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

$$
\begin{array}{rcl}
e & ::= & c \mid \lambda x.\ e \mid x \mid e\ e \\
v & ::= & c \mid \lambda x.\ e \\
\tau & ::= & \mathsf{int} \mid \tau \to \tau \\
\Gamma & ::= & \cdot \mid \Gamma, x{:}\tau
\end{array}
$$

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

$\boxed{e \to e'}$

$$
\textsc{E-Apply}\ \frac{}{(\lambda x.\ e)\ v \to e[v/x]}
\qquad
\textsc{E-App1}\ \frac{e_1 \to e_1'}{e_1\ e_2 \to e_1'\ e_2}
\qquad
\textsc{E-App2}\ \frac{e_2 \to e_2'}{v\ e_2 \to v\ e_2'}
$$

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

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

$$
\textsc{T-Const}\ \frac{}{\Gamma \vdash c : \mathsf{int}}
\qquad
\textsc{T-Var}\ \frac{}{\Gamma \vdash x : \Gamma(x)}
\qquad
\textsc{T-Fun}\ \frac{\Gamma, x : \tau_1 \vdash e : \tau_2 \qquad x \notin \mathrm{Dom}(\Gamma)}{\Gamma \vdash \lambda x.\ e : \tau_1 \to \tau_2}
$$

$$
\textsc{T-App}\ \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \qquad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}
$$

Type Soundness

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

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 τ *is* int, *then* v *is a constant, i.e., some* c.

ii If τ *is* $\tau_1 \to \tau_2$, *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 τ is int, then the only rule which lets us give a value this type is T-Const.

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

□

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

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 Γ 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, $\cdot \vdash e_1 : \tau' \to \tau$ and $\cdot \vdash e_2 : \tau'$ for some τ'.

If e_1 is not a value, then $\cdot \vdash e_1 : \tau' \to \tau$ and the induction hypothesis ensures $e_1 \to e_1'$ for some e_1'. Therefore, by E-App1, $e_1\ e_2 \to e_1'\ e_2$.

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

Else e_1 and e_2 are values. Then $\cdot \vdash e_1 : \tau' \to \tau$ and the Canonical Forms Lemma ensures e_1 is some $\lambda x.\ e'$. And $(\lambda x.\ e')\ e_2 \to e'[e_2/x]$ 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 Γ is $\cdot$, but proving the Substitution Lemma itself requires the stronger induction hypothesis using any Γ.

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* $x \notin \mathrm{Dom}(\Gamma)$*, 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 $c[e'/x]$ is c. By T-Const, $\Gamma \vdash c : \mathsf{int}$.

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 $\Gamma, x{:}\tau' \vdash e_1 : \tau_2 \to \tau_1$ and $\Gamma, x{:}\tau' \vdash e_2 : \tau_2$ for some τ_2.

Therefore, by induction, $\Gamma \vdash e_1[e'/x] : \tau_2 \to \tau_1$ 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 α-convert $\lambda y.\ e_b$ to ensure $y \notin \mathrm{Dom}(\Gamma)$ and $y \neq x$.

We know $\Gamma, x{:}\tau' \vdash \lambda y.\ e_b : \tau_1 \to \tau_2$, 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 Γ 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 $\Gamma \vdash \lambda y.\ e_b[e'/x] : \tau_1 \to \tau_2$.

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

$\square$

Theorem (Preservation)**.** *If* $\cdot \vdash e : \tau$ *and* $e \to e'$*, 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 $c \to e'$.

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 $\lambda x.\ e_b \to e'$.

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

By inversion on the typing rule, $\cdot \vdash e_1 : \tau_2 \to \tau$ and $\cdot \vdash e_2 : \tau_2$ for some τ_2.

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

E-App1 Then $e' = e_1'\ e_2$ and $e_1 \to e_1'$.
By $\cdot \vdash e_1 : \tau_2 \to \tau$, $e_1 \to e_1'$, and induction, $\cdot \vdash e_1' : \tau_2 \to \tau$.
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 $e_2 \to e_2'$.
By $\cdot \vdash e_2 : \tau_2$, $e_2 \to e_2'$, and induction $\cdot \vdash e_2' : \tau_2$.
Using this and $\cdot \vdash e_1 : \tau_2 \to \tau$, 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 $\cdot \vdash e_1 : \tau_2 \to \tau$, 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 $\cdot \vdash e_b[e_2/x] : \tau$.

$\square$