CIS 500

Software Foundations

Fall 2005

24 October, 2005

The Simply Typed Lambda-Calculus

Lambda-calculus with booleans

t ::=		*terms*
	x	*variable*
	λx.t	*abstraction*
	t t	*application*
	true	*constant true*
	false	*constant false*
	if t then t else t	*conditional*
v ::=		*values*
	λx.t	*abstraction value*
	true	*true value*
	false	*false value*

Operational semantics

$$(\lambda x.t_{12})\ v_2 \longrightarrow [x \mapsto v_2]t_{12} \qquad \text{(E-AppAbs)}$$

$$\frac{t_1 \longrightarrow t_1'}{t_1\ t_2 \longrightarrow t_1'\ t_2} \qquad \text{(E-App1)}$$

$$\frac{t_2 \longrightarrow t_2'}{v_1\ t_2 \longrightarrow v_1\ t_2'} \qquad \text{(E-App2)}$$

$$\texttt{if true then } t_2 \texttt{ else } t_3 \longrightarrow t_2 \qquad \text{(E-IfTrue)}$$

$$\texttt{if false then } t_2 \texttt{ else } t_3 \longrightarrow t_3 \qquad \text{(E-IfFalse)}$$

$$\frac{t_1 \longrightarrow t_1'}{\texttt{if } t_1 \texttt{ then } t_2 \texttt{ else } t_3 \longrightarrow \texttt{if } t_1' \texttt{ then } t_2 \texttt{ else } t_3} \qquad \text{(E-If)}$$

"Simple Types"

T ::=		*types*
	Bool	*type of booleans*
	T→T	*types of functions*

Typing rules

$$\texttt{true} : \texttt{Bool} \qquad \text{(T-True)}$$

$$\texttt{false} : \texttt{Bool} \qquad \text{(T-False)}$$

$$\frac{t_1 : \texttt{Bool} \qquad t_2 : T \qquad t_3 : T}{\texttt{if } t_1 \texttt{ then } t_2 \texttt{ else } t_3 : T} \qquad \text{(T-If)}$$

Typing rules

$$\texttt{true} : \texttt{Bool} \qquad \text{(T-True)}$$

$$\texttt{false} : \texttt{Bool} \qquad \text{(T-False)}$$

$$\frac{t_1 : \texttt{Bool} \qquad t_2 : T \qquad t_3 : T}{\texttt{if } t_1 \texttt{ then } t_2 \texttt{ else } t_3 : T} \qquad \text{(T-If)}$$

$$\frac{}{x : T} \qquad \text{(T-Var)}$$

Typing rules

$$\texttt{true} : \texttt{Bool} \qquad \text{(T-True)}$$

$$\texttt{false} : \texttt{Bool} \qquad \text{(T-False)}$$

$$\frac{t_1 : \texttt{Bool} \qquad t_2 : T \qquad t_3 : T}{\texttt{if } t_1 \texttt{ then } t_2 \texttt{ else } t_3 : T} \qquad \text{(T-If)}$$

$$\frac{x{:}T \in \Gamma}{\Gamma \vdash x : T} \qquad \text{(T-Var)}$$

Typing rules

$$\Gamma \vdash \texttt{true} : \texttt{Bool} \qquad \text{(T-True)}$$

$$\Gamma \vdash \texttt{false} : \texttt{Bool} \qquad \text{(T-False)}$$

$$\frac{\Gamma \vdash \texttt{t}_1 : \texttt{Bool} \quad \Gamma \vdash \texttt{t}_2 : \texttt{T} \quad \Gamma \vdash \texttt{t}_3 : \texttt{T}}{\Gamma \vdash \texttt{if } \texttt{t}_1 \texttt{ then } \texttt{t}_2 \texttt{ else } \texttt{t}_3 : \texttt{T}} \qquad \text{(T-If)}$$

$$\frac{\texttt{x}:\texttt{T} \in \Gamma}{\Gamma \vdash \texttt{x} : \texttt{T}} \qquad \text{(T-Var)}$$

Typing rules

$$\Gamma \vdash \texttt{true} : \texttt{Bool} \qquad \text{(T-True)}$$

$$\Gamma \vdash \texttt{false} : \texttt{Bool} \qquad \text{(T-False)}$$

$$\frac{\Gamma \vdash \texttt{t}_1 : \texttt{Bool} \quad \Gamma \vdash \texttt{t}_2 : \texttt{T} \quad \Gamma \vdash \texttt{t}_3 : \texttt{T}}{\Gamma \vdash \texttt{if } \texttt{t}_1 \texttt{ then } \texttt{t}_2 \texttt{ else } \texttt{t}_3 : \texttt{T}} \qquad \text{(T-If)}$$

$$\frac{\texttt{x}:\texttt{T} \in \Gamma}{\Gamma \vdash \texttt{x} : \texttt{T}} \qquad \text{(T-Var)}$$

$$\frac{\Gamma, \texttt{x}:\texttt{T}_1 \vdash \texttt{t}_2 : \texttt{T}_2}{\Gamma \vdash \lambda \texttt{x}:\texttt{T}_1.\texttt{t}_2 : \texttt{T}_1 \to \texttt{T}_2} \qquad \text{(T-Abs)}$$

Typing rules

$$\Gamma \vdash \texttt{true} : \texttt{Bool} \qquad \text{(T-True)}$$

$$\Gamma \vdash \texttt{false} : \texttt{Bool} \qquad \text{(T-False)}$$

$$\frac{\Gamma \vdash \texttt{t}_1 : \texttt{Bool} \quad \Gamma \vdash \texttt{t}_2 : \texttt{T} \quad \Gamma \vdash \texttt{t}_3 : \texttt{T}}{\Gamma \vdash \texttt{if } \texttt{t}_1 \texttt{ then } \texttt{t}_2 \texttt{ else } \texttt{t}_3 : \texttt{T}} \qquad \text{(T-If)}$$

$$\frac{\texttt{x}:\texttt{T} \in \Gamma}{\Gamma \vdash \texttt{x} : \texttt{T}} \qquad \text{(T-Var)}$$

$$\frac{\Gamma, \texttt{x}:\texttt{T}_1 \vdash \texttt{t}_2 : \texttt{T}_2}{\Gamma \vdash \lambda \texttt{x}:\texttt{T}_1.\texttt{t}_2 : \texttt{T}_1 \to \texttt{T}_2} \qquad \text{(T-Abs)}$$

$$\frac{\Gamma \vdash \texttt{t}_1 : \texttt{T}_{11} \to \texttt{T}_{12} \quad \Gamma \vdash \texttt{t}_2 : \texttt{T}_{11}}{\Gamma \vdash \texttt{t}_1\ \texttt{t}_2 : \texttt{T}_{12}} \qquad \text{(T-App)}$$

Typing Derivations

What derivations justify the following typing statements?

- $\vdash (\lambda \texttt{x}:\texttt{Bool}.\texttt{x})\ \texttt{true} : \texttt{Bool}$
- $\texttt{f}:\texttt{Bool}\to\texttt{Bool} \vdash \texttt{f (if false then true else false)} : \texttt{Bool}$
- $\texttt{f}:\texttt{Bool}\to\texttt{Bool} \vdash \lambda \texttt{x}:\texttt{Bool}.\ \texttt{f (if x then false else x)} : \texttt{Bool}\to\texttt{Bool}$

Properties of $\lambda_{\rightarrow}$

As before, the fundamental property of the type system we have just defined is soundness with respect to the operational semantics.

Properties of $\lambda_{\rightarrow}$

As before, the fundamental property of the type system we have just defined is soundness with respect to the operational semantics.

1. Progress: A closed, well-typed term is not stuck

   If $\vdash \mathtt{t} : \mathtt{T}$, then either $\mathtt{t}$ is a value or else $\mathtt{t} \longrightarrow \mathtt{t}'$ for some $\mathtt{t}'$.

2. Preservation: Types are preserved by one-step evaluation

   If $\vdash \mathtt{t} : \mathtt{T}$ and $\mathtt{t} \longrightarrow \mathtt{t}'$, then $\vdash \mathtt{t}' : \mathtt{T}$.

Proving progress

Same steps as before...

Proving progress

Same steps as before...

- canonical forms lemma
- progress theorem

Typing rules again (for reference)

$$\Gamma \vdash \texttt{true} : \texttt{Bool} \qquad \text{(T-True)}$$

$$\Gamma \vdash \texttt{false} : \texttt{Bool} \qquad \text{(T-False)}$$

$$\frac{\Gamma \vdash \texttt{t}_1 : \texttt{Bool} \quad \Gamma \vdash \texttt{t}_2 : \texttt{T} \quad \Gamma \vdash \texttt{t}_3 : \texttt{T}}{\Gamma \vdash \texttt{if } \texttt{t}_1 \texttt{ then } \texttt{t}_2 \texttt{ else } \texttt{t}_3 : \texttt{T}} \qquad \text{(T-If)}$$

$$\frac{\texttt{x:T} \in \Gamma}{\Gamma \vdash \texttt{x} : \texttt{T}} \qquad \text{(T-Var)}$$

$$\frac{\Gamma, \texttt{x:T}_1 \vdash \texttt{t}_2 : \texttt{T}_2}{\Gamma \vdash \lambda \texttt{x:T}_1 . \texttt{t}_2 : \texttt{T}_1 \to \texttt{T}_2} \qquad \text{(T-Abs)}$$

$$\frac{\Gamma \vdash \texttt{t}_1 : \texttt{T}_{11} \to \texttt{T}_{12} \quad \Gamma \vdash \texttt{t}_2 : \texttt{T}_{11}}{\Gamma \vdash \texttt{t}_1\ \texttt{t}_2 : \texttt{T}_{12}} \qquad \text{(T-App)}$$

Canonical Forms

Lemma:

Canonical Forms

Lemma:

1. If v is a value of type Bool, then

Canonical Forms

Lemma:

1. If v is a value of type Bool, then v is either true or false.

Canonical Forms

Lemma:

1. If `v` is a value of type `Bool`, then `v` is either `true` or `false`.
2. If `v` is a value of type $T_1 \to T_2$, then

Canonical Forms

Lemma:

1. If `v` is a value of type `Bool`, then `v` is either `true` or `false`.
2. If `v` is a value of type $T_1 \to T_2$, then `v` has the form $\lambda x{:}T_1.t_2$.

Progress

Theorem: Suppose `t` is a closed, well-typed term (that is, $\vdash t : T$ for some `T`). Then either `t` is a value or else there is some `t'` with $t \longrightarrow t'$.

Proof: By induction

Progress

Theorem: Suppose `t` is a closed, well-typed term (that is, $\vdash t : T$ for some `T`). Then either `t` is a value or else there is some `t'` with $t \longrightarrow t'$.

Proof: By induction on typing derivations.

Progress

Theorem: Suppose $\mathtt{t}$ is a closed, well-typed term (that is, $\vdash \mathtt{t} : \mathtt{T}$ for some $\mathtt{T}$). Then either $\mathtt{t}$ is a value or else there is some $\mathtt{t}'$ with $\mathtt{t} \longrightarrow \mathtt{t}'$.

Proof: By induction on typing derivations. The cases for boolean constants and conditions are the same as before. The variable case is trivial (because $\mathtt{t}$ is closed). The abstraction case is immediate, since abstractions are values.

Progress

Theorem: Suppose $\mathtt{t}$ is a closed, well-typed term (that is, $\vdash \mathtt{t} : \mathtt{T}$ for some $\mathtt{T}$). Then either $\mathtt{t}$ is a value or else there is some $\mathtt{t}'$ with $\mathtt{t} \longrightarrow \mathtt{t}'$.

Proof: By induction on typing derivations. The cases for boolean constants and conditions are the same as before. The variable case is trivial (because $\mathtt{t}$ is closed). The abstraction case is immediate, since abstractions are values.

Consider the case for application, where $\mathtt{t} = \mathtt{t}_1\ \mathtt{t}_2$ with $\vdash \mathtt{t}_1 : \mathtt{T}_{11} \to \mathtt{T}_{12}$ and $\vdash \mathtt{t}_2 : \mathtt{T}_{11}$.

Progress

Theorem: Suppose $\mathtt{t}$ is a closed, well-typed term (that is, $\vdash \mathtt{t} : \mathtt{T}$ for some $\mathtt{T}$). Then either $\mathtt{t}$ is a value or else there is some $\mathtt{t}'$ with $\mathtt{t} \longrightarrow \mathtt{t}'$.

Proof: By induction on typing derivations. The cases for boolean constants and conditions are the same as before. The variable case is trivial (because $\mathtt{t}$ is closed). The abstraction case is immediate, since abstractions are values.

Consider the case for application, where $\mathtt{t} = \mathtt{t}_1\ \mathtt{t}_2$ with $\vdash \mathtt{t}_1 : \mathtt{T}_{11} \to \mathtt{T}_{12}$ and $\vdash \mathtt{t}_2 : \mathtt{T}_{11}$. By the induction hypothesis, either $\mathtt{t}_1$ is a value or else it can make a step of evaluation, and likewise $\mathtt{t}_2$.

Progress

Theorem: Suppose $\mathtt{t}$ is a closed, well-typed term (that is, $\vdash \mathtt{t} : \mathtt{T}$ for some $\mathtt{T}$). Then either $\mathtt{t}$ is a value or else there is some $\mathtt{t}'$ with $\mathtt{t} \longrightarrow \mathtt{t}'$.

Proof: By induction on typing derivations. The cases for boolean constants and conditions are the same as before. The variable case is trivial (because $\mathtt{t}$ is closed). The abstraction case is immediate, since abstractions are values.

Consider the case for application, where $\mathtt{t} = \mathtt{t}_1\ \mathtt{t}_2$ with $\vdash \mathtt{t}_1 : \mathtt{T}_{11} \to \mathtt{T}_{12}$ and $\vdash \mathtt{t}_2 : \mathtt{T}_{11}$. By the induction hypothesis, either $\mathtt{t}_1$ is a value or else it can make a step of evaluation, and likewise $\mathtt{t}_2$. If $\mathtt{t}_1$ can take a step, then rule E-App1 applies to $\mathtt{t}$. If $\mathtt{t}_1$ is a value and $\mathtt{t}_2$ can take a step, then rule E-App2 applies. Finally, if both $\mathtt{t}_1$ and $\mathtt{t}_2$ are values, then the canonical forms lemma tells us that $\mathtt{t}_1$ has the form $\lambda\mathtt{x}{:}\mathtt{T}_{11}.\mathtt{t}_{12}$, and so rule E-AppAbs applies to $\mathtt{t}$.

Progress

Theorem: Suppose t is a closed, well-typed term (that is, $\vdash t : T$ for some T). Then either t is a value or else there is some t' with $t \longrightarrow t'$.

Proof: By induction on typing derivations. The cases for boolean constants and conditions are the same as before. The variable case is trivial (because t is closed). The abstraction case is immediate, since abstractions are values.

Consider the case for application, where $t = t_1\ t_2$ with $\vdash t_1 : T_{11} \to T_{12}$ and $\vdash t_2 : T_{11}$. By the induction hypothesis, either t_1 is a value or else it can make a step of evaluation, and likewise t_2. If t_1 can take a step, then rule E-App1 applies to t. If t_1 is a value and t_2 can take a step, then rule E-App2 applies. Finally, if both t_1 and t_2 are values, then the canonical forms lemma tells us that t_1 has the form $\lambda x{:}T_{11}.t_{12}$, and so rule E-AppAbs applies to t.

What if t weren't closed?

Proving Preservation

Theorem: If $\vdash t : T$ and $t \longrightarrow t'$, then $\vdash t' : T$.

Proof: By induction

Proving Preservation

Theorem: If $\vdash t : T$ and $t \longrightarrow t'$, then $\vdash t' : T$.

Proof: By induction on typing derivations.
[Which case is the hard one?]

Proving Preservation

Theorem: If $\vdash t : T$ and $t \longrightarrow t'$, then $\vdash t' : T$.

Proof: By induction on typing derivations.
[Which case is the hard one?]

Case T-App: Given $t = t_1\ t_2$

$\vdash t_1 : T_{11} \to T_{12}$

$\vdash t_2 : T_{11}$

$T = T_{12}$

Show $\vdash t' : T_{12}$

Proving Preservation

Theorem: If $\vdash t : T$ and $t \longrightarrow t'$, then $\vdash t' : T$.

Proof: By induction on typing derivations.
[Which case is the hard one?]

Case T-App: Given $t = t_1\ t_2$

$\vdash t_1 : T_{11} \to T_{12}$

$\vdash t_2 : T_{11}$

$T = T_{12}$

Show $\vdash t' : T_{12}$

By the inversion lemma for evaluation, there are three subcases...

Proving Preservation

Theorem: If $\vdash t : T$ and $t \longrightarrow t'$, then $\vdash t' : T$.

Proof: By induction on typing derivations.
[Which case is the hard one?]

Case T-App: Given $t = t_1\ t_2$

$\vdash t_1 : T_{11} \to T_{12}$

$\vdash t_2 : T_{11}$

$T = T_{12}$

Show $\vdash t' : T_{12}$

By the inversion lemma for evaluation, there are three subcases...

Subcase: $t_1 = \lambda x{:}T_{11}.\ t_{12}$

t_2 a value v_2

$t' = [x \mapsto v_2]t_{12}$

Proving Preservation

Theorem: If $\vdash t : T$ and $t \longrightarrow t'$, then $\vdash t' : T$.

Proof: By induction on typing derivations.
[Which case is the hard one?]

Case T-App: Given $t = t_1\ t_2$

$\vdash t_1 : T_{11} \to T_{12}$

$\vdash t_2 : T_{11}$

$T = T_{12}$

Show $\vdash t' : T_{12}$

By the inversion lemma for evaluation, there are three subcases...

Subcase: $t_1 = \lambda x{:}T_{11}.\ t_{12}$

t_2 a value v_2

$t' = [x \mapsto v_2]t_{12}$

Uh oh.

The "Substitution Lemma"

Lemma: Types are preserved under substitition.

If $x{:}S \vdash t : T$ and $\vdash s : S$, then $\vdash [x \mapsto s]t : T$.

The "Substitution Lemma"

Lemma: Types are preserved under substititition.

If $\texttt{x:S} \vdash \texttt{t : T}$ and $\vdash \texttt{s : S}$, then $\vdash [\texttt{x} \mapsto \texttt{s}]\texttt{t : T}$.

Proof: ...

Strengthened induction hypothesis

Lemma: Types are preserved under substititition.

If $\Gamma, \texttt{x:S} \vdash \texttt{t : T}$ and $\Gamma \vdash \texttt{s : S}$, then $\Gamma \vdash [\texttt{x} \mapsto \texttt{s}]\texttt{t : T}$.

Lemmas about the context

Lemma [Permutation]: If $\Gamma \vdash \texttt{t : T}$ and Δ is a permutation of Γ then $\Delta \vdash \texttt{t : T}$. Moreover the latter derivation has the same depth as the former.

Lemma [Weakening]: If $\Gamma \vdash \texttt{t : T}$ and $\texttt{x} \notin \mathit{dom}(\Gamma)$ then $\Gamma, \texttt{x : S} \vdash \texttt{t : T}$. Moreover the latter derivation has the same depth as the former.

Summary

- Typing rules need a context to deal with variables.
- Progress lemma must have an empty context to be true.
- Preservation lemma need not.
- Application case of preservation lemma needs substitution lemma.
- Substitution lemma must be strenthened to non-empty contexts.
- Substitution lemma cannot be proved by induction on typing derivation.