Chapter 9: Simply Typed Lambda-Calculus

- Function Types
- The Typing Relation
- Properties of Typing
- The Curry-Howard Correspondence
- Erasure and Typability
Function Types

- **T₁→T₂**
  - classifying functions that expect arguments of type T₁ and return results of type T₂.
  
  (The type constructor → is right-associative. 
  T₁→T₂→T₃ stands for T₁→(T₂→T₃) )

- We will consider Booleans with lambda calculus
  - T ::= Bool
    
    T → T

- Examples
  - Bool→Bool
  - (Bool→Bool) → (Bool→Bool)
Assume all variables in $\Gamma$ are different.
Type Derivation Tree

\[
\frac{x : \text{Bool}}{x : \text{Bool} \vdash x : \text{Bool}} \quad \text{T-VAR}
\]

\[
\frac{x : \text{Bool} \vdash x : \text{Bool}}{\vdash \lambda x : \text{Bool}. x : \text{Bool} \rightarrow \text{Bool}} \quad \text{T-ABS}
\]

\[
\frac{\vdash \text{true} : \text{Bool}}{\vdash (\lambda x : \text{Bool}. x) \text{true} : \text{Bool}} \quad \text{T-APP}
\]
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 $\Gamma \vdash t_2, t_3 : R$. $\Box$

**Exercise:** Is there any context $\Gamma$ and type $T$ such that $\Gamma \vdash x : T$?
Uniqueness of Types

- **Theorem** [Uniqueness of Types]: In a given typing context $\Gamma$, a term $t$ (with free variables all in the domain of $\Gamma$) 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 $\text{Bool}$, 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 $\Delta$ is a permutation of $\Gamma$, 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 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

<table>
<thead>
<tr>
<th>LOGIC</th>
<th>PROGRAMMING LANGUAGES</th>
</tr>
</thead>
<tbody>
<tr>
<td>propositions</td>
<td>types</td>
</tr>
<tr>
<td>proposition $P \supset Q$</td>
<td>type $P \rightarrow Q$</td>
</tr>
<tr>
<td>proposition $P \land Q$</td>
<td>type $P \times Q$ (see §11.6)</td>
</tr>
<tr>
<td>proof of proposition $P$</td>
<td>term $t$ of type $P$</td>
</tr>
<tr>
<td>proposition $P$ is provable</td>
<td>type $P$ is inhabited (by some term)</td>
</tr>
</tbody>
</table>
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:

$$
erase(x) = x$$
$$
erase(\lambda x : T_1 . t_2) = \lambda x . erase(t_2)$$
$$
erase(t_1 \ t_2) = erase(t_1) \ erase(t_2)$$

**THEOREM:**

1. If $t \rightarrow t'$ under the typed evaluation relation, then $erase(t) \rightarrow erase(t')$.

2. If $erase(t) \rightarrow m'$ under the typed evaluation relation, then there is a simply typed term $t'$ such that $t \rightarrow t'$ and $erase(t') = m'$.
Curry-Style vs. Church-Style

• Curry Style
  – Syntax $\rightarrow$ Semantics $\rightarrow$ Typing
  – Semantics is defined on untyped terms
  – Often used for implicit typed languages

• Church Style
  – Syntax $\rightarrow$ Typing $\rightarrow$ Semantics
  – Semantics is defined only on well-typed terms
  – Often used for explicit typed languages
Homework

• Read Chapter 9.
• Do Exercise 9.3.9.

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

Proof: Exercise [Recommended, ★★★]. The structure is very similar to the proof of the type preservation theorem for arithmetic expressions (8.3.3), except for the use of the substitution lemma. □