = b \quad (n_1 \text{ or } n_2 \text{ secret})}{\langle e_1 = e_2, \sigma\rangle \to b^s}$$

We prove our property on commands using well-founded induction on the proper subderivation $\prec$ between the executions of commands. Consider when we have an assignment statement:

$$\frac{\langle e, \sigma_1\rangle \to n^p}{\langle X := e, \sigma_1\rangle \to \sigma_1[X \to n^p]},$$
$$\frac{\langle e, \sigma_1\rangle \to n^s}{\langle X := e, \sigma_1\rangle \to \sigma_1[X \to n^s]}.$$

When we have $\sigma_1(x) = n^p \iff \sigma_2(x) = n^p$, we must have $\langle e, \sigma\rangle \to n^p \iff \langle e, \sigma\rangle \to n^p$, so $\forall x. \sigma_1'(x) \Rightarrow \sigma_2'(x)$.

Now consider conditionals. When the conditional expression evaluates to a secret value, we do not modify σ, so our desired property holds. Consider the case when the condition is public:

$$d = \frac{\langle e_1, \sigma_1\rangle \to \text{true}^p \quad \langle c_t, \sigma_1\rangle \to \sigma_1'}{\langle \text{if } e_1 \text{ then } c_t \text{ else } c_f, \sigma_1\rangle \to \sigma_1'},$$
$$d_0 = \langle c_t, \sigma_1\rangle \to \sigma_1'.$$

We have $d_0 \prec d$, so if d_0 preserves privacy, then d preserves privacy, since the resulting environment σ_1' of d is the same as the resulting environment σ_1' of d_0. The argument is analogous for the false case.

The base case of the while loop and the case when the loop condition is secret do not change the environment. We have left the loop case:

$$d = \frac{\langle e_1, \sigma_1\rangle \to \text{true}^p \quad \langle c, \sigma_1\rangle \to \sigma_1'' \quad \langle \text{while } e_1 \text{ then } c, \sigma_1''\rangle \to \sigma_1'}{\langle \text{while } e_1 \text{ then } c, \sigma_1\rangle \to \sigma_1'},$$

$$d_0 = \frac{\vdots}{\langle c, \sigma_1\rangle \to \sigma_1''},$$

$$d_1 = \frac{\vdots}{\langle \text{while } e_1 \text{ then } c, \sigma_1''\rangle \to \sigma_1'}.$$

Since $d_0 \prec d$, we know $\forall x. \sigma_1''(x) \to m^p \Rightarrow \sigma_2''(x) \to m^p$. Since $d_1 \prec d$, we get $\forall x. \sigma_1'(x) \to m^p \Rightarrow \sigma_2'(x) \to m^p$.

$$d = \frac{\langle c, \sigma_1\rangle \to \sigma_1''}{\langle c_1; c_2, \sigma_1\rangle \to \sigma_1'}$$

$$d_0 = \frac{\vdots}{\langle c_1, \sigma_1\rangle \to \sigma_1''},$$

$$d_1 = \frac{\vdots}{\langle c_2, \sigma_1''\rangle \to \sigma_1'}.$$

Since $d_0 \prec d$, we know $\forall x. \sigma_1''(x) \to m^p \Rightarrow \sigma_2''(x) \to m^p$. Since $d_1 \prec d$, we get $\forall x. \sigma_1'(x) \to m^p \Rightarrow \sigma_2'(x) \to m^p$. □