Programming Language Theory

Higher-Order Functions

陳亮廷 Chen, Liang-Ting
2020 邏輯、語言與計算暑期研習營
Formosan Summer School on Logic, Language, and Computation

Institute of Information Science, Academia Sinica

Simply Typed λ-Calculus: Statics

Typing judgement

A typing judgement is of the form

$$\Gamma \vdash M : \sigma$$

saying the *term M is of type σ under the context* Γ where

context Γ	free variables $x : \tau$ available in M
term M	possibly with free variables in Γ,
type σ	for M

$$x_1 : \tau_1, x_2 : \tau_2 \vdash x_1 : \tau_1$$

'*Under the context consisting of variables $x_1 : \tau_1, x_2 : \tau_2$, the term x_1 is of type τ_1.*'

Context

Definition 1

A *typing context* Γ is a sequence

$$\Gamma \equiv x_1 : \sigma_1,\ x_2 : \sigma_2,\ \ldots,\ x_n : \sigma_n$$

of *distinct variables* x_i of type σ_i.

Higher-order function type

Definition 2

Define the judgement $\tau : \mathtt{Type}$ by

$$\frac{\sigma \text{ is a type variable}}{\sigma : \mathtt{Type}}\ (\text{tvar}) \qquad \frac{\sigma : \mathtt{Type} \qquad \tau : \mathtt{Type}}{\sigma \to \tau : \mathtt{Type}}\ (\text{fun})$$

where $\sigma \to \tau$ represents a function type from σ to τ.

Also $\sigma_1 \to \tau_1 = \sigma_2 \to \tau_2$ if and only if $\sigma_1 = \sigma_2$ and $\tau_1 = \tau_2$.

Convention

$$\sigma_1 \to \sigma_2 \to \dots \sigma_n \quad := \quad \sigma_1 \to (\sigma_2 \to (\cdots \to (\sigma_{n-1} \to \sigma_n) \dots))$$

The function type is higher-order, because

1. functions can be arguments of another function;
2. functions can be the result of a computation.

Example 3

$(\sigma_1 \to \sigma_2) \to \tau$ a function type whose argument is of type $\sigma_1 \to \sigma_2$;

$\sigma_1 \to (\sigma_2 \to \tau)$ a function whose return type is $\sigma_2 \to \tau$.

For a term M, how to construct a *typing judgement*

$$\Gamma \vdash M : \sigma \to \tau$$

Typing rule – Curry-style typing system

A *typing rule* is an inference rule with its conclusion a typing judgement.

Definition 4 (Implicit typing)

$$\frac{x : \sigma \text{ occurs in } \Gamma}{\Gamma \vdash x : \sigma}\ (\text{var})$$

$$\frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash \lambda x.\ M : \sigma \to \tau}\ (\text{abs})$$

$$\frac{\Gamma \vdash M : \sigma \to \tau \qquad \Gamma \vdash N : \sigma}{\Gamma \vdash M\ N : \tau}\ (\text{app})$$

It is also known as the implicit typing system since the typing information is an add-on to the term.

Typing derivation

The judgement $\vdash \lambda x.\, x : \sigma \to \sigma$, for all $\sigma \in \mathbb{T}$ has a derivation

$$\dfrac{\dfrac{}{x : \sigma \vdash x : \sigma}\ (\text{var})}{\vdash \lambda x.\, x : (\sigma \to \sigma)}\ (\text{abs})$$

The judgement $\vdash \lambda x\, y.\, x : \sigma \to \tau \to \sigma$ has a derivation

$$\dfrac{\dfrac{\dfrac{}{x : \sigma, y : \tau \vdash x : \sigma}\ (\text{var})}{x : \sigma \vdash \lambda y.\, x : \tau \to \sigma}\ (\text{abs})}{\vdash \lambda x\, y.\, x : \sigma \to \tau \to \sigma}\ (\text{abs})$$

Not every λ-term has a type:

$$\lambda x.\, x\, x$$

there is no τ satisfying $\vdash \lambda x.\, x\, x : \tau$.

Syntax-directedness

A typing system is *syntax-directed* if it has *exactly* one typing rule for each term construct. Therefore,

Lemma 5 (Typing inversion)

Suppose

$$\Gamma \vdash M : \tau$$

is derivable. If

$M \equiv x$	*then* $x : \tau$ *occurs in* Γ.
$M \equiv \lambda x.\, M'$	*then* $\tau = \sigma \to \tau'$ *for some* σ *and* $\Gamma, x : \sigma \vdash M' : \tau'$.
$M \equiv L\ N$	*there is some* σ *such that* $\Gamma \vdash L : \sigma \to \tau$ *and* $\Gamma \vdash N : \sigma$.

Explicit typing: Typed terms

Definition 6 (Typed terms)

The formation $M : \texttt{Term}_{\lambda_\to}$ of typed terms is defined by

$$\frac{x \in V}{x : \texttt{Term}_{\lambda_\to}}$$

$$\frac{M : \texttt{Term}_{\lambda_\to} \qquad N : \texttt{Term}_{\lambda_\to}}{M\,N : \texttt{Term}_{\lambda_\to}}$$

$$\frac{M : \texttt{Term}_{\lambda_\to} \qquad x \in V \qquad \tau : \texttt{Type}}{\lambda x : \tau.\, M : \texttt{Term}_{\lambda_\to}}$$

Explicit typing: Typing rules

Definition 7 (Typing Rules)

Typing derivations on *typed terms* are defined by

$$\frac{x : \sigma \text{ occurs in } \Gamma}{\Gamma \vdash x : \sigma} \text{(var)}$$

$$\frac{\Gamma \vdash M : \sigma \to \tau \qquad \Gamma \vdash N : \sigma}{\Gamma \vdash M\ N : \tau} \text{(app)}$$

$$\frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash \lambda x : \sigma .\ M : \sigma \to \tau} \text{(abs)}$$

Unicity

Proposition 8

For every (typed) term M, context Γ, and types σ_i,

$$\Gamma \vdash M : \sigma_1 \quad \textit{and} \quad \Gamma \vdash M : \sigma_2 \implies \sigma_1 = \sigma_2$$

Proof sketch.

Use the inversion lemma and the structural induction on M.

For example, suppose that M is of the form

$$L\ M'$$

By inversion there are τ_i such that $\Gamma \vdash L : \tau_i \to \sigma_i$ and $\Gamma \vdash M' : \tau_i$. By induction hypothesis, $\tau_1 \to \sigma_1 = \tau_2 \to \sigma_2$, so $\sigma_1 = \sigma_2$. $\square$

Exercise

1. Derive the judgement

$$\vdash \lambda f\, g\, x.\, f\, x\, (g\, x) : (\sigma \to \tau \to \rho) \to (\sigma \to \tau) \to \sigma \to \rho$$

   for every $\sigma, \tau, \rho \in \mathbb{T}$.

2. Describe all possible types for Church numeral c_n.

Type erasure

An *erasing map* $|-|: \texttt{Term}_{\lambda_\to} \to \texttt{Term}_\lambda$ is defined by

$$|x| = x$$
$$|M\ N| = |M|\ |N|$$
$$|\lambda x : \sigma.\, M| = \lambda x.\, |M|$$

Example 9

1. $|\lambda(f : \sigma \to \tau)\,(x : \sigma).\, f\,x| = \lambda f\,x.\, f\,x$
2. $|(\lambda(x : \sigma)\,(y : \tau).y)\ z| = (\lambda x\,y.\, y)\ z$

$|-|$ is an translation from $\texttt{Term}_{\lambda_\to}$ to $\texttt{Term}_\lambda$. Does $|-|$ respect the behaviour of $\texttt{Term}_{\lambda_\to}$?

From typed terms to untyped and back

Proposition 10

Let M and N be typed λ-terms in $\texttt{Term}_{\lambda\rightarrow}$. Then,

$$\Gamma \vdash M : \sigma \textit{ implies } \Gamma \vdash |M| : \sigma$$

$$M \longrightarrow_{\beta *} N \textit{ implies } |M| \longrightarrow_{\beta *} |N|$$

Proposition 11

Let M and N be λ-terms in $\texttt{Term}_{\lambda}$. Then,

1. *If $\Gamma \vdash M : \sigma$, then there is $M' : \texttt{Term}_{\lambda\rightarrow}$ with*
   $|M'| = M \quad \textit{and} \quad \Gamma \vdash M' : \sigma$
2. *If $M \longrightarrow_{\beta *} N$ and $M = |M'|$ for some $M' : \texttt{Term}_{\lambda\rightarrow}$, then there exists N' with $|N'| = N$ and $M' \longrightarrow_{\beta *} N'$.*

Type inference

Can we answer the following questions

Typability Given a closed term M, is there a type σ such that $\vdash M : \sigma$?

Type checking Given Γ and σ, is $\Gamma \vdash M : \sigma$ derivable?

algorithmically?

Typability is reducible to type checking problem of

$$x_0 : \tau \vdash \mathsf{K}_1\, x_0\, M : \tau$$

Theorem 12

Type checking is decidable in simply typed λ-calculus.

Check *bidirectional type inference.*

Programming in Simply Typed λ-Calculus

Church encodings of natural numbers i

The type of natural numbers is of the form

$$\mathtt{nat}_\tau := (\tau \to \tau) \to \tau \to \tau$$

for every type $\tau \in \mathbb{T}$.

Church numerals

$$\mathsf{c}_n := \lambda f x. f^n x$$
$$\vdash \mathsf{c}_n : \mathtt{nat}_\tau$$

Church encodings of natural numbers ii

Successor

$$\begin{aligned} \mathtt{suc} &:= \lambda n\, f\, x\, .\, f\,(n\, f\, x) \\ \vdash \mathtt{suc} &: \mathtt{nat}_\tau \to \mathtt{nat}_\tau \end{aligned}$$

Addition

$$\begin{aligned} \mathtt{add} &:= \lambda n\, m\, f\, x.\, (m\, f)\,(n\, f\, x) \\ \vdash \mathtt{add} &: \mathtt{nat}_\tau \to \mathtt{nat}_\tau \to \mathtt{nat}_\tau \end{aligned}$$

Muliplication

$$\begin{aligned} \mathtt{mul} &:= \lambda n\, m\, f\, x.\, (m\,(n\, f))\, x \\ \vdash \mathtt{mul} &: \mathtt{nat}_\tau \to \mathtt{nat}_\tau \to \mathtt{nat}_\tau \end{aligned}$$

Conditional

$$
\begin{aligned}
\texttt{ifz} &:= \lambda n\, x\, y.\, n\ (\lambda z.\, x)\ y \\
\vdash \texttt{ifz} &: ?
\end{aligned}
$$

The type of `ifz` may not be as obvious as you may expect. Try to find one as general as possible and justify your guess.

Church encodings of boolean values

We can also define the type of Boolean values for each type variable as

$$\mathtt{bool}_\tau := \tau \to \tau \to \tau$$

Boolean values

$$\mathtt{true} := \lambda x y.\, x \quad \text{and} \quad \mathtt{false} := \lambda x y.\, y$$

Conditional

$$\mathtt{cond} := \lambda b\, x\, y.\, b\, x\, y$$
$$\vdash \mathtt{cond} : \mathtt{bool}_\tau \to \tau \to \tau \to \tau$$

Exercise

1. Define conjunction `and`, disjunction `or`, and negation `not` in simply typed lambda calculus.
2. Prove that `and`, `or`, and `not` are well-typed.

Properties of Simply Typed λ-Calculus

Type safety = Preservation + Progress

> *"Well-typed programs cannot 'go wrong.'"*
>
> —*(Milner, 1978)*

Preservation If $\Gamma \vdash M : \sigma$ is derivable and $M \longrightarrow_{\beta 1} N$, then $\Gamma \vdash N : \sigma$.

Progress If $\Gamma \vdash M : \sigma$ is derivable, then either M is in *normal form* or there is N with $M \longrightarrow_{\beta 1} N$.

Example 13

Recall that

1. $\mathsf{I} = \lambda x.\, x$
2. $\mathsf{K}_1 = \lambda x\, y.\, x$
3. $\Omega = (\lambda x.\, x\, x)\,(\lambda x.\, x\, x)$

and $\mathsf{K}_1\,\mathsf{I}\,\Omega \longrightarrow_{\beta *} \mathsf{I}$. However,

$$\vdash \mathsf{I} : \sigma \to \sigma \;\not\Longrightarrow\; \vdash \mathsf{K}_1\,\mathsf{I}\,\Omega : \sigma \to \sigma.$$

How to prove it?

Converse of Preservation ii

Lemma 14 (Typability of subterms)

Let M be a term with $\Gamma \vdash M : \tau$ derivable. Then, for every subterm M' of M there exists Γ' such that

$$\Gamma' \vdash M' : \sigma'.$$

Proof.

By induction on $\Gamma \vdash M : \sigma$. $\square$

Ω is not typable, so $\mathsf{K}_1\,\mathsf{I}\,\Omega$ is not typable.

A prelude to the preservation proof

Weakening If $\Gamma \vdash M : \tau$ and $x \notin \Gamma$, then $\Gamma, x : \sigma \vdash M : \tau$.

Substitution If $\Gamma, x : \tau \vdash M : \sigma$ and $\Gamma \vdash N : \tau$ then $\Gamma \vdash M[N/x] : \sigma$.

Corollary 15 (Variable renaming)

If $\Gamma, x : \tau \vdash M : \sigma$ and $y \notin \mathrm{dom}(\Gamma)$, then $\Gamma, y : \tau \vdash M[y/x] : \sigma$ where $\mathrm{dom}(\Gamma)$ denotes the set of variables which occur in Γ.

Proof.

y is not in Γ, so $\Gamma, y : \tau, x : \tau \vdash M$ by weakening and by definition $\Gamma, y : \tau \vdash y : \tau$. Thus, by substitution, we have

$$\Gamma, y : \tau \vdash M[x/y] : \sigma$$

□

Preservation Theorem

Theorem 16

If $\Gamma \vdash M : \sigma$ is derivable and $M \longrightarrow_{\beta 1} N$, then $\Gamma \vdash N : \sigma$.

Proof sketch.

By induction on both the derivation of $\Gamma \vdash M : \sigma$ and $M \longrightarrow_{\beta 1} N$.

The only non-trivial case is

$$\Gamma \vdash (\lambda x_1 : \tau.\, M_1)\ N : \sigma$$

with the induction hypothesis applied to

$$\Gamma, x_1 : \tau \vdash M_1 : \sigma \quad \text{and} \quad \Gamma \vdash N : \tau.$$

$\square$

Normal form

The notion of normal form can be characterised syntactically:

Definition 17

Define judgements $\texttt{Neutral}\ M$ and $\texttt{Normal}\ M$ mutually by

$$\frac{}{\texttt{Neutral}\ x} \qquad \frac{\texttt{Neutral}\ M}{\texttt{Normal}\ M}$$

$$\frac{\texttt{Neutral}\ M \qquad \texttt{Normal}\ N}{\texttt{Neutral}\ M\ N} \qquad \frac{\texttt{Normal}\ M}{\texttt{Normal}\ \lambda x.\, M}$$

Idea. N is in normal form iff

$$N \equiv \lambda x_1 \cdots x_n.\, x\ N_1 \cdots N_k$$

where N_i's are in normal form.

Soundness and completeness of the inductive characterisation

Lemma 18

Let M be a (typed or untyped) term.

Soundness *If* $\texttt{Normal}\ M$ *(resp.* $\texttt{Neutral}\ M$*) is derivable, then* M *is in normal form.*

Completeness *If* M *is in normal form, then* $\texttt{Normal}\ M$ *is derivable.*

Proof sketch.

Soundness By mutual induction on the derivation of $\texttt{Normal}\ M$ and $\texttt{Neutral}\ M$.

Completeness By induction on the formation of M.

□

Progress

Theorem 19

If $\Gamma \vdash M : \sigma$ is derivable, then `Normal` *M or there is N with $M \longrightarrow_{\beta 1} N$.*

Proof sketch.

By induction on the derivation of $\Gamma \vdash M : \sigma$. $\square$

Weak normalisation

Definition 20

M is *weakly normalising* denoted by $M \downarrow$ if

$$\frac{\mathtt{Normal}\ M}{M \downarrow} \qquad\qquad \frac{M \longrightarrow_{\beta 1} N \qquad N \downarrow}{M \downarrow}$$

That is, M is weakly normalising if there is a sequence

$$M \longrightarrow_{\beta 1} M_1 \longrightarrow_{\beta 1} M_2 \longrightarrow_{\beta 1} \ldots N \not\longrightarrow_{\beta 1}$$

Theorem 21 (Weak normalisation)

Every term M with $\Gamma \vdash M : \tau$ is weakly normalising.

Strong normalisation

Definition 22

M is *strongly normalising* denoted by $M \Downarrow$ if

$$\frac{\forall N.\,(M \longrightarrow_{\beta 1} N \implies N \Downarrow)}{M \Downarrow}$$

Intuitively, *strong normalisation* says every sequence

$$M \longrightarrow_{\beta 1} M_1 \longrightarrow_{\beta 1} M_2 \cdots$$

terminates.

Theorem 23

Every term M with $\Gamma \vdash M : \tau$ is strongly normalising.

Definability

A function $f \colon \mathbb{N}^k \to \mathbb{N}$ is called *$\lambda_{\to}$-definable* if there is a λ-term F of type $\mathtt{nat} \to \mathtt{nat} \to \ldots \mathtt{nat} \to \mathtt{nat}$ such that

$$F\, \mathsf{c}_{n_1} \ldots \mathsf{c}_{n_k} \longrightarrow_{\beta *} \mathsf{c}_{f(n_1,\ldots,n_k)}$$

for every sequence $(n_1, n_2, \ldots, n_k) \in \mathbb{N}^k$. Diagrammatically,

$$\begin{tikzcd}
(n_1, n_2, \ldots, n_k) \arrow[r, mapsto] \arrow[d, mapsto, "(\mathsf{c}_{-})^k"'] & f(n_1, n_2, \ldots, n_k) \arrow[d, mapsto, "\mathsf{c}_{-}"] \\
(\mathsf{c}_{n_1}, \mathsf{c}_{n_2}, \ldots, \mathsf{c}_{n_k}) \arrow[r, mapsto] & F\ \mathsf{c}_{n_1}\ \mathsf{c}_{n_2}\ \ldots\ \mathsf{c}_{n_k} = \mathsf{c}_{f(n_1,n_2,\ldots,n_k)}
\end{tikzcd}$$

The limit of $\lambda_\to$

Theorem 24

The $\lambda_\to$-definable functions are the class of functions of the form $f\colon \mathbb{N}^k \to \mathbb{N}$ closed under compositions which contains

- *the constant functions,*
- *projections,*
- *additions,*
- *multiplications,*
- *and the conditional*

$$\mathrm{ifz}(n_0, n_1, n_2) = \begin{cases} n_1 & \text{if } n_0 = 0 \\ n_2 & \text{otherwise.} \end{cases}$$

Proof of confluence: Takahashi's approach

Confluence: Parallel reduction

Consider untyped λ-calculus.

Let $M \Longrightarrow_\beta N$ denote the *parallel reduction* defined by

$$\frac{}{x \Longrightarrow_\beta x} \qquad \frac{M \Longrightarrow_\beta M' \qquad N \Longrightarrow_\beta N'}{M\ N \Longrightarrow_\beta M'\ N'}$$

$$\frac{M \Longrightarrow_\beta N}{\lambda x.\, M \Longrightarrow_\beta \lambda x.\, N} \qquad \frac{M \Longrightarrow_\beta M' \qquad N \Longrightarrow_\beta N'}{(\lambda x.\, M)\ N \Longrightarrow_\beta M'[N'/x]}$$

For example,

$$\underline{(\lambda x.\,(\lambda y.\,y)\ x)}\ \underline{((\lambda x.\,x)\ \texttt{false})} \Longrightarrow_\beta \texttt{false}$$

because $(\lambda y.\,y)\ x \Longrightarrow_\beta x$ and $(\lambda x.\,x)\ \texttt{false} \Longrightarrow_\beta \texttt{false}$.

Confluence: Properties of parallel reduction

Lemma 25

1. *$M \Longrightarrow_\beta M$ holds for any term M,*
2. *$M \longrightarrow_{\beta 1} N$ implies $M \Longrightarrow_\beta N$, and*
3. *$M \Longrightarrow_\beta N$ implies $M \longrightarrow_{\beta *} N$.*

Therefore, $M \Longrightarrow_\beta^* N$ is equivalent to $M \longrightarrow_{\beta *} N$.

Lemma 26 (Substitution respects parallel reduction)

$M \Longrightarrow_\beta M'$ and $N \Longrightarrow_\beta N'$ imply $M[N/x] \Longrightarrow_\beta M'[N'/x]$.

Proof sketch.

By induction on the derivation of $M \Longrightarrow_\beta M'$.

$\square$

Complete development

The *complete development* M^* of a λ-term M is defined by

$$
\begin{aligned}
x^* &= x \\
(\lambda x.\, M)^* &= \lambda x.\, M^* \\
((\lambda x.\, M)\ N)^* &= M^*[N^*/x] \\
(M\ N)^* &= M^*\ N^* && \text{if } M \not\equiv \lambda x.\, M'
\end{aligned}
$$

Theorem 27 (Triangle property)

If $M \Longrightarrow_\beta N$, then $N \Longrightarrow_\beta M^*$.

Proof sketch.

By induction on $M \Longrightarrow_\beta N$.

$\square$

Strip Lemma

Theorem 28

If $L \Longrightarrow_\beta^* M_1$ and $L \Longrightarrow_\beta M_2$, then there exists N satisfying that $M_1 \Longrightarrow_\beta N$ and $M_2 \Longrightarrow_\beta^* N$, i.e.

$$\begin{tikzcd}
 & L \arrow[dl, Rightarrow, "\beta"', "*"] \arrow[dr, Rightarrow] & \\
M_1 \arrow[dr, Rightarrow] & & M_2 \arrow[dl, Rightarrow, "\beta", "*"'] \\
 & N &
\end{tikzcd}$$

Proof sketch.

By induction on $L \Longrightarrow_\beta^* M_1$. $\square$

Theorem 29

If $L \Longrightarrow^{*}_{\beta} M_1$ and $L \Longrightarrow^{*}_{\beta} M_2$, then there exists N such that $M_1 \Longrightarrow^{*}_{\beta} N$ and $M_2 \Longrightarrow^{*}_{\beta} N$.

$$\begin{tikzcd}
 & L \arrow[dl, Rightarrow, "^{*}_{\beta}"'] \arrow[dr, Rightarrow, "^{*}_{\beta}"] & \\
M_1 \arrow[dr, Rightarrow, "^{*}_{\beta}"'] & & M_2 \arrow[dl, Rightarrow, "^{*}_{\beta}"] \\
 & N &
\end{tikzcd}$$

Corollary 30

The confluence of $\longrightarrow_{\beta *}$ holds.

Homework

1. (25%) Show the Preservation Theorem.
   Hint. Apply the Substitution Lemma if applicable.
2. (25%) Show the Progress Theorem.
3. (25%) Show that if `Normal` M (resp. `Neutral` M), then M is in normal form.
4. (25%) Show that if M is in normal form then `Normal` M.
   Hint. Try to analyse possible cases of the induction hypothesis.