Graduate Programming Languages: 
Type Safety for STLC with Constants

Most of this is available in the slides. However, it can help to see it all in one place.

Syntax

\[

e ::= c | \lambda x. \ e | x | e \ e \\
\nu ::= c | \lambda x. \ e \\
\tau ::= \text{int} | \tau \rightarrow \tau \\
\Gamma ::= \cdot \mid \Gamma, x : \tau
\]

Evaluation Rules (a.k.a. Dynamic Semantics)

\[
e \rightarrow \ e'
\]

E-APPY 
\[
(\lambda x. \ e) \ v \rightarrow e[v/x]
\]

E-APP1 
\[
e_1 \rightarrow e'_1 \\
e_1 \ e_2 \rightarrow e'_1 \ e_2
\]

E-APP2 
\[
e_2 \rightarrow e'_2 \\
v \ e_2 \rightarrow v \ e'_2
\]

Typing Rules (a.k.a. Static Semantics)

\[
\Gamma \vdash e : \tau
\]

T-CONST 
\[
\Gamma \vdash c : \text{int}
\]

T-Var 
\[
\Gamma \vdash x : \Gamma(x)
\]

T-Fun 
\[
\Gamma, x : \tau_1 \vdash e : \tau_2 \\
x \notin \text{Dom}(\Gamma) \\
\Gamma \vdash \lambda x. \ e : \tau_1 \rightarrow \tau_2
\]

T-App 
\[
\Gamma \vdash e_1 : \tau_2 \rightarrow \tau_1 \\
\Gamma \vdash e_2 : \tau_2 \\
\Gamma \vdash e_1 \ e_2 : \tau_1
\]

Type Soundness

Theorem (Type Soundness). If \( \vdash e : \tau \) and \( e \rightarrow^* e' \), then either \( e' \) is a value or there exists an \( e'' \) such that \( e' \rightarrow e'' \).
Proof

The Type Soundness Theorem follows as a simple corollary to the Progress and Preservation Theorems stated and proven below: Given the Preservation Theorem, a trivial induction on the number of steps taken to reach $e'$ from $e$ establishes that $\vdash e' : \tau$. Then the Progress Theorem ensures $e'$ is a value or can step to some $e''$.

We need the following lemma for our proof of Progress, below.

Lemma (Canonical Forms). If $\vdash v : \tau$, then

i If $\tau$ is int, then $v$ is a constant, i.e., some $c$.

ii If $\tau$ is $\tau_1 \rightarrow \tau_2$, then $v$ is a lambda, i.e., $\lambda x. e$ for some $x$ and $e$.

Canonical Forms. The proof is by inspection of the typing rules.

i If $\tau$ is int, then the only rule which lets us give a value this type is T-Const.

ii If $\tau$ is $\tau_1 \rightarrow \tau_2$, then the only rule which lets us give a value this type is T-Fun.

\begin{proof}

Theorem (Progress). If $\vdash e : \tau$, then either $e$ is a value or there exists some $e'$ such that $e \rightarrow e'$.

Progress. The proof is by induction on (the height of) the derivation of $\vdash e : \tau$, proceeding by cases on the bottommost rule used in the derivation.

T-Const $e$ is a constant, which is a value, so we are done.

T-Var Impossible, as $\Gamma$ is $\cdot$.

T-Fun $e$ is $\lambda x. e'$, which is a value, so we are done.

T-App $e$ is $e_1 e_2$.

By inversion, $\vdash e_1 : \tau' \rightarrow \tau$ and $\vdash e_2 : \tau'$ for some $\tau'$. If $e_1$ is not a value, then $\vdash e_1 : \tau' \rightarrow \tau$ and our induction hypothesis ensures $e_1 \rightarrow e'_1$ for some $e'_1$. Therefore, by E-App1, $e_1 e_2 \rightarrow e'_1 e_2$.

Else $e_1$ is a value. If $e_2$ is not a value, then $\vdash e_2 : \tau'$ and our induction hypothesis ensures $e_2 \rightarrow e'_2$ for some $e'_2$. Therefore, by E-App2, $e_1 e_2 \rightarrow e_1 e'_2$.

Else $e_1$ and $e_2$ are values. Then $\vdash e_1 : \tau' \rightarrow \tau$ and the Canonical Forms Lemma ensures $e_1$ is some $\lambda x. e'$. And $(\lambda x. e') e_2 \rightarrow e'[e_2/x]$ by E-Apply, so $e_1 e_2$ can take a step.
\end{proof}
We will need the following lemma for our proof of Preservation, below. Actually, in the proof of Preservation, we need only a Substitution Lemma where $\Gamma$ is $\cdot$, but proving the Substitution Lemma itself requires the stronger induction hypothesis using any $\Gamma$.

**Lemma (Substitution).** If $\Gamma, x:\tau' \vdash e : \tau$ and $\Gamma \vdash e' : \tau'$, then $\Gamma \vdash e[\ell' / x] : \tau$.

To prove this lemma, we will need the following two technical lemmas, which we will assume without proof (they’re not that difficult).

**Lemma (Weakening).** If $\Gamma \vdash e : \tau$ and $x \notin \text{Dom}(\Gamma)$, then $\Gamma, x:\tau' \vdash e : \tau$.

**Lemma (Exchange).** If $\Gamma, y:\tau_2, x:\tau_1 \vdash e : \tau$ and $y \neq x$, then $\Gamma, y:\tau_2, x:\tau_1 \vdash e : \tau$.

Now we prove Substitution.

**Substitution.** The proof is by induction on the derivation of $\Gamma, x:\tau' \vdash e : \tau$. There are four cases. In all cases, we know $\Gamma \vdash e' : \tau'$ by assumption.

**T-Const** $e$ is $c$, so $c[\ell'/x]$ is $c$. By **T-Const**, $\Gamma \vdash c : \text{int}$.

**T-Var** $e$ is $y$ and $\Gamma, x:\tau' \vdash y : \tau$.

If $y \neq x$, then $y[e'/x]$ is $y$. By inversion on the typing rule, we know that $(\Gamma, x:\tau')(y) = \tau$. Since $y \neq x$, we know that $\Gamma(y) = \tau$. So by **T-Var**, $\Gamma \vdash y : \tau$.

If $y = x$, then $y[e'/x]$ is $e'$. $\Gamma, x:\tau' \vdash x : \tau$, so by inversion, $(\Gamma, x:\tau')(x) = \tau$, so $\tau = \tau'$. We know $\Gamma \vdash e' : \tau'$, which is exactly what we need.

**T-App** $e$ is $e_1 e_2$, so $e[e'/x]$ is $(e_1[e'/x]) (e_2[e'/x])$.

We know $\Gamma, x:\tau' \vdash e_1 e_2 : \tau_1$, so, by inversion on the typing rule, we know $\Gamma, x:\tau' \vdash e_1 : \tau_2 \rightarrow \tau_1$ and $\Gamma, x:\tau' \vdash e_2 : \tau_2$ for some $\tau_2$.

Therefore, by induction, $\Gamma \vdash e_1[e'/x] : \tau_2 \rightarrow \tau_1$ and $\Gamma \vdash e_2[e'/x] : \tau_2$.

Given these, **T-App** lets us derive $\Gamma \vdash (e_1[e'/x]) (e_2[e'/x]) : \tau_1$.

So by the definition of substitution $\Gamma \vdash (e_1 e_2)[e'/x] : \tau_1$.

**T-Fun** $e$ is $\lambda y. e_b$, so $e[e'/x]$ is $\lambda y. (e_b[e'/x])$.

We can $\alpha$-convert $\lambda y. e_b$ to ensure $y \notin \text{Dom}(\Gamma)$ and $y \neq x$.

We know $\Gamma, x:\tau' \vdash \lambda y. e_b : \tau_1 \rightarrow \tau_2$, so, by inversion on the typing rule, we know $\Gamma, x:\tau', y:\tau_1 \vdash e_b : \tau_2$.

By Exchange, we know that $\Gamma, y:\tau_1, x:\tau' \vdash e_b : \tau_2$.

By Weakening, we know that $\Gamma, y:\tau_1 \vdash e' : \tau'$.

We have rearranged the two typing judgments so that our induction hypothesis applies (using $\Gamma, y:\tau_1$ for the typing context called $\Gamma$ in the statement of the lemma), so, by induction, $\Gamma, y:\tau_1 \vdash e_b[e'/x] : \tau_2$.

Given this, **T-Fun** lets us derive $\Gamma \vdash \lambda y. e_b[e'/x] : \tau_1 \rightarrow \tau_2$.

So by the definition of substitution, $\Gamma \vdash (\lambda y. e_b)[e'/x] : \tau_1 \rightarrow \tau_2$. 

3
**Theorem** (Preservation). If $\cdot \vdash e : \tau$ and $e \rightarrow e'$, then $\cdot \vdash e' : \tau$.

**Preservation.** The proof is by induction on the derivation of $\cdot \vdash e : \tau$. There are four cases.

T-Const $e$ is $c$. This case is impossible, as there is no $e'$ such that $c \rightarrow e'$.

T-Var $e$ is $x$. This case is impossible, as $x$ cannot be typechecked under the empty context.

T-Fun $e$ is $\lambda x. e_b$. This case is impossible, as there is no $e'$ such that $\lambda x. e_b \rightarrow e'$.

T-App $e$ is $e_1 e_2$, so $\cdot \vdash e_1 e_2 : \tau$.

By inversion on the typing rule, $\cdot \vdash e_1 : \tau_2 \rightarrow \tau$ and $\cdot \vdash e_2 : \tau_2$ for some $\tau_2$.

There are three possible rules for deriving $e_1 e_2 \rightarrow e'$.

E-App1 Then $e' = e'_1 e_2$ and $e_1 \rightarrow e'_1$.
By $\cdot \vdash e_1 : \tau_2 \rightarrow \tau$, $e_1 \rightarrow e'_1$, and induction, $\cdot \vdash e'_1 : \tau_2 \rightarrow \tau$.
Using this and $\cdot \vdash e_2 : \tau_2$, T-App lets us derive $\cdot \vdash e'_1 e_2 : \tau$.

E-App2 Then $e' = e_1 e'_2$ and $e_2 \rightarrow e'_2$.
By $\cdot \vdash e_2 : \tau_2$, $e_2 \rightarrow e'_2$, and induction $\cdot \vdash e'_2 : \tau_2$.
Using this and $\cdot \vdash e_1 : \tau_2 \rightarrow \tau$, T-App lets us derive $\cdot \vdash e_1 e'_2 : \tau$.

E-Apply Then $e_1$ is $\lambda x. e_b$ for some $x$ and $e_b$, and $e' = e_b[e_2/x]$.
By inversion of the typing of $\cdot \vdash e_1 : \tau_2 \rightarrow \tau$, we have $\cdot, x: \tau_2 \vdash e_b : \tau$.
This and $\cdot \vdash e_2 : \tau_2$ lets us use the Substitution Lemma to conclude $\cdot \vdash e_b[e_2/x] : \tau$. 