

CS738: Advanced Compiler Optimizations

Simply Typed Lambda Calculus

Amey Karkare

karkare@cse.iitk.ac.in

<http://www.cse.iitk.ac.in/~karkare/cs738>

Department of CSE, IIT Kanpur



Reference Book

Types and Programming Languages by Benjamin C. Pierce

Simple Types over Bool

T := – Types

Simple Types over Bool

T := – Types
 Bool – Boolean Type

Simple Types over Bool

$$\begin{array}{ll} T & ::= \text{ -- Types} \\ \text{Bool} & \text{ -- Boolean Type} \\ T \rightarrow T & \text{ -- Function Type} \end{array}$$

Simple Types over Bool

$$\begin{array}{ll} T & ::= \text{ -- Types} \\ \text{Bool} & \text{ -- Boolean Type} \\ T \rightarrow T & \text{ -- Function Type} \end{array}$$

type constructor \rightarrow is right-associative, i.e., $T_1 \rightarrow T_2 \rightarrow T_3$ stands for $T_1 \rightarrow (T_2 \rightarrow T_3)$

Examples

For each of the type below, write a function (in your favourite programming language) that has the required type:

► $\text{Bool} \rightarrow \text{Bool}$

Examples

For each of the type below, write a function (in your favourite programming language) that has the required type:

- ▶ $\text{Bool} \rightarrow \text{Bool}$
- ▶ $\text{Bool} \rightarrow \text{Bool} \rightarrow \text{Bool}$

Examples

For each of the type below, write a function (in your favourite programming language) that has the required type:

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

Examples

For each of the type below, write a function (in your favourite programming language) that has the required type:

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

Examples

For each of the type below, write a function (in your favourite programming language) that has the required type:

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

Examples

For each of the type below, write a function (in your favourite programming language) that has the required type:

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

Examples

For each of the type below, write a function (in your favourite programming language) that has the required type:

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

The Abstract Syntax

Simply Typed λ -terms with conditions and Booleans

$t ::= x$

– *Variable*

The Abstract Syntax

Simply Typed λ -terms with conditions and Booleans

t	$:=$	x	– <i>Variable</i>
	$ $	$\lambda x : T. t$	– <i>Abstraction</i>

The Abstract Syntax

Simply Typed λ -terms with conditions and Booleans

t	$:=$	x	– <i>Variable</i>
		$\lambda x : T. t$	– <i>Abstraction</i>
		$t t$	– <i>Application</i>

The Abstract Syntax

Simply Typed λ -terms with conditions and Booleans

t	$:=$	x	– <i>Variable</i>
		$\lambda x : T. t$	– <i>Abstraction</i>
		$t t$	– <i>Application</i>
		<code>true</code>	– <i>constant true</i>

The Abstract Syntax

Simply Typed λ -terms with conditions and Booleans

t	$:=$	x	– <i>Variable</i>
		$\lambda x : T. t$	– <i>Abstraction</i>
		$t t$	– <i>Application</i>
		<code>true</code>	– <i>constant true</i>
		<code>false</code>	– <i>constant false</i>

The Abstract Syntax

Simply Typed λ -terms with conditions and Booleans

t	$:=$	x	– <i>Variable</i>
		$\lambda x : T. t$	– <i>Abstraction</i>
		$t t$	– <i>Application</i>
		<code>true</code>	– <i>constant true</i>
		<code>false</code>	– <i>constant false</i>
		<code>if t then t else t</code>	– <i>conditional</i>

Recap: The Set of Values

v $:=$ $-$ *values*
 $\lambda x : T. t$ $-$ *Abstraction Value*

Recap: The Set of Values

v := – *values*
 $\lambda x : T. t$ – *Abstraction Value*
 | $t \text{ true}$ – *value true*

Recap: The Set of Values

v := – *values*
 $\lambda x : T. t$ – *Abstraction Value*
 | `true` – *value true*
 | `false` – *value false*

Evaluation

$$\frac{t_1 \rightarrow t'_1}{t_1 \ t_2 \rightarrow t'_1 \ t_2}$$

(E-APP1)

Evaluation

$$\frac{t_1 \rightarrow t'_1}{t_1 \ t_2 \rightarrow t'_1 \ t_2} \quad (\text{E-APP1})$$

$$\frac{t_2 \rightarrow t'_2}{\nu \ t_2 \rightarrow \nu \ t'_2} \quad (\text{E-APP2})$$

Evaluation

$$\frac{t_1 \rightarrow t'_1}{t_1 \ t_2 \rightarrow t'_1 \ t_2} \quad (\text{E-APP1})$$

$$\frac{t_2 \rightarrow t'_2}{v \ t_2 \rightarrow v \ t'_2} \quad (\text{E-APP2})$$

$$(\lambda x : T_1. t_1) v_2 \rightarrow [x \mapsto v_2] t_1 \quad (\text{E-APPABS})$$

The Typing Relation

- ▶ A *Typing Context* or *Type Environment*, Γ , is a sequence of variables with their types

The Typing Relation

- ▶ A *Typing Context* or *Type Environment*, Γ , is a sequence of variables with their types
- ▶ $\Gamma, x : T$ denotes extending Γ with a new variable x having type T

The Typing Relation

- ▶ A *Typing Context* or *Type Environment*, Γ , is a sequence of variables with their types
- ▶ $\Gamma, x : T$ denotes extending Γ with a new variable x having type T
 - ▶ The name x is assumed to be distinct from any existing names in Γ

The Typing Relation

$$\frac{\Gamma, x : T_1 \vdash t_2 : T_2}{\Gamma \vdash \lambda x : T_1. t_2 : T_1 \rightarrow T_2} \quad (\text{T-ABS})$$

The Typing Relation

$$\frac{\Gamma, x : T_1 \vdash t_2 : T_2}{\Gamma \vdash \lambda x : T_1. t_2 : T_1 \rightarrow T_2} \quad (\text{T-ABS})$$

$$\frac{x : T \in \Gamma}{\Gamma \vdash x : T} \quad (\text{T-VAR})$$

The Typing Relation

$$\frac{\Gamma, x : T_1 \vdash t_2 : T_2}{\Gamma \vdash \lambda x : T_1. t_2 : T_1 \rightarrow T_2} \quad (\text{T-ABS})$$

$$\frac{x : T \in \Gamma}{\Gamma \vdash x : T} \quad (\text{T-VAR})$$

$$\frac{\Gamma \vdash t_1 : T_1 \rightarrow T_2 \quad \Gamma \vdash t_2 : T_1}{\Gamma \vdash t_1 t_2 : T_2} \quad (\text{T-APP})$$

Inversion of the Typing Relation

- ▶ If $\Gamma \vdash x : R$, then $x : R \in \Gamma$.

Inversion of the Typing Relation

- ▶ If $\Gamma \vdash x : R$, then $x : R \in \Gamma$.
- ▶ If $\Gamma \vdash \lambda x : T_1. t_2 : R$, then $R = T_1 \rightarrow R_2$ for some R_2 with $\Gamma, x : T_1 \vdash t_2 : R_2$.

Inversion of the Typing Relation

- ▶ If $\Gamma \vdash x : R$, then $x : R \in \Gamma$.
- ▶ If $\Gamma \vdash \lambda x : T_1. t_2 : R$, then $R = T_1 \rightarrow R_2$ for some R_2 with $\Gamma, x : T_1 \vdash t_2 : R_2$.
- ▶ If $\Gamma \vdash t_1 t_2 : R$, then $\exists T_1$ s.t. $\Gamma \vdash t_1 : T_1 \rightarrow R$ and $\Gamma \vdash t_2 : T_1$.

Inversion of the Typing Relation

- ▶ If $\Gamma \vdash x : R$, then $x : R \in \Gamma$.
- ▶ If $\Gamma \vdash \lambda x : T_1. t_2 : R$, then $R = T_1 \rightarrow R_2$ for some R_2 with $\Gamma, x : T_1 \vdash t_2 : R_2$.
- ▶ If $\Gamma \vdash t_1 t_2 : R$, then $\exists T_1$ s.t. $\Gamma \vdash t_1 : T_1 \rightarrow R$ and $\Gamma \vdash t_2 : T_1$.
- ▶ If $\Gamma \vdash \text{true} : R$, then $R = \text{Bool}$.

Inversion of the Typing Relation

- ▶ If $\Gamma \vdash x : R$, then $x : R \in \Gamma$.
- ▶ If $\Gamma \vdash \lambda x : T_1. t_2 : R$, then $R = T_1 \rightarrow R_2$ for some R_2 with $\Gamma, x : T_1 \vdash t_2 : R_2$.
- ▶ If $\Gamma \vdash t_1 t_2 : R$, then $\exists T_1$ s.t. $\Gamma \vdash t_1 : T_1 \rightarrow R$ and $\Gamma \vdash t_2 : T_1$.
- ▶ If $\Gamma \vdash \text{true} : R$, then $R = \text{Bool}$.
- ▶ If $\Gamma \vdash \text{false} : R$, then $R = \text{Bool}$.

Inversion of the Typing Relation

- ▶ If $\Gamma \vdash x : R$, then $x : R \in \Gamma$.
- ▶ If $\Gamma \vdash \lambda x : T_1. t_2 : R$, then $R = T_1 \rightarrow R_2$ for some R_2 with $\Gamma, x : T_1 \vdash t_2 : R_2$.
- ▶ If $\Gamma \vdash t_1 t_2 : R$, then $\exists T_1$ s.t. $\Gamma \vdash t_1 : T_1 \rightarrow R$ and $\Gamma \vdash t_2 : T_1$.
- ▶ If $\Gamma \vdash \text{true} : R$, then $R = \text{Bool}$.
- ▶ If $\Gamma \vdash \text{false} : R$, then $R = \text{Bool}$.
- ▶ If $\Gamma \vdash \text{if } t_1 \text{ then } t_2 \text{ else } t_3 : R$, then

Inversion of the Typing Relation

- ▶ If $\Gamma \vdash x : R$, then $x : R \in \Gamma$.
- ▶ If $\Gamma \vdash \lambda x : T_1. t_2 : R$, then $R = T_1 \rightarrow R_2$ for some R_2 with $\Gamma, x : T_1 \vdash t_2 : R_2$.
- ▶ If $\Gamma \vdash t_1 t_2 : R$, then $\exists T_1$ s.t. $\Gamma \vdash t_1 : T_1 \rightarrow R$ and $\Gamma \vdash t_2 : T_1$.
- ▶ If $\Gamma \vdash \text{true} : R$, then $R = \text{Bool}$.
- ▶ If $\Gamma \vdash \text{false} : R$, then $R = \text{Bool}$.
- ▶ If $\Gamma \vdash \text{if } t_1 \text{ then } t_2 \text{ else } t_3 : R$, then
 - ▶ $\Gamma \vdash t_1 : \text{Bool}$

Inversion of the Typing Relation

- ▶ If $\Gamma \vdash x : R$, then $x : R \in \Gamma$.
- ▶ If $\Gamma \vdash \lambda x : T_1. t_2 : R$, then $R = T_1 \rightarrow R_2$ for some R_2 with $\Gamma, x : T_1 \vdash t_2 : R_2$.
- ▶ If $\Gamma \vdash t_1 t_2 : R$, then $\exists T_1$ s.t. $\Gamma \vdash t_1 : T_1 \rightarrow R$ and $\Gamma \vdash t_2 : T_1$.
- ▶ If $\Gamma \vdash \text{true} : R$, then $R = \text{Bool}$.
- ▶ If $\Gamma \vdash \text{false} : R$, then $R = \text{Bool}$.
- ▶ If $\Gamma \vdash \text{if } t_1 \text{ then } t_2 \text{ else } t_3 : R$, then
 - ▶ $\Gamma \vdash t_1 : \text{Bool}$
 - ▶ $\Gamma \vdash t_2 : R$

Inversion of the Typing Relation

- ▶ If $\Gamma \vdash x : R$, then $x : R \in \Gamma$.
- ▶ If $\Gamma \vdash \lambda x : T_1. t_2 : R$, then $R = T_1 \rightarrow R_2$ for some R_2 with $\Gamma, x : T_1 \vdash t_2 : R_2$.
- ▶ If $\Gamma \vdash t_1 t_2 : R$, then $\exists T_1$ s.t. $\Gamma \vdash t_1 : T_1 \rightarrow R$ and $\Gamma \vdash t_2 : T_1$.
- ▶ If $\Gamma \vdash \text{true} : R$, then $R = \text{Bool}$.
- ▶ If $\Gamma \vdash \text{false} : R$, then $R = \text{Bool}$.
- ▶ If $\Gamma \vdash \text{if } t_1 \text{ then } t_2 \text{ else } t_3 : R$, then
 - ▶ $\Gamma \vdash t_1 : \text{Bool}$
 - ▶ $\Gamma \vdash t_2 : R$
 - ▶ $\Gamma \vdash t_3 : R$

Exercises

- ▶ For each of the term t below, find context Γ and type T such that

$$\Gamma \vdash t : T$$

Exercises

- ▶ For each of the term t below, find context Γ and type T such that

$$\Gamma \vdash t : T$$

- ▶ t is $\lambda x. x$

Exercises

- ▶ For each of the term t below, find context Γ and type T such that

$$\Gamma \vdash t : T$$

- ▶ t is $\lambda x. x$
- ▶ t is $((x\ z)\ (y\ z))$

Exercises

- ▶ For each of the term t below, find context Γ and type T such that

$$\Gamma \vdash t : T$$

- ▶ t is $\lambda x. x$
- ▶ t is $((x\ z)\ (y\ z))$
- ▶ t is $\lambda y. x$

Exercises

- ▶ For each of the term t below, find context Γ and type T such that

$$\Gamma \vdash t : T$$

- ▶ t is $\lambda x. x$
- ▶ t is $((x\ z)\ (y\ z))$
- ▶ t is $\lambda y. x$
- ▶ t is $x\ x$

Uniqueness of Types

- ▶ In a given type context Γ , A term t , such that the free variables of t are in Γ , has at most one type.

Uniqueness of Types

- ▶ In a given type context Γ , A term t , such that the free variables of t are in Γ , has at most one type.
- ▶ If t is typeable, then its type is unique.

Uniqueness of Types

- ▶ In a given type context Γ , A term t , such that the free variables of t are in Γ , has at most one type.
- ▶ If t is typeable, then its type is unique.
- ▶ Moreover, there is just one derivation of this typing built from the inference rules.

Some Properties

- ▶ **Permutation:** If $\Gamma \vdash t : T$ and Δ is a permutation of Γ , then $\Delta \vdash t : T$.

Some Properties

- ▶ **Permutation:** If $\Gamma \vdash t : T$ and Δ is a permutation of Γ , then $\Delta \vdash t : T$.
 - ▶ The derivation with Δ has the same depth as the derivation with Γ .

Some Properties

- ▶ **Permutation:** If $\Gamma \vdash t : T$ and Δ is a permutation of Γ , then $\Delta \vdash t : T$.
 - ▶ The derivation with Δ has the same depth as the derivation with Γ .
- ▶ **Weakening:** If $\Gamma \vdash t : T$ and $x \notin \text{domain}(\Gamma)$, then $\Gamma, x : S \vdash t : T$.

Some Properties

- ▶ **Permutation:** If $\Gamma \vdash t : T$ and Δ is a permutation of Γ , then $\Delta \vdash t : T$.
 - ▶ The derivation with Δ has the same depth as the derivation with Γ .
- ▶ **Weakening:** If $\Gamma \vdash t : T$ and $x \notin \text{domain}(\Gamma)$, then $\Gamma, x : S \vdash t : T$.
 - ▶ The derivation with $\Gamma, x : S$ has the same depth as the derivation with Γ .

Progress

- ▶ **Progress:** A well-typed term is not stuck.

Progress

- ▶ **Progress:** A well-typed term is not stuck.
 - ▶ If $\vdash t : T$, then t is either a value or there exists some t' such that $t \rightarrow t'$.

Preservation

- **Preservation of Types under Substitution:** If $\Gamma, x : S \vdash t : T$ and $\Gamma \vdash s : S$, then $\Gamma \vdash [x \mapsto s]t : T$.

Preservation

- ▶ **Preservation of Types under Substitution:** If $\Gamma, x : S \vdash t : T$ and $\Gamma \vdash s : S$, then $\Gamma \vdash [x \mapsto s]t : T$.
- ▶ **Preservation:** If a well-typed term takes a step of evaluation, then the resulting term is also well-typed.

Preservation

- ▶ **Preservation of Types under Substitution:** If $\Gamma, x : S \vdash t : T$ and $\Gamma \vdash s : S$, then $\Gamma \vdash [x \mapsto s]t : T$.
- ▶ **Preservation:** If a well-typed term takes a step of evaluation, then the resulting term is also well-typed.
 - ▶ If $\Gamma \vdash t : T$ and $t \rightarrow t'$, then $\Gamma \vdash t' : T$.