Type Safety for \textit{ST\lam\cal C} with Constants
CSE 505, Fall 2009

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

Syntax

\[ e ::= c \mid \lambda x. e \mid x \mid e \ e \]
\[ v ::= c \mid \lambda x. e \]
\[ \tau ::= \text{int} \mid \tau \to \tau \]
\[ \Gamma ::= \cdot \mid \Gamma, x: \tau \]

Evaluation Rules

\[ e \rightarrow e' \]

\[
\begin{array}{c}
\text{E-Apply} \\
\frac{(\lambda x. e) \ v \rightarrow e[v/x]}{}
\end{array}
\]

\[
\begin{array}{c}
\text{E-App1} \\
\frac{e_1 \rightarrow e_1'}{e_1 \ e_2 \rightarrow e_1' \ e_2}
\end{array}
\]

\[
\begin{array}{c}
\text{E-App2} \\
\frac{e_2 \rightarrow e_2'}{v \ e_2 \rightarrow v \ e_2'}
\end{array}
\]

Typing Rules

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

\[
\begin{array}{c}
\text{T-Const} \\
\frac{}{\Gamma \vdash c : \text{int}}
\end{array}
\]

\[
\begin{array}{c}
\text{T-Var} \\
\frac{}{\Gamma \vdash x : \Gamma(x)}
\end{array}
\]

\[
\begin{array}{c}
\text{T-Fun} \\
\frac{\Gamma, x : \tau_1 \vdash e : \tau_2 \quad x \notin \text{Dom}(\Gamma)}{\Gamma \vdash \lambda x. e : \tau_1 \to \tau_2}
\end{array}
\]

\[
\begin{array}{c}
\text{T-App} \\
\frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1 \ e_2 : \tau_1}
\end{array}
\]
Type Soundness

Theorem (Type Soundness). If $\cdot \vdash e : \tau$ 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 $\cdot \vdash e' : \tau$. 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 $\cdot \vdash v : \tau$, 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.

\[ \square \]

Theorem (Progress). If $\cdot \vdash e : \tau$, 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 $\cdot \vdash e : \tau$, 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, $\cdot \vdash e_1 : \tau_2 \rightarrow \tau_1$ and $\cdot \vdash e_2 : \tau_2$.

If $e_1$ is not a value, then $\cdot \vdash e_1 : \tau_2 \rightarrow \tau_1$ and the 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 $\cdot \vdash e_2 : \tau_2$ 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 $\cdot \vdash e_1 : \tau_2 \rightarrow \tau_1$ 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.
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 \( \Gamma \vdash e[e'/x] : \tau \).

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 \( x \not\in \text{Dom}(\Gamma) \), then \( \Gamma, x: \tau' \vdash e : \tau \).

**Lemma (Exchange).** If \( \Gamma, x: \tau_1, y: \tau_2 \vdash e : \tau \) 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 \( e[e'/x] \) 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
\[
\Gamma, x: \tau' \vdash e_1 : \tau_2 \rightarrow \tau_1 \quad \text{and} \quad \Gamma, x: \tau' \vdash e_2 : \tau_2 \quad \text{for some} \ \tau_2.
\]

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 \( y \not\in \text{Dom}(\Gamma) \).

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, \( \Gamma \vdash (\lambda y. \, e_b)[e'/x] : \tau_1 \rightarrow \tau_2 \).
Theorem (Preservation). If · ⊢ e : τ and e → e′, then · ⊢ e′ : τ.

Preservation. The proof is by induction on the derivation of · ⊢ e : τ. There are four cases.

**T-Const** e is c. This case is impossible, as there is no e′ such that c → e′.

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

**T-Fun** e is λx. eb. This case is impossible, as there is no e′ such that λx. eb → e′.

**T-App** e is e1 e2, so · ⊢ e1 e2 : τ.

By inversion on the typing rule, · ⊢ e1 : τ2 → τ and · ⊢ e2 : τ2 for some τ2.

There are three possible rules for deriving e1 e2 → e′.

**E-App1** Then e′ = e′1 e2 and e1 → e′1.

By · ⊢ e1 : τ2 → τ, e1 → e′1, and induction, · ⊢ e′1 : τ2 → τ.

Using this and · ⊢ e2 : τ2, T-App lets us derive · ⊢ e′1 e2 : τ.

**E-App2** Then e′ = e1 e′2 and e2 → e′2.

By · ⊢ e2 : τ2, e2 → e′2, and induction · ⊢ e′2 : τ2.

Using this and · ⊢ e1 : τ2 → τ, T-App lets us derive · ⊢ e1 e′2 : τ.

**E-Apply** Then e1 is λx. eb for some x and eb, and e′ = eb[e2/x].

By inversion of the typing of · ⊢ e1 : τ2 → τ, we have ·, x:τ2 ⊢ eb : τ.

This and · ⊢ e2 : τ2 lets us use the Substitution Lemma to conclude · ⊢ eb[e2/x] : τ.