CS720

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:

```
HT2 : empty |- v \in T_v
HT1 : empty |- \ x : T, e \in (T_v → T_e) (* {}  ⊢ λx: T_v. e ∈ T_v → T_e *)
----------------------------------------(1/1)
empty |- [x := v] e \in T_e
```

We can simplify `HT1` and get:

```
HT2 : empty |- v \in T_v     (* {} ⊢ v   ∈   T_v *)
H1 : x |→ T_v |- e \in T_e (* {x:T_v}  ⊢ λx: T_v. e    ∈    T_v → T_e *)
----------------------------------------(1/1)
empty |- [x := v] e \in T_e
```

Type preservation

```
HT2 : empty |- v \in T_v
H1 : x |-> T_v |- e \in T_e
----------------------------------------(1/1)
empty |- [x := v] e \in T_e
```

In English...

	Formula	***Meaning***
Assumption:	$\emptyset \vdash v \in T_v$	v has type T_v
Assumption:	$x \mapsto T_v \vdash e \in T_e$	If x has type T_v, then e has type T_e
Goal:	$\emptyset \vdash [x := v]e \in T_e$	e has type T_e by replacing x by v

> 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:

```
HT2 : empty |- v \in T_v
H1 : x |-> T_v |- e \in T_e
______________________________________(1/1)
 empty |- [x := v] e \in T_e
```

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?

Substitution type-preservation

Restating the previous proof state:

```
HT2 : empty |- v \in T_v
H1 : x |-> T_v |- e \in T_e
______________________________________(1/1)
 empty |- [x := v] e \in T_e
```

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 $\{\} \vdash v \in V$ and $\{x \mapsto V\} \vdash t \in T$, then $\{\} \vdash [x := v]t \in T$.

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,
      empty & {{x ⟶ U}} |- t \in T → empty |- v \in U → empty |- [x := v] t \in T
H0 : empty |- v \in V
H6 : empty & {{x ⟶ V; y ⟶ U}} |- t \in T
Heq : x <> y
______________________________________(1/1)
 empty & {{y ⟶ U}} |- [x := v] t \in T
```

Substitution Lemma

Lemma `substitution_preserves_typing_try0`**.** If $\{\} \vdash v \in V$ and $\{x \mapsto V\} \vdash t \in T$, then $\{\} \vdash [x := v]t \in T$.

The proof follows by induction on the structure of t. We quickly get stuck on the case for `T_Abs` when $t = \lambda y\colon U.t'$ and $x \neq y$.

```
IHt : forall x U v T,
      empty & {{x --> U}} |- t \in T -> empty |- v \in U -> empty |- [x := v] t \in T
H0 : empty |- v \in V
H6 : empty & {{x --> V; y --> U}} |- t \in T
Heq : x <> y
______________________________________(1/1)
  empty & {{y --> U}} |- [x := v] t \in T
```

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

Lemma. If $\{\} \vdash v \in V$ and $\Gamma \& \{x \mapsto V\} \vdash t \in T$, 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 |- v \in U
H2 : (Gamma & {{x --> U}}) s = Some T
______________________________________(1/1)
Gamma |- if beq_string x s then v else tvar s \in T
```

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

```
Ht' : empty |- v \in T
______________________________________(1/1)
Gamma |- v \in T
```

> Let us prove the above in a new lemma: **context weakening**.

Substitution Lemma (2/3)

Case `T_Abs` when $t = \lambda y\colon T.t_0$ and $x \neq y$.

```
Gamma & {{x ⟶ U; y ⟶ T}} |- t0 \in T12
Hxy : x <> y
_______________________________________(1/1)
Gamma & {{y ⟶ T; x ⟶ U}} |- t0 \in T12
```

> 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 $\{\} \vdash v \in T$, then $\Gamma \vdash v \in T$ for any context Γ.
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 Γ.

```
Lemma context_weakening:
  forall v T,
  empty |- v \in T ->
  forall Gamma, Gamma |- v \in T.
```

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

```
IHv : forall T : ty, empty |- v \in T -> forall Gamma : context, Gamma |- v \in T
H5 : empty & {{s --> t}} |- v \in T12
______________________________________(1/1)
Gamma & {{s --> t}} |- v \in T12
```

> **We can't use the induction hypothesis.** We need a stronger theorem.

Context weakening

```
Lemma context_weakening:
  forall v T,
  empty |- v \in T ->
  forall Gamma, Gamma |- v \in T.
Proof.
  induction v; intros; inversion H; subst; clear H.
  - inversion H2.
  - eapply T_App; eauto.
  - apply T_Abs.
    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:

$$\frac{}{x \in \text{free}(x)} \qquad \frac{x \in \text{free}(t_1)}{x \in \text{free}(t_1\ t_2)} \qquad \frac{x \in \text{free}(t_2)}{x \in \text{free}(t_1\ t_2)} \qquad \frac{x \neq y \qquad x \in \text{free}(t)}{x \in \text{free}(\lambda y\colon T.t)}$$

$$\frac{x \in \text{free}(t_1)}{x \in \text{free}(\mathtt{if}\ t_1\ \mathtt{then}\ t_2\ \mathtt{else}\ t_3)} \qquad \frac{x \in \text{free}(t_2)}{x \in \text{free}(\mathtt{if}\ t_1\ \mathtt{then}\ t_2\ \mathtt{else}\ t_3)}$$

$$\frac{x \in \text{free}(t_3)}{x \in \text{free}(\mathtt{if}\ t_1\ \mathtt{then}\ t_2\ \mathtt{else}\ t_3)}$$

Context invariance (proof)

```
Lemma context_invariance : forall Gamma Gamma' t T,
     Gamma |- t \in T  ->
     (forall x, appears_free_in x t -> Gamma x = Gamma' x) ->
     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) ->  Gamma x = Gamma' x
Hafi : appears_free_in x1 t12
______________________________________(1/1)
(Gamma & {{y --> T11}}) x1 = (Gamma' & {{y --> T11}}) x1
```

Which holds by unfolding update and testing whether `x1 = y`.

Type preservation

↓

Substitution lemma

↓

Context weakening

Context weakening (proof)

```
Lemma context_weakening:
  forall v T,
  empty |- v \in T ->
  forall Gamma, Gamma |- v \in T.
```

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

```
H : empty |- v \in T
H0 : appears_free_in x v
______________________________________(1/1)
empty x = Gamma x
```

> How do we solve this?

Context weakening (proof)

```
Lemma context_weakening:
  forall v T,
  empty |- v \in T ->
  forall Gamma, Gamma |- v \in T.
```

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

```
H : empty |- v \in T
H0 : appears_free_in x v
______________________________________(1/1)
empty x = Gamma x
```

> 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 $x \notin \mathrm{free}(v)$ 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 $x \in \mathrm{free}(t)$ 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.

UMass Boston

Progress

Progress

```
Theorem progress : forall t T,
  empty |- t \in T ->
  value t \/ exists t', t ==> t'.
```