Logical Foundations of Computer Science
Lecture 19: STLC Properties
Tiago Cogumbreiro
Objectives for today

- Look at a larger-scale formalization of a programming language
- Prove two properties about this language
STLC Properties

1. **Type preservation** (the type of a well-typed term is preserved by reduction):
   If $\{\} \vdash t \in T$ and $t \Rightarrow t'$, then $\{\} \vdash t' \in T$.

2. **Progress** (a well-typed term is either a value or it reduces):
   $\{\} \vdash t \in T$, then either $t$ is a value, or $t \Rightarrow t'$ for some $t'$. 
Type preservation

The interesting case of type preservation is:

\[ \text{HT2} : \text{empty} \vdash v \in \; \text{T}_v \]
\[ \text{HT1} : \text{empty} \vdash \lambda x : \text{T}_v. \; e \in (\text{T}_v \rightarrow \text{T}_e) \]
\[ (\ast \; \{\ast\} \vdash \lambda x : \text{T}_v. \; e \in \text{T}_v \rightarrow \text{T}_e \ast) \]
\[ \text{empty} \vdash [x := v] \; e \in \text{T}_e \]

We can simplify HT1 and get:

\[ \text{HT2} : \text{empty} \vdash v \in \; \text{T}_v \]
\[ \text{H1} : x \rightarrow \text{T}_v \vdash e \in \text{T}_e \]
\[ (\ast \; \{\ast\} \vdash v \in \text{T}_v \ast) \]
\[ (\ast \; \{x : \text{T}_v\} \vdash \lambda x : \text{T}_v. \; e \in \text{T}_v \rightarrow \text{T}_e \ast) \]
\[ \text{empty} \vdash [x := v] \; e \in \text{T}_e \]
Before, we can prove type-preservation, we must show that substitution preserves the type of the expression.
Substitution type-preservation

Restating the previous proof state:

\[
\begin{align*}
HT2 & : \text{empty} \vdash v \in T_v \\
H1 & : x \rightarrow T_v \vdash e \in T_e \\
\text{empty} & \vdash [x := v] e \in T_e
\end{align*}
\]

Notice, in order to know that \( e \) has type \( T_e \) we must know that \( x \) has a type \( T_v \), however the \textbf{typing context} in our goal has no \( x \). The typing context in the goal is \textbf{stronger} than that of \( H1 \).

So, how can this be provable?
Substitution type-preservation

Restating the previous proof state:

\[ \text{HT2 : empty |- } v \text{ in T}_v \]
\[ \text{H1 : x |\rightarrow T}_v |\rightarrow e \text{ in T}_e \]

\[ \text{empty |- } [x := v] e \text{ in T}_e \] (1/1)

Notice, in order to know that \( e \) has type \( T_e \) we must know that \( x \) has a type \( T_v \), however the **typing context** in our goal has no \( x \). The typing context in the goal is **stronger** than that of H1.

So, how can this be provable?

The reason is that \( v \) is well typed with an **empty** context, it doesn't need any typing information to be well typed. Which means, it does not need to know the type of \( x \) and, therefore, we can **strengthen** the typing context of H1 and get that of the goal.
Type preservation

Substitution lemma
**Substitution Lemma**

**Lemma substitution_preserves_typing_try0.** If \{\} ⊢ v ∈ V and \{x ↦ V\} ⊢ t ∈ T, then \{\} ⊢ [x := v]t ∈ T.

The proof follows by induction on the structure of t. We quickly get stuck on the case for T_Abs when \(t = \lambda y: U.t'\) and \(x ≠ y\).

IHt : \(\forall x U v T,\) 
  \(\text{empty} &\{\{x → U\}\} \vdash t \in T → \text{empty} \vdash v \in U → \text{empty} \vdash [x := v]t \in T\)

H0 : \(\text{empty} \vdash v \in V\)

H6 : \(\text{empty} &\{\{x → V; y → U\}\} \vdash t \in T\)

Heq : \(x <> y\)

\[\text{\(\vdash [x := v]t \in T\) (1/1)}\]

\(\text{empty} &\{\{y → U\}\} \vdash [x := v]t \in T\)
Substitution Lemma

**Lemma substitution_preserves_typing_try0.** If \( \left\{ \right\} \vdash v \in V \) and \( \left\{ x \mapsto V \right\} \vdash t \in T \), then \( \left\{ \right\} \vdash [x := v] t \in T \).

The proof follows by induction on the structure of \( t \). We quickly get stuck on the case for \( T\_Abs \) when \( t = \lambda y: U.t' \) and \( x \neq y \).

IHt : forall \( x U v T \),

\[
\begin{align*}
\text{empty} & \land \left\{ x \mapsto U \right\} \vdash t \text{ in } T \rightarrow \text{empty} \\
\text{empty} & \rightarrow v \text{ in } U \rightarrow \text{empty} \\
\text{empty} & \rightarrow [x := v] t \text{ in } T
\end{align*}
\]

H0 : empty \vdash v \text{ in } V

H6 : empty \land \left\{ x \mapsto V ; y \mapsto U \right\} \vdash t \text{ in } T

Heq : \( x \neq y \)

\[\vdash \text{empty} \land \left\{ y \mapsto U \right\} \vdash [x := v] t \text{ in } T\] ————————————————————(1/1)

**We need to prove a stronger result! We need to generalize the context.**

**Lemma.** If \( \left\{ \right\} \vdash v \in V \) and \( \Gamma \& \left\{ x \mapsto V \right\} \vdash t \in T \), then \( \Gamma \vdash [x := v] t \in T \).
Substitution Lemma (1/3)

**Lemma.** If \{\} \vdash v \in V and \Gamma \& \{x \mapsto V\} \vdash t \in T, then \Gamma \vdash [x := v]t \in T.

**Proof.** There are two interesting cases to consider: T_Var and T_Abs. Case T_Var:

Ht' : empty \vdash v \in U
H2 : (\Gamma \& \{x \rightarrow U\}) s = Some T

\[\begin{array}{c}
\hline
1/1
\end{array}\]

\Gamma \vdash \text{if beq_string x s then } v \text{ else } \text{tvar s } \in T

After doing a case analysis on whether \(x = s\) (see goal), we get:

Ht' : empty \vdash v \in T

\[\begin{array}{c}
\hline
1/1
\end{array}\]

\Gamma \vdash v \in T

Let us prove the above in a new lemma: **context weakening**.
Substitution Lemma (2/3)

Case $T_{Abs}$ when $t = \lambda y : T.t_0$ and $x \neq y$.

Let us prove the above in a new lemma: context rearrange.
Substitution Lemma (3/3)

To be able to prove the substitution lemma we need the auxiliary lemmas:

1. **Context weakening:**
   If $\emptyset \vdash v \in T$, then $\Gamma \vdash v \in T$ for any context $\Gamma$.

2. **Context rearrange:**
   If $\Gamma \& \{x \mapsto U; y \mapsto T\} \vdash t \in V$ and $x \neq y$, then $\Gamma \& \{y \mapsto T; x \mapsto U\} \vdash t \in V$
Type preservation

↓

Substitution lemma

↓

Context weakening
Context weakening

**Theorem.** If $\{\} \vdash v \in T$, then $\Gamma \vdash v \in T$ for any context $\Gamma$.

**Lemma** context_weakening:

forall $v$ $T$,

empty $\vdash v \in T \Rightarrow$

forall $\Gamma$, $\Gamma \vdash -v \in T$.

By induction on $v$ we get the following when $v$ is tabs $s \rightarrow t$ $v'$ (after renaming $v'$ to $v$):

IHv : forall $T : ty$, empty $\vdash v \in T \Rightarrow$ forall $\Gamma : context$, $\Gamma \vdash -v \in T$

H5 : empty & {{s \rightarrow t}} $\vdash v \in T_{12}$

We can't use the induction hypothesis. We need a stronger theorem.
Lemma context_weakening:

\[
\forall v \ T, \\
\emptyset \vdash v \in T \rightarrow \\
\forall \Gamma, \Gamma \vdash v \in T.
\]

Proof.

\begin{itemize}
  \item induction \( v \); intros; inversion \( H \); subst; clear \( H \).
  \item inversion \( H_2 \).
  \item eapply \( T_{\text{App}} \); eauto.
  \item apply \( T_{\text{Abs}} \).
\end{itemize}

Abort.
Type preservation

Substitution lemma

Context weakening

Context invariance
Context invariance

Let restricted equivalence of contexts be defined as $\Gamma \equiv |_P \Gamma' := \forall x, P(x) \implies \Gamma(x) = \Gamma'(x)$.

**Theorem.** If $\Gamma \vdash t \in T$ and $\Gamma \equiv |_{\text{free}(t)} \Gamma'$, then $\Gamma' \vdash t \in T$.

**Definition (free variables).** We say that $x$ is free in term $t$, with the following inductive definition:

- $x \in \text{free}(x)$
- $x \in \text{free}(t_1)$ implies $x \in \text{free}(t_2)$ implies $x \in \text{free}(t_1 \, t_2)$
- $x \in \text{free}(\lambda y : T. t)$
- $x \in \text{free}(\text{if } t_1 \text{ then } t_2 \text{ else } t_3)$
- $x \in \text{free}(t_1)$ implies $x \in \text{free}(t_2)$ implies $x \in \text{free}(t_1 \, t_2)$
- $x \not= y$ implies $x \in \text{free}(t)$
- $x \in \text{free}(\text{if } t_1 \text{ then } t_2 \text{ else } t_3)$
- $x \in \text{free}(t_1)$ implies $x \in \text{free}(t_2)$ implies $x \in \text{free}(\text{if } t_1 \text{ then } t_2 \text{ else } t_3)$
Lemma context_invariance : forall Gamma Gamma' t T,
    Gamma |- t \in T \rightarrow
    (forall x, appears_free_in x t \rightarrow Gamma x = Gamma' x) \rightarrow
    Gamma' |- t \in T.

By induction on the derivation of \( \Gamma \vdash t \in T \). The interesting case is that of \( T_{Abs} \), where after applying \( T_{Abs} \) and the induction hypothesis, we get the following proof state.

H0 : forall x : string, appears_free_in x (tabs y T11 t12) \rightarrow Gamma x = Gamma' x
Hafi : appears_free_in x1 t12

\( \begin{align*}
\text{(Gamma & \{\{y \rightarrow T11\}\}) x1} &= (\text{Gamma' & \{\{y \rightarrow T11\}\}) x1} \\
\text{-----------------------------------------------(1/1)}
\end{align*} \)

Which holds by unfolding update and testing whether \( x1 = y \).
Type preservation

Substitution lemma

Context weakening
Lemma context_weakening:

\[
\forall v \ T, \\
\emptyset \vdash v \in T \rightarrow \\
\forall \Gamma, \Gamma \vdash v \in T.
\]

The proof follows by applying lemma context_invariance, which yields the following proof state.

\[
\begin{align*}
H : \emptyset \vdash v \in T \\
H_0 : \text{appears_free_in } x v \\
\hline
\text{empty } x = \Gamma x
\end{align*}
\]

How do we solve this?
Lemma context_weakening:
\[
\forall v, T, \quad \text{empty} \vdash v \in T \Rightarrow \\
\forall \Gamma, \Gamma \vdash v \in T.
\]

The proof follows by applying lemma context_invariance, which yields the following proof state.

\[
\begin{align*}
H &: \text{empty} \vdash v \in T \\
H_0 &: \text{appears_free_in } x \in v \\
\text{-----------------------------------------------(1/1)} \\
\text{empty } x &= \Gamma x
\end{align*}
\]

How do we solve this? Notice, we are saying that there is a free variable in \(v\) and that \(v\) is typable with an empty context.
No free names in an empty context

Lemma typable_empty__closed. If $\{\} \vdash v \in T$, then $x \notin \text{free}(v)$ for any $x$.

A direct proof, by induction on the structure of $v$, quickly leads us astray. Proving negative values is generally more complicated. Instead, show a positive result.

Lemma free_in_context. If $x \in \text{free}(t)$ and $\Gamma \vdash T$, then $\Gamma(x) = T'$ for some type $T'$.

Proof. The proof is trivial and follows induction on the derivation of the first hypothesis.
Progress
Progress

**Theorem progress**: \( \forall t \in T, \) 
\( \text{empty} \vdash t \) \( \text{in} \ T \) \( \rightarrow \) 
\( \text{value} \ t \lor \exists t', t \implies t'. \)