The last page of this document contains a reference for STLC with booleans.

Problems from after Lecture on Wednesday 4/15

Problem 1. Consider the following statement.

If $\cdot \vdash e : \tau$ then e is closed.

(a) Enumerate all the things you could try to induct on. Say which ones are reasonable choices.
$\cdot \vdash e : \tau$, τ or e. $\cdot \vdash e : \tau$ and e are reasonable choices

(b) For the most reasonable choice (your choice!) of thing to induct on, say why the direct proof by induction will not work. Be specific.

I choose to induct on $\cdot \vdash e : \tau$. When proceeding at the case $\dfrac{[x \mapsto \tau_1] \vdash e : \tau_2}{\cdot \vdash \lambda x.e : \tau_1 \to \tau_2}$, I cannot apply the induction hypothesis on e since we are assuming an empty context Γ

(c) State a stronger lemma and prove it by induction on a thing of your choice. Be sure to state your strengthened lemma clearly. Also, explain briefly and informally why your strengthening is, in fact, stronger than the statement above.

For all e, x and context Γ, if $x \in \mathsf{FV}(e)$ and $\Gamma \vdash e : \tau$ for some type τ, then there exists a type τ' such that $\Gamma \vdash x : \tau'$

Proof. By induction on $\mathsf{FV}(e)$.

i. Case $\mathsf{FV}(b) = \emptyset$. This case e does not have any free variable, thus this case is vacuous.

ii. Case $\mathsf{FV}(x) = \{x\}$. In this case, e is x. Since $\Gamma \vdash e : \tau$, exists $\tau' = \tau$ such that $\Gamma \vdash x : \tau'$.

iii. Case $\mathsf{FV}(e_1\ e_2) = \mathsf{FV}(e_1) \cup \mathsf{FV}(e_2)$.

A. $x \in \mathsf{FV}(e_1)$. Since $\Gamma \vdash e_1\ e_2 : \tau$, according to the typing rule, the only way to get this is

$$\frac{\Gamma \vdash e_1 : \tau_1 \to \tau \qquad \Gamma \vdash e_2 : \tau_1}{\Gamma \vdash e_1\ e_2 : \tau}$$

Therefore, $\Gamma \vdash e_1 : \tau_1 \to \tau$. Since, in this case, $x \in \mathsf{FV}(e_1)$, according to the induction hypothesis, there exists a type τ' such that $\Gamma \vdash x : \tau'$.

B. $x \in \mathsf{FV}(e_2)$. Similar as Case A.

Case $\mathsf{FV}(\texttt{if } e_1 \texttt{ then } e_2 \texttt{else } e_3) = \mathsf{FV}(e_1) \cup \mathsf{FV}(e_2) \cup \mathsf{FV}(e_3)$. Similar as Case iii.

Case $\mathsf{FV}(\lambda y.e) = \mathsf{FV}(e) - \{y\}$. In this case, x is free in e. Since $\Gamma[y \mapsto \tau_1] \vdash e : \tau$, and note that $y \neq x$ since x is also free in $\lambda y.e$, according to the induction hypothesis, there exists a τ' such that $\Gamma \vdash x : \tau'$. □

This lemma is stronger. Having $\cdot \vdash e : \tau$ and assuming that e is not closed, then there exists a free variable $x \in \mathsf{FV}(e)$ such that $\cdot \vdash x : \tau'$. However, $\cdot$ is an empty envrionment, so the domain of $\cdot$ is $\emptyset$, and no free variable can be assigned a type in this context. This contradicts with the conclusion of the lemma that $\exists \tau', \cdot \vdash x : \tau'$. So if $\cdot \vdash e : \tau$, e must be closed.

Problem 2. This problem is about the substitution operator $e_1[e/x]$.

(a) In the definition of substitution, for the λ case, there are two side conditions, $y \neq x$ (which we forgot to write in lecture) and $y \notin \mathsf{FV}(e)$. For the first side condition, $y \neq x$, explain what can go wrong if we leave it out by giving a concrete example where substitution behaves unexpectedly.

If we leave the restriction $y \neq x$ out, the substitution can change the semantics of the lambda abstraction by substituting a bound variable. For instance: if we have $(\lambda x.x\ \ x)[v/x]$, this would be rewritten to $\lambda x.v\ \ v$, which changed the semantics of the lambda abstraction.

(b) Explain what *should* happen if $y = x$. Why is it ok to *not* handle this case explicitly in the definition of substitution?

The body of the lambda abstraction should remain the same after substitution. It is ok to not handle this case since if $y = x$, the behavior is defined in application

(c) Now consider the second side condition from the λ case, namely $y \notin \mathsf{FV}(e)$. Describe a simple condition on e that (1) ensures this side condition is always met; and (2) is sufficient to cover the cases we encountered in proving type safety. In your answer, state your condition clearly, and explain briefly and informally why it satisfies (1) and (2).

(d) Suppose we remove this second side condition. Explain informally why any expression that is well typed in the empty context still evaluates the same way without this side condition.

According to the theorem proved in Question 1, if an expression e is well-typed in an empty context, then e is closed, hence $\mathsf{FV}(e) = \emptyset$, and $\forall y.\ y \notin \emptyset$, so ignoring the second condition does not affect the way of evaluating the substitution.

(e) Find a well-typed expression (in a non-empty context!) that steps differently with and without this second side condition. In your answer, state your expression and its typing context clearly, and show informally the two different executions it has with and without this side condition.

Consider the expression $(\lambda x.\texttt{if}\ x\ \texttt{then}\ y\ \texttt{else}\ x)\ \mathit{true}$ with the context $[y \mapsto \mathsf{bool}]$, and we are to substitute y with x, i.e. $((\lambda x.\ \texttt{if}\ x\ \texttt{then}\ y\ \texttt{else}\ x)[x/y])\ \mathit{true}$. Obviously, x is free in current context, and $x \neq y$ so we can proceed the substitution (ignoring the second condition). After substitution, the expression becomes $(\lambda x.\ \texttt{if}\ x\ \texttt{then}\ x\ \texttt{else}\ x)\ \mathit{true}$. These two expressions can execute differently: consider having $y = \mathit{false}$, the original application yields false but after substitution, the expression evaluates to true instead.

Problem 3. This problem considers adding pairs to the language. Your job is to add syntax and rules, and to update the proofs.

(a) Add new syntax.

- For expressions, add (e, e), to construct a pair, and $e.1$ and $e.2$, to project out the components.
  extends e with $e ::= (e, e) \mid e.1 \mid e.2$
- For values, add a new branch to the grammar so that a pair of values is considered a value.
  Extends v with $v ::= (v, v)$
- For types, make it so the product of two types, written $\tau_1 \times \tau_2$ is a type.
  Extends τ with $\tau ::= \tau \times \tau$

(b) Add semantics. (4 boring rules and 2 rules "where stuff happens".)

- Add rules to $e \longrightarrow e$ such that pairs (e_1, e_2) get evaluated in left to right order.

$$\frac{e_1 \longrightarrow e_1'}{(e_1, e_2) \longrightarrow (e_1', e_2)} \qquad \frac{e_2 \longrightarrow e_2'}{(v, e_2) \longrightarrow (v, e_2')}$$

- For $e.1$ and $e.2$, make sure that e gets evaluated to a value before the projection occurs.

$$\frac{e_1 \longrightarrow e_1'}{(e_1, e_2).1 \longrightarrow (e_1', e_2).1} \qquad \frac{e_1 \longrightarrow e_1'}{(e_1, e_2).2 \longrightarrow (e_1', e_2).2}$$

$$\frac{e_2 \longrightarrow e_2'}{(v, e_2).1 \longrightarrow (v, e_2').1} \qquad \frac{e_2 \longrightarrow e_2'}{(v, e_2).2 \longrightarrow (v, e_2').2}$$

$$\frac{}{(v_1, v_2).1 \longrightarrow v_1} \qquad \frac{}{(v_1, v_2).2 \longrightarrow v_2}$$

(c) Add typing rules. Add one rule per new expression AST node.

$$\frac{\Gamma \vdash e_1 : \tau_1 \qquad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash (e_1, e_2) : \tau_1 \times \tau_2} \qquad\qquad \frac{\Gamma \vdash e : \tau_1 \times \tau_2}{\Gamma \vdash e.1 : \tau_1} \qquad\qquad \frac{\Gamma \vdash e : \tau_1 \times \tau_2}{\Gamma \vdash e.2 : \tau_2}$$

(d) Extend the proof of type safety, as follows:

- Add cases to the proof of the progress lemma from lecture for each new typing rule you added. No need to repeat the cases we covered in lecture, just handle your new rules. If you need any lemmas, clearly state them, and describe in one sentence how you *would* prove them (by induction or some other way? induction on what?), but no need to prove your lemmas.

  i. Case $\dfrac{\Gamma \vdash e_1 : \tau_1 \qquad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash (e_1, e_2) : \tau_1 \times \tau_2}$. According to the induction hypothesis, e_1 is not a stuck, therefore:

  - e_1 is a value. Then according to the induction hypothesis, e_2 is not a stuck. If e_2 is a value., then (e_1, e_2) is value, and it is not a stuck; if $e_2 \longrightarrow e_2'$, then the expression can take this step: $(e_1, e_2) \longrightarrow (e_1, {e_2}')$. Thus it is not a stuck in both cases.
  - e_1 can step to e_1'. Then directly, (e_1, e_2) can step to (e_1', e_2), thus it is not a stuck.

  ii. Case $\dfrac{\Gamma \vdash e : \tau_1 \times \tau_2}{\Gamma \vdash e.1 : \tau_1}$. Since $\Gamma \vdash e : \tau_1 \times \tau_2$, $e = (e_1, e_2)$ for some e_1 and e_2, and $\Gamma \vdash e_1 : \tau_1$ and $\Gamma \vdash e_2 : \tau_2$. According to the induction hypothesis, e is not a stuck, therefore:

  - e is a value. Then $e = (v_1, v_2)$, and $e.1 = v_1$, which is a value, and thus $e.1$ is not a stuck.
  - e can step. Then by case analysis on $e \longrightarrow e'$

    A. $\dfrac{e_1 \longrightarrow e_1'}{(e_1, e_2) \longrightarrow (e_1', e_2)}$. According to the evaluation rule, $\dfrac{e_1 \longrightarrow e_1'}{(e_1, e_2).1 \longrightarrow (e_1', e_2).1}$, therefore, e is not a stuck.

    B. $\dfrac{e_2 \longrightarrow e_2'}{(v, e_2) \longrightarrow (v, e_2')}$. Similar as Case A.

  iii. Case $\dfrac{\Gamma \vdash e : \tau_1 \times \tau_2}{\Gamma \vdash e.2 : \tau_2}$. Similar as Case ii.

- Add cases to the proof of the preservation lemma from lecture. Same directions as above about repeated cases and lemmas.

  i. Case $\dfrac{\Gamma \vdash e_1 : \tau_1 \qquad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash (e_1, e_2) : \tau_1 \times \tau_2}$. According to the induction hypothesis, if $e_1 \longrightarrow e_1'$, then $\Gamma \vdash e_1' : \tau_1$, similar for e_2'. Therefore, by case analysis on $e \longrightarrow e'$:

  - $\dfrac{e_1 \longrightarrow e_1'}{(e_1, e_2) \longrightarrow (e_1', e_2)}$. In this case, according to the induction hypothesis,

    $$\frac{\Gamma \vdash e_1' : \tau_1 \qquad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash (e_1', e_2) : \tau_1 \times \tau_2}$$

  - $\dfrac{e_2 \longrightarrow e_2'}{(v, e_2) \longrightarrow (v, e_2')}$. Similar as case above.

  ii. Case $\dfrac{\Gamma \vdash e : \tau_1 \times \tau_2}{\Gamma \vdash e.1 : \tau_1}$. By case analysis on e:

  - e is a value. Then $e = (v_1, v_2)$. Since $\Gamma \vdash e : \tau_1 \times \tau_2$, $\Gamma \vdash v_1 : \tau_1$. Therefore, in this case $e.1 = v_1$, and thus $\Gamma e.1 : \tau_1$.

- e can step. Then there are two ways to step,

  A. $\dfrac{e_1 \rightarrow e_1'}{(e_1, e_2) \rightarrow (e_1', e_2)}$. Since $\Gamma \vdash e_1 : \tau_1$, according to the induction hypothesis, $\Gamma e_1' : \tau_1$. Therefore, $\dfrac{\Gamma \vdash e_1' : \tau_1 \qquad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash (e_1', e_2) : \tau_1 \times \tau_2}$. According to the typing rule, $\Gamma \vdash (e_1', e_2).1 : \tau_1$.

  B. $\dfrac{e_2 \rightarrow e_2'}{(v, e_2) \rightarrow (v, e_2')}$. Similar as case A.

iii. Case $\dfrac{\Gamma \vdash e : \tau_1 \times \tau_2}{\Gamma \vdash e.2 : \tau_2}$. Similar as Case ii.

Problems from after Lecture on Friday 4/17

TBD

STLC with booleans

$$\begin{array}{rcl}
e & ::= & x \mid \lambda x.\, e \mid e\ e \mid b \mid \texttt{if}\ e\ \texttt{then}\ e\ \texttt{else}\ e \\
v & ::= & b \mid \lambda x.\, e \\
\tau & ::= & \mathrm{bool} \mid \tau \longrightarrow \tau \\
\Gamma & \in & Var \rightharpoonup Type
\end{array}$$

$\boxed{e \longrightarrow e}$

$$\frac{}{(\lambda x.\, e)\ v \longrightarrow e[v/x]} \qquad \frac{e_1 \longrightarrow e_1'}{e_1\ e_2 \longrightarrow e_1'\ e_2} \qquad \frac{e_2 \longrightarrow e_2'}{v\ e_2 \longrightarrow v\ e_2'}$$

$$\frac{e_1 \longrightarrow e_1'}{\texttt{if}\ e_1\ \texttt{then}\ e_2\ \texttt{else}\ e_3 \longrightarrow \texttt{if}\ e_1'\ \texttt{then}\ e_2\ \texttt{else}\ e_3}$$

$$\frac{}{\texttt{if true then}\ e_2\ \texttt{else}\ e_3 \longrightarrow e_2} \qquad \frac{}{\texttt{if false then}\ e_2\ \texttt{else}\ e_3 \longrightarrow e_3}$$

Note that we use $\longrightarrow$ for both the small-step semantics and for function types. You can always tell which one we mean by seeing if the arguments are types or expressions.

$\boxed{e_1[e/x]}$

$$\begin{array}{rcll}
x[e/x] & = & e & \\
y[e/x] & = & y & (y \neq x) \\
(\lambda y.\, e_1)[e/x] & = & \lambda y.\, e_1[e/x] & (y \neq x \text{ and } y \notin \mathrm{FV}(e)) \\
(e_1\ e_2)[e/x] & = & e_1[e/x]\ e_2[e/x] & \\
b[e/x] & = & b & \\
(\texttt{if}\ e_1\ \texttt{then}\ e_2\ \texttt{else}\ e_3)[e/x] & = & \texttt{if}\ e_1[e/x] & \\
 & & \texttt{then}\ e_2[e/x] & \\
 & & \texttt{else}\ e_3[e/x] &
\end{array}$$

$\boxed{\mathrm{FV}(e)}$

$$\begin{array}{rcl}
\mathrm{FV}(x) & = & \{x\} \\
\mathrm{FV}(\lambda x.\, e) & = & \mathrm{FV}(e) - \{x\} \\
\mathrm{FV}(e_1\ e_2) & = & \mathrm{FV}(e_1) \cup \mathrm{FV}(e_2) \\
\mathrm{FV}(b) & = & \emptyset \\
\mathrm{FV}(\texttt{if}\ e_1\ \texttt{then}\ e_2\ \texttt{else}\ e_3) & = & \mathrm{FV}(e_1) \cup \mathrm{FV}(e_2) \cup \mathrm{FV}(e_3)
\end{array}$$

We say that e is *closed* if $\mathrm{FV}(e) = \emptyset$.

$\boxed{\Gamma \vdash e : \tau}$

$$\frac{}{\Gamma \vdash b : \mathrm{bool}} \qquad \frac{\Gamma \vdash e_1 : \mathrm{bool} \qquad \Gamma \vdash e_2 : \tau \qquad \Gamma \vdash e_3 : \tau}{\Gamma \vdash \texttt{if}\ e_1\ \texttt{then}\ e_2\ \texttt{else}\ e_3 : \tau} \qquad \frac{x \in \mathrm{dom}\ \Gamma \qquad \Gamma(x) = \tau}{\Gamma \vdash x : \tau}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \longrightarrow \tau_2 \qquad \Gamma \vdash e_2 : \tau_1}{\Gamma \vdash e_1\ e_2 : \tau_2} \qquad \frac{\Gamma[x \mapsto \tau_1] \vdash e : \tau_2}{\Gamma \vdash \lambda x.\, e : \tau_1 \longrightarrow \tau_2}$$