

Chapter 9: Simply Typed Lambda-Calculus

Function Types

The Typing Relation

Properties of Typing

The Curry-Howard Correspondence

Erasure and Typability



Function Types



- $T_1 \rightarrow T_2$

- classifying functions that expect arguments of type T_1 and return results of type T_2 .

(The type constructor \rightarrow is **right-associative**.)

$T_1 \rightarrow T_2 \rightarrow T_3$ stands for $T_1 \rightarrow (T_2 \rightarrow T_3)$)

- We will consider booleans with lambda calculus

- $T ::= \text{Bool}$
 $T \rightarrow T$

- Examples

- $\text{Bool} \rightarrow \text{Bool}$
- $(\text{Bool} \rightarrow \text{Bool}) \rightarrow (\text{Bool} \rightarrow \text{Bool})$



The Typing Relation

Syntax

$t ::=$	x	terms: variable
	$\lambda x:T. t$	abstraction
	$t t$	application
$v ::=$	$\lambda x:T. t$	values: abstraction value
$T ::=$	$T \rightarrow T$	types: type of functions
$\Gamma ::=$	\emptyset	contexts: empty context
	$\Gamma, x:T$	term variable binding

Assume all variables in Γ are different

Evaluation

	$t_1 \rightarrow t'_1$	$t_1 t_2 \rightarrow t'_1 t_2$	(E-APP1)
	$t_2 \rightarrow t'_2$	$v_1 t_2 \rightarrow v_1 t'_2$	(E-APP2)
	$(\lambda x:T_{11}. t_{12}) v_2 \rightarrow [x \mapsto v_2] t_{12}$		(E-APPABS)

Typing

	$\frac{x:T \in \Gamma}{\Gamma \vdash x : T}$	(T-VAR)
	$\frac{\Gamma, x:T_1 \vdash t_2 : T_2}{\Gamma \vdash \lambda x:T_1. t_2 : T_1 \rightarrow T_2}$	(T-ABS)
	$\frac{\Gamma \vdash t_1 : T_{11} \rightarrow T_{12} \quad \Gamma \vdash t_2 : T_{11}}{\Gamma \vdash t_1 t_2 : T_{12}}$	(T-APP)



Type Derivation Tree

$$\frac{\frac{\frac{x:\text{Bool} \in x:\text{Bool}}{x:\text{Bool} \vdash x:\text{Bool}} \text{T-VAR}}{\vdash \lambda x:\text{Bool}.x:\text{Bool} \rightarrow \text{Bool}} \text{T-ABS} \quad \frac{}{\vdash \text{true}:\text{Bool}} \text{T-TRUE}}{\vdash (\lambda x:\text{Bool}.x) \text{true}:\text{Bool}} \text{T-APP}$$



Properties of Typing

Inversion Lemma

Uniqueness of Types

Canonical Forms

Safety: Progress + Preservation



Inversion Lemma

LEMMA [INVERSION OF THE TYPING RELATION]:

1. If $\Gamma \vdash x : R$, then $x : R \in \Gamma$.
2. If $\Gamma \vdash \lambda x : T_1. t_2 : R$, then $R = T_1 \rightarrow R_2$ for some R_2 with $\Gamma, x : T_1 \vdash t_2 : R_2$.
3. If $\Gamma \vdash t_1 t_2 : R$, then there is some type T_{11} such that $\Gamma \vdash t_1 : T_{11} \rightarrow R$ and $\Gamma \vdash t_2 : T_{11}$.
4. If $\Gamma \vdash \text{true} : R$, then $R = \text{Bool}$.
5. If $\Gamma \vdash \text{false} : R$, then $R = \text{Bool}$.
6. If $\Gamma \vdash \text{if } t_1 \text{ then } t_2 \text{ else } t_3 : R$, then $\Gamma \vdash t_1 : \text{Bool}$ and $\Gamma \vdash t_2, t_3 : R$. \square

Exercise: Is there any context Γ and type T such that $\Gamma \vdash x x : T$?



Uniqueness of Types

- **Theorem** [Uniqueness of Types]: In a given typing context Γ , a term t (with free variables all in the domain of Γ) has **at most one type**. Moreover, there is just **one derivation** of this typing built from the inference rules that generate the typing relation.



Canonical Form



- **Lemma [Canonical Forms]:**
 - If v is a value of type `Bool`, then v is either `true` or `false`.
 - If v is a value of type $T_1 \rightarrow T_2$, then $v = \lambda x:T_1.t_2$.



Progress

- **Theorem** [Progress]: 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 $t \rightarrow t'$.

Proof: By induction on typing derivations.



Two Structural Lemmas



- **Lemma [Permutation]**: If $\Gamma \vdash t : T$ and Δ is a permutation of Γ , then $\Delta \vdash t : T$.
- **Lemma [Weakening]**: If $\Gamma \vdash t : T$ and x is not in $\text{dom}(\Gamma)$, then $\Gamma, x : S \vdash t : T$.

Note: All can be easily proved by induction on derivation



Preservation



- **Lemma** [Preservation of types under substitution]:
If $\Gamma, x:S \vdash t:T$ and $\Gamma \vdash s:S$,
then $\Gamma \vdash [x \rightarrow s]t:T$.

Proof: By induction on a derivation of $\Gamma, x:S \vdash t : T$.

- **Theorem** [Preservation]:
If $\Gamma \vdash t:T$ and $t \rightarrow t'$, then $\Gamma \vdash t' :T$.



The Curry-Howard Correspondence

- A connection between logic and type theory

LOGIC	PROGRAMMING LANGUAGES
propositions	types
proposition $P \supset Q$	type $P \rightarrow Q$
proposition $P \wedge Q$	type $P \times Q$ (see §11.6)
proof of proposition P	term t of type P
proposition P is provable	type P is inhabited (by some term)



Erasure and Typability

- Types are used during type checking, but do not appear in the compiled form of the program.

DEFINITION: The *erasure* of a simply typed term t is defined as follows:

$$\begin{aligned} \text{erase}(x) &= x \\ \text{erase}(\lambda x:T_1. t_2) &= \lambda x. \text{erase}(t_2) \\ \text{erase}(t_1 t_2) &= \text{erase}(t_1) \text{erase}(t_2) \end{aligned}$$

THEOREM:

1. If $t \rightarrow t'$ under the typed evaluation relation, then $\text{erase}(t) \rightarrow \text{erase}(t')$.
2. If $\text{erase}(t) \rightarrow m'$ under the typed evaluation relation, then there is a simply typed term t' such that $t \rightarrow t'$ and $\text{erase}(t') = m'$. □

↓
Untyped?



Curry-Style vs. Church-Style



- Curry Style
 - Syntax \rightarrow Semantics \rightarrow Typing
 - Often used for implicit typed languages
- Church Style
 - Syntax \rightarrow Typing \rightarrow Semantics
 - Often used for explicit typed languages



Homework

- Read Chapter 9.
- Do Exercise 9.3.9.

