CSE 505: Programming Languages

Lecture 12 — Safely Extending STLC: Progress, Preservation, Lets, Branches

Zach Tatlock
Fall 2013

Review

$$e ::= \lambda x.\, e \mid x \mid e\ e \mid c \qquad \tau ::= \textbf{int} \mid \tau \to \tau$$
$$v ::= \lambda x.\, e \mid c \qquad \Gamma ::= \cdot \mid \Gamma, x : \tau$$

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

$e[e'/x]$: capture-avoiding substitution of e' for free x in e

$$\frac{}{\Gamma \vdash c : \textbf{int}} \qquad \frac{}{\Gamma \vdash x : \Gamma(x)} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda x.\, e : \tau_1 \to \tau_2}$$

$$\frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \qquad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}$$

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

Adding Stuff

Time to use STLC as a foundation for understanding other common language constructs

We will add things via a *principled methodology* thanks to a *proper education*

- Extend the syntax
- Extend the operational semantics
  - Derived forms (syntactic sugar), or
  - Direct semantics
- Extend the type system
- Extend soundness proof (new stuck states, proof cases)

In fact, extensions that add new types have even more structure

Let bindings (CBV)

$$e ::= \ldots \mid \textbf{let } x = e_1 \textbf{ in } e_2$$

$$\frac{e_1 \to e_1'}{\textbf{let } x = e_1 \textbf{ in } e_2 \to \textbf{let } x = e_1' \textbf{ in } e_2} \qquad \frac{}{\textbf{let } x = v \textbf{ in } e \to e[v/x]}$$

$$\frac{\Gamma \vdash e_1 : \tau' \qquad \Gamma, x : \tau' \vdash e_2 : \tau}{\Gamma \vdash \textbf{let } x = e_1 \textbf{ in } e_2 : \tau}$$

(Also need to extend definition of substitution...)

Progress: If e is a let, 1 of the 2 new rules apply (using induction)

Preservation: Uses Substitution Lemma

Substitution Lemma: Uses Weakening and Exchange

Derived forms

let seems just like λ, so can make it a derived form

- **let** $x = e_1$ **in** e_2 "a macro" / "desugars to" $(\lambda x.\ e_2)\ e_1$
- A "derived form"

(Harder if λ needs explicit type)

Or just define the semantics to replace let with λ:

$$\frac{}{\textbf{let } x = e_1 \textbf{ in } e_2 \rightarrow (\lambda x.\ e_2)\ e_1}$$

These 3 semantics are *different* in the state-sequence sense $(e_1 \rightarrow e_2 \rightarrow \ldots \rightarrow e_n)$

- But (totally) *equivalent* and you could prove it (not hard)

Note: ML type-checks let and λ differently (later topic)
Note: Don't desugar early if it hurts error messages!

Booleans and Conditionals

$$\begin{array}{rcl} e & ::= & \ldots \mid \textbf{true} \mid \textbf{false} \mid \textbf{if } e_1\ e_2\ e_3 \\ v & ::= & \ldots \mid \textbf{true} \mid \textbf{false} \\ \tau & ::= & \ldots \mid \textbf{bool} \end{array}$$

$$\frac{e_1 \rightarrow e_1'}{\textbf{if } e_1\ e_2\ e_3 \rightarrow \textbf{if } e_1'\ e_2\ e_3}$$

$$\frac{}{\textbf{if true } e_2\ e_3 \rightarrow e_2} \qquad \frac{}{\textbf{if false } e_2\ e_3 \rightarrow e_3}$$

$$\frac{\Gamma \vdash e_1 : \textbf{bool} \qquad \Gamma \vdash e_2 : \tau \qquad \Gamma \vdash e_3 : \tau}{\Gamma \vdash \textbf{if } e_1\ e_2\ e_3 : \tau}$$

$$\frac{}{\Gamma \vdash \textbf{true} : \textbf{bool}} \qquad \frac{}{\Gamma \vdash \textbf{false} : \textbf{bool}}$$

Also extend definition of substitution (will stop writing that)...
Notes: CBN, new Canonical Forms case, all lemma cases easy