The Simply Typed Lambda-Calculus

The simply typed lambda-calculus

The system we are about to define is commonly called the *simply typed lambda-calculus*, or $\lambda_{\rightarrow}$ for short.

Unlike the untyped lambda-calculus, the "pure" form of $\lambda_{\rightarrow}$ (with no primitive values or operations) is not very interesting; to talk about $\lambda_{\rightarrow}$, we always begin with some set of "base types."

- So, strictly speaking, there are *many* variants of $\lambda_{\rightarrow}$, depending on the choice of base types.
- For now, we'll work with a variant constructed over the booleans.

Untyped lambda-calculus with booleans

$\mathtt{t}$	$::=$		*terms*
		$\mathtt{x}$	*variable*
		$\lambda\mathtt{x.t}$	*abstraction*
		$\mathtt{t\ t}$	*application*
		$\mathtt{true}$	*constant true*
		$\mathtt{false}$	*constant false*
		$\mathtt{if\ t\ then\ t\ else\ t}$	*conditional*
$\mathtt{v}$	$::=$		*values*
		$\lambda\mathtt{x.t}$	*abstraction value*
		$\mathtt{true}$	*true value*
		$\mathtt{false}$	*false value*

"Simple Types"

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

Type Annotations

We now have a choice to make. Do we...

- annotate lambda-abstractions with the expected type of the argument

  $$\lambda \mathtt{x}:\mathtt{T}_1.\ \mathtt{t}_2$$

  (as in most mainstream programming languages), or
- continue to write lambda-abstractions as before

  $$\lambda \mathtt{x}.\ \mathtt{t}_2$$

  and ask the typing rules to "guess" an appropriate annotation (as in OCaml)?

Both are reasonable choices, but the first makes the job of defining the typin rules simpler. Let's take this choice for now.

Typing rules

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

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

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

Typing rules

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

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

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

$$\frac{???}{\lambda \texttt{x}:\texttt{T}_1.\texttt{t}_2 : \texttt{T}_1 \rightarrow \texttt{T}_2} \qquad \text{(T-Abs)}$$

Typing rules

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

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

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

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

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

Typing rules

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

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

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

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

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

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

Typing Derivations

What derivations justify the following typing statements?

- $\vdash$ `(λx:Bool.x) true` : `Bool`
- `f:Bool→Bool` $\vdash$ `f (if false then true else false)` : `Bool`
- `f:Bool→Bool` $\vdash$ `λx:Bool. f (if x then false else x)` : `Bool→Bool`

Properties of $\lambda_{\rightarrow}$

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 $\vdash t : T$*, then either* t *is a value or else* $t \longrightarrow t'$ *for some* t'.

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

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

Proving progress

Same steps as before...

Proving progress

Same steps as before...

- inversion lemma for typing relation
- canonical forms lemma
- progress theorem

Inversion

Lemma:

1. If $\Gamma \vdash \texttt{true} : \texttt{R}$, then $\texttt{R} = \texttt{Bool}$.
2. If $\Gamma \vdash \texttt{false} : \texttt{R}$, then $\texttt{R} = \texttt{Bool}$.
3. If $\Gamma \vdash \texttt{if } \texttt{t}_1 \texttt{ then } \texttt{t}_2 \texttt{ else } \texttt{t}_3 : \texttt{R}$, then $\Gamma \vdash \texttt{t}_1 : \texttt{Bool}$ and $\Gamma \vdash \texttt{t}_2, \texttt{t}_3 : \texttt{R}$.

Inversion

Lemma:

1. If $\Gamma \vdash \mathtt{true} : \mathtt{R}$, then $\mathtt{R} = \mathtt{Bool}$.
2. If $\Gamma \vdash \mathtt{false} : \mathtt{R}$, then $\mathtt{R} = \mathtt{Bool}$.
3. If $\Gamma \vdash \mathtt{if}\ \mathtt{t}_1\ \mathtt{then}\ \mathtt{t}_2\ \mathtt{else}\ \mathtt{t}_3 : \mathtt{R}$, then $\Gamma \vdash \mathtt{t}_1 : \mathtt{Bool}$ and $\Gamma \vdash \mathtt{t}_2, \mathtt{t}_3 : \mathtt{R}$.
4. If $\Gamma \vdash \mathtt{x} : \mathtt{R}$, then

Inversion

Lemma:

1. If $\Gamma \vdash \texttt{true} : \texttt{R}$, then $\texttt{R} = \texttt{Bool}$.
2. If $\Gamma \vdash \texttt{false} : \texttt{R}$, then $\texttt{R} = \texttt{Bool}$.
3. If $\Gamma \vdash \texttt{if } \texttt{t}_1 \texttt{ then } \texttt{t}_2 \texttt{ else } \texttt{t}_3 : \texttt{R}$, then $\Gamma \vdash \texttt{t}_1 : \texttt{Bool}$ and $\Gamma \vdash \texttt{t}_2, \texttt{t}_3 : \texttt{R}$.
4. If $\Gamma \vdash \texttt{x} : \texttt{R}$, then $\texttt{x}{:}\texttt{R} \in \Gamma$.

Inversion

Lemma:

1. If $\Gamma \vdash \texttt{true} : \texttt{R}$, then $\texttt{R} = \texttt{Bool}$.
2. If $\Gamma \vdash \texttt{false} : \texttt{R}$, then $\texttt{R} = \texttt{Bool}$.
3. If $\Gamma \vdash \texttt{if } \texttt{t}_1 \texttt{ then } \texttt{t}_2 \texttt{ else } \texttt{t}_3 : \texttt{R}$, then $\Gamma \vdash \texttt{t}_1 : \texttt{Bool}$ and $\Gamma \vdash \texttt{t}_2, \texttt{t}_3 : \texttt{R}$.
4. If $\Gamma \vdash \texttt{x} : \texttt{R}$, then $\texttt{x:R} \in \Gamma$.
5. If $\Gamma \vdash \lambda\texttt{x:T}_1\texttt{.t}_2 : \texttt{R}$, then

Inversion

Lemma:

1. If $\Gamma \vdash \texttt{true} : \texttt{R}$, then $\texttt{R} = \texttt{Bool}$.
2. If $\Gamma \vdash \texttt{false} : \texttt{R}$, then $\texttt{R} = \texttt{Bool}$.
3. If $\Gamma \vdash \texttt{if } \texttt{t}_1 \texttt{ then } \texttt{t}_2 \texttt{ else } \texttt{t}_3 : \texttt{R}$, then $\Gamma \vdash \texttt{t}_1 : \texttt{Bool}$ and $\Gamma \vdash \texttt{t}_2, \texttt{t}_3 : \texttt{R}$.
4. If $\Gamma \vdash \texttt{x} : \texttt{R}$, then $\texttt{x}{:}\texttt{R} \in \Gamma$.
5. If $\Gamma \vdash \lambda \texttt{x}{:}\texttt{T}_1 . \texttt{t}_2 : \texttt{R}$, then $\texttt{R} = \texttt{T}_1 \to \texttt{R}_2$ for some $\texttt{R}_2$ with $\Gamma, \texttt{x}{:}\texttt{T}_1 \vdash \texttt{t}_2 : \texttt{R}_2$.

Inversion

Lemma:

1. If $\Gamma \vdash \texttt{true} : \texttt{R}$, then $\texttt{R} = \texttt{Bool}$.
2. If $\Gamma \vdash \texttt{false} : \texttt{R}$, then $\texttt{R} = \texttt{Bool}$.
3. If $\Gamma \vdash \texttt{if } \texttt{t}_1 \texttt{ then } \texttt{t}_2 \texttt{ else } \texttt{t}_3 : \texttt{R}$, then $\Gamma \vdash \texttt{t}_1 : \texttt{Bool}$ and $\Gamma \vdash \texttt{t}_2, \texttt{t}_3 : \texttt{R}$.
4. If $\Gamma \vdash \texttt{x} : \texttt{R}$, then $\texttt{x}:\texttt{R} \in \Gamma$.
5. If $\Gamma \vdash \lambda \texttt{x}:\texttt{T}_1.\texttt{t}_2 : \texttt{R}$, then $\texttt{R} = \texttt{T}_1 \rightarrow \texttt{R}_2$ for some $\texttt{R}_2$ with $\Gamma, \texttt{x}:\texttt{T}_1 \vdash \texttt{t}_2 : \texttt{R}_2$.
6. If $\Gamma \vdash \texttt{t}_1\ \texttt{t}_2 : \texttt{R}$, then

Inversion

Lemma:

1. If $\Gamma \vdash \texttt{true} : \texttt{R}$, then $\texttt{R} = \texttt{Bool}$.
2. If $\Gamma \vdash \texttt{false} : \texttt{R}$, then $\texttt{R} = \texttt{Bool}$.
3. If $\Gamma \vdash \texttt{if } \texttt{t}_1 \texttt{ then } \texttt{t}_2 \texttt{ else } \texttt{t}_3 : \texttt{R}$, then $\Gamma \vdash \texttt{t}_1 : \texttt{Bool}$ and $\Gamma \vdash \texttt{t}_2, \texttt{t}_3 : \texttt{R}$.
4. If $\Gamma \vdash \texttt{x} : \texttt{R}$, then $\texttt{x}{:}\texttt{R} \in \Gamma$.
5. If $\Gamma \vdash \lambda \texttt{x}{:}\texttt{T}_1.\texttt{t}_2 : \texttt{R}$, then $\texttt{R} = \texttt{T}_1 \to \texttt{R}_2$ for some $\texttt{R}_2$ with $\Gamma, \texttt{x}{:}\texttt{T}_1 \vdash \texttt{t}_2 : \texttt{R}_2$.
6. If $\Gamma \vdash \texttt{t}_1\ \texttt{t}_2 : \texttt{R}$, then there is some type $\texttt{T}_{11}$ such that $\Gamma \vdash \texttt{t}_1 : \texttt{T}_{11} \to \texttt{R}$ and $\Gamma \vdash \texttt{t}_2 : \texttt{T}_{11}$.

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

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

Progress

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

Proof: By induction on typing derivations.

Progress

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

Proof: By induction on typing derivations. The cases for boolean constants and conditions are the same as before. The variable case is trivial (because $\texttt{t}$ 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 $\vdash \mathtt{t}_1 : \mathtt{T}_{11} \rightarrow \mathtt{T}_{12}$ 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 $\mathtt{t}$.