Type Safety for STλC with Constants

Most of this is available in Dan’s slides. However it is good to see all of it in one place.

Syntax

\[
\begin{align*}
e & ::= c \mid \lambda x. e \mid x \mid e \; e \\
v & ::= c \mid \lambda x. e \\
\tau & ::= \text{int} \mid \tau \rightarrow \tau \\
\Gamma & ::= \cdot \mid \Gamma, x : \tau
\end{align*}
\]

Evaluation Rules

\[
\begin{align*}
e & \rightarrow e'
\end{align*}
\]

E-Apply

\[
\frac{}{\left(\lambda x. e\right) v \rightarrow e[v/x]}
\]

E-App1

\[
\frac{e_1 \rightarrow e'_1 \quad e_2 \rightarrow e'_2}{v \; e_2 \rightarrow v \; e'_2}
\]

E-App2

Typing Rules

\[
\begin{align*}
\Gamma & \vdash e : \tau
\end{align*}
\]

T-Const

\[
\frac{}{\Gamma \vdash c : \text{int}}
\]

T-Var

\[
\frac{}{\Gamma \vdash x : \Gamma(x)}
\]

T-Fun

\[
\frac{\Gamma, x : \tau_1 \vdash e : \tau_2 \quad x \not\in \text{Dom}(\Gamma)}{\Gamma \vdash \lambda x. e : \tau_1 \rightarrow \tau_2}
\]

T-App

\[
\frac{\Gamma \vdash e_1 : \tau_2 \rightarrow \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1 \; e_2 : \tau_1}
\]
Proof

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

Lemma (Canonical Forms). If \( e \) is a value and \( \Gamma \vdash e : \tau \), then

i If \( \tau \) is int, \( e \) is of the form \( c \), and

ii If \( \tau \) is \( \tau_1 \to \tau_2 \), \( e \) is of the form \( \lambda x. e' \).

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

i If \( \tau \) is int, the only rule which allows us to give a value this type is T-Const, which
requires that \( e \) be of the form \( c \).

ii If \( \tau \) is \( \tau_1 \to \tau_2 \), the only rule which allows us to give a value this type is T-Fun, which
requires that \( e \) be of the form \( \lambda x. e' \).

\[
\Box
\]

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 \( \Gamma \vdash e : \tau \). There are four cases.

T-Const \( e \) is \( c \), 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, \( \Gamma \vdash e_1 : \tau_2 \to T_1 \) and \( \Gamma \vdash e_2 : \tau_2 \).

If \( e_1 \) is not a value, and we know above that \( \Gamma \vdash e_1 : \tau_2 \to \tau_1 \), so by our IH, \( e_1 \to e'_1 \) for some \( e'_1 \). Therefore, by E-App1, \( e_1 e_2 \to e'_1 e_2 \).

If \( e_1 \) is a value and \( e_2 \) is not a value, and we know above that \( \Gamma \vdash e_2 : \tau_2 \), so by our IH, \( e_2 \to e'_2 \) for some \( e'_2 \). Therefore, by E-App2, \( e_1 e_2 \to e_1 e'_2 \).

If both \( e_1 \) and \( e_2 \) are values, and we know above that \( \Gamma \vdash e_1 : \tau_2 \to \tau_1 \), \( e_1 \) is some \( \lambda x. e' \) by Canonical Forms, so \( \lambda x. e' e_2 \to e'[e_2/x] \) by E-Apply.

\[
\Box
\]

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

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 lemmas, which I will not prove.

**Lemma (Weakening).** If $\Gamma \vdash e : T$, then $\Gamma, x : \tau' \vdash e : \tau$

*Weakening.* By induction on the derivation of $\Gamma \vdash e : \tau$. □

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

*Exchange.* By induction on the derivation of $\Gamma \vdash e : \tau$. □

Now we prove Substitution.

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

### T-Const $e$ is $c$, and $\Gamma, x : \tau' \vdash c : \text{int}$.

- $c[e'/x]$ is $c$, and by T-Const, $\Gamma \vdash c : \text{int}$.

### T-Var $e$ is $\gamma$ 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 $\Gamma(y) = \tau$. Bt T-Var, we know $\Gamma \vdash y : \tau$.
- 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 $\Gamma \vdash e' : \tau'$, so $\Gamma \vdash e' : \tau$.

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

- 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 $\Gamma, x : \tau' \vdash e_2 : \tau_2$.
- By induction, we know that $\Gamma \vdash e_1[e'/x] : \tau_2 \rightarrow \tau_1$ and $\Gamma \vdash e_2[e'/x] : \tau_2$.
- From these, by T-App, we know $\Gamma \vdash (e_1 e_2)[e'/x] : \tau_1$.

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

- We know that $\Gamma, x : \tau' \vdash \lambda y. e_b : \tau_1 \rightarrow \tau_2$, so, by inversion on the typing rule, we know that $\Gamma, x : \tau', y : \tau_1 \vdash e_b : \tau_2$.
- 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, so, by induction, $\Gamma, y : \tau_1 \vdash e_b[e'/x] : \tau_2$.
- By T-Fun, $\Gamma \vdash \lambda y. e_b[e'/x] : \tau_1 \rightarrow \tau_2$.
- By the definition of substitution, $\Gamma \vdash \lambda y. e_b[e'/x] : \tau_1 \rightarrow \tau_2$.

□

**Theorem.** Preservation If $\Gamma \vdash e : \tau$ and $e \rightarrow e'$, then $\Gamma \vdash e' : \tau$.
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 $c$ does not evaluate.

**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 $\lambda x. e_b$ does not evaluate.

**T-App** $e$ is $e_1 e_2$, so $\cdot \vdash e_1 e_2 : \tau_1$.

By inversion on the typing rule, $\cdot \vdash e_1 : \tau_2 \rightarrow \tau_1$ and $\cdot \vdash e_2 : \tau_2$.

There are three cases for $e_1 e_2 \rightarrow e'$.

**E-App1** $e_1 e_2 \rightarrow e'_1 e_2$.

By inversion on the evaluation rule, $e_1 \rightarrow e'_1$.

By induction, $\cdot \vdash e'_1 : \tau_2 \rightarrow \tau_1$.

By T-App, $\cdot \vdash e'_1 e_2 : \tau_1$.

**E-App2** $v e \rightarrow v e'_2$.

By inversion on the evaluation rule, $e_2 \rightarrow e'_2$.

By induction, $\cdot \vdash e'_2 : \tau_2$.

By T-App, $\cdot \vdash v e'_2 : \tau_1$.

**E-Apply** $\lambda x. e_b v \rightarrow e_b[v/x]$.

$e_1$ is $\lambda x. e_b$, and we know $\cdot \vdash e_1 : \tau_2 \tau_1$, so, by inversion on the typing rule, we know $x : \tau_2 \vdash e_b : \tau_1$.

We know $\cdot \vdash e_2 : \tau_2$.

By Substitution, we know $\cdot \vdash e_b[v/x] : \tau_1$.

$\square$