Polymorphism and System F

Haitian Wang

6 May 2026

Outline

Generic Identity Function

Untyped λ-calculus

$$\lambda x.\, x$$

Generic Identity Function

Untyped λ-calculus

$$\lambda x.\, x$$

Simply Typed λ-calculus (STLC)

$$\lambda x : \mathit{Nat}.\, x \qquad \lambda x : \mathit{Bool}.\, x \qquad \cdots$$

(one identity function *per type*)

Generic Identity Function

Untyped λ-calculus

$$\lambda x.\, x$$

Simply Typed λ-calculus (STLC)

$$\lambda x : \mathit{Nat}.\, x \qquad \lambda x : \mathit{Bool}.\, x \qquad \cdots$$

(one identity function *per type*)

Polymorphism (System F)

$$\Lambda\alpha.\, \lambda x : \alpha.\, x$$

(a *single* identity function for all types)

Outline

Second-Order Propositional Logic

We work in (constructive) second-order propositional logic built with $\to$ and $\forall$ only.

Formulas (FORM)

FORM is generated by:

- propositional variables: $p, q, r, \cdots \in \text{FORM}$
- implication: if $\varphi, \psi \in \text{FORM}$ then $(\varphi \to \psi) \in \text{FORM}$
- second-order quantification: if p is a propositional variable and $\varphi \in \text{FORM}$, then $(\forall p.\, \varphi) \in \text{FORM}$

They serve as the polymorphic types for system F.

Natural Deduction ($\to$ and $\forall$)

Axiom

$$(\mathsf{Ax}) \qquad \Gamma, \varphi \vdash \varphi$$

Implication Rules

$$(\to I)\ \frac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \to \psi} \qquad (\to E)\ \frac{\Gamma \vdash \varphi \to \psi \qquad \Gamma \vdash \varphi}{\Gamma \vdash \psi}$$

Second-Order Quantifier Rules

$$(\forall I)\ \frac{\Gamma \vdash \varphi}{\Gamma \vdash \forall p\, \varphi}\ (p \notin FV(\Gamma)) \qquad (\forall E)\ \frac{\Gamma \vdash \forall p\, \varphi}{\Gamma \vdash \varphi[p := \theta]}$$

Derived Connectives

We can define the other logical connectives and the constant $\bot$ using $\to$ and $\forall$.

$$\bot :\equiv \forall p.\, p$$

$$\varphi \wedge \psi :\equiv \forall p.\, (\varphi \to \psi \to p) \to p$$

$$\varphi \vee \psi :\equiv \forall p.\, (\varphi \to p) \to (\psi \to p) \to p$$

$$\exists q.\, \varphi :\equiv \forall p.\, (\forall q.(\varphi \to p)) \to p$$

(Assume $p \notin FV(\varphi) \cup FV(\psi)$.)

Outline

System F: Raw Terms

Raw terms extend the simply typed λ-calculus with polymorphic abstraction and type applications.

Grammar

$$M ::= x \mid (M\, M) \mid (\lambda x : \tau.\, M) \mid (\Lambda p.\, M) \mid (M\, \tau)$$

Type inference rules

A **context** is a finite set of declarations $(x : \tau)$, for different variables.

We can now decorate the proof trees and get the following typing rules:

$$(\text{Var}) \qquad \Gamma, x : \tau \vdash x : \tau$$

$$(\mathit{Abs}) \ \frac{\Gamma, x : \tau \vdash M : \sigma}{\Gamma \vdash \lambda x : \tau.\, M : \tau \to \sigma} \qquad (\mathit{App}) \ \frac{\Gamma \vdash N : \tau \to \sigma \qquad \Gamma \vdash M : \tau}{\Gamma \vdash N\, M : \sigma}$$

$$(\mathit{Gen}) \ \frac{\Gamma \vdash M : \sigma}{\Gamma \vdash \Lambda\alpha.\, M : \forall\alpha.\, \sigma} \quad (\alpha \notin FV(\Gamma)) \qquad (\mathit{Inst}) \ \frac{\Gamma \vdash M : \forall\alpha.\, \sigma}{\Gamma \vdash M\,\tau : \sigma[\alpha := \tau]}$$

Well-typed lambda-terms are defined by these rules.

Curry-Howard Isomorphism

Theorem (Curry-Howard Isomorphism)

For the polymorphic λ-calculus (System F),

$$\Gamma \vdash M : \tau \quad \text{iff} \quad |\Gamma| \vdash \tau$$

has a proof in the $\{\forall, \rightarrow\}$-fragment of second-order intuitionistic propositional logic.

Theorem

It is undecidable whether a given formula $\varphi \in 2\Phi$ has a proof.

Corollary (Undecidability of Inhabitation)

The inhabitation problem for the polymorphic λ-calculus is undecidable:

Given a type τ, is there a closed term M such that $\vdash M : \tau$?

β-Reduction

There are two kinds of β-reduction rules in System F:

Object Reduction (term-level)

$$(\lambda x : \tau.\, M)\ N \ \rightarrow_\beta \ M[x := N]$$

Type Reduction (type-level)

$$(\Lambda \alpha.\, M)\ \tau \ \rightarrow_\beta \ M[\alpha := \tau]$$

Properties

This reduction relation is **Church–Rosser**, **type-preserving** and all typable terms are **strongly normalizing**.

Outline

A revisit of second order propositional logic

We can define the other logical connectives and the constant $\bot$ using $\to$ and $\forall$.

$$\bot :\equiv \forall p.\, p$$

$$\varphi \wedge \psi :\equiv \forall p.\, (\varphi \to \psi \to p) \to p$$

$$\varphi \vee \psi :\equiv \forall p.\, (\varphi \to p) \to (\psi \to p) \to p$$

$$\exists q.\, \varphi :\equiv \forall p.\, (\forall q.(\varphi \to p)) \to p$$

(Assume $p \notin FV(\varphi) \cup FV(\psi)$.)

Empty type

Definition (Absurdity)

$$\bot := \forall\alpha.\,\alpha$$

Elimination Rule ($\bot E$)

$$(\bot E) \qquad \frac{\Gamma \vdash M : \bot}{\Gamma \vdash M\,\tau : \tau}$$

(From contradiction we can derive any type.)

Product type

Definition (Conjunction)

Let $\alpha \notin FV(\tau, \sigma)$. Define

$$\tau \wedge \sigma \;:=\; \forall\alpha.\,((\tau \to \sigma \to \alpha) \to \alpha).$$

Pairs and Projections

$$\langle P, Q\rangle \;:=\; \Lambda\alpha.\,\lambda z^{\tau\to\sigma\to\alpha}.\; z\,P\,Q$$

$$\pi_1(M^{\tau\wedge\sigma}) \;:=\; M\,\tau\,(\lambda x^{\tau}.\,\lambda y^{\sigma}.\,x)$$

$$\pi_2(M^{\tau\wedge\sigma}) \;:=\; M\,\sigma\,(\lambda x^{\tau}.\,\lambda y^{\sigma}.\,y)$$

Typing Rules (Natural Deduction)

$$\frac{\Gamma \vdash M : \tau \qquad \Gamma \vdash N : \sigma}{\Gamma \vdash \langle M, N\rangle : \tau \wedge \sigma} \qquad \frac{\Gamma \vdash M : \tau \wedge \sigma}{\Gamma \vdash \pi_1(M) : \tau} \qquad \frac{\Gamma \vdash M : \tau \wedge \sigma}{\Gamma \vdash \pi_2(M) : \sigma}$$

Coproduct Type

Definition (Disjunction)

Let $\alpha \notin FV(\tau, \sigma)$. Define

$$\tau \vee \sigma \;:=\; \forall\alpha.\,((\tau \to \alpha) \to (\sigma \to \alpha) \to \alpha).$$

Injections and Case Eliminator

$$\mathrm{in}_1(M^\tau) \;:=\; \Lambda\alpha.\,\lambda u^{\tau\to\alpha}.\,\lambda v^{\sigma\to\alpha}.\; u\,M$$

$$\mathrm{in}_2(M^\sigma) \;:=\; \Lambda\alpha.\,\lambda u^{\tau\to\alpha}.\,\lambda v^{\sigma\to\alpha}.\; v\,M$$

$$\mathrm{case}(L^{\tau\vee\sigma};\, x^\tau.M^\rho;\, y^\sigma.N^\rho) \;:=\; L\,\rho\,(\lambda x^\tau.\,M)\,(\lambda y^\sigma.\,N)$$

Typing Rules (Natural Deduction)

$$\frac{\Gamma \vdash M : \tau}{\Gamma \vdash \mathrm{in}_1(M) : \tau \vee \sigma} \qquad \frac{\Gamma \vdash M : \sigma}{\Gamma \vdash \mathrm{in}_2(M) : \tau \vee \sigma}$$

$$\frac{\Gamma \vdash L : \tau \vee \sigma \qquad \Gamma, x : \tau \vdash M : \rho \qquad \Gamma, y : \sigma \vdash N : \rho}{\Gamma \vdash \mathrm{case}(L;\, x.M;\, y.N) : \rho}$$

Existential Types

Existential Quantifier

Assume $\beta \notin FV(\sigma)$. Define

$$\exists\alpha.\sigma \;:=\; \forall\beta.\,(\forall\alpha.(\sigma \to \beta)) \to \beta.$$

Packing / Unpacking Terms

$$\text{pack } M, \tau \text{ to } \exists\alpha.\sigma \;:=\; \Lambda\beta.\,\lambda x^{\forall\alpha.(\sigma\to\beta)}.\; x\,\tau\,M$$

$$\text{abstype } \alpha \text{ with } x : \sigma \text{ is } M \text{ in } N^{\rho} \;:=\; M\,\rho\,(\lambda\alpha.\,\lambda x^{\sigma}.\,N)$$

Typing Rules

$$(\exists I) \qquad \frac{\Gamma \vdash M : \sigma[\alpha := \tau]}{\Gamma \vdash \text{pack } M, \tau \text{ to } \exists\alpha.\sigma : \exists\alpha.\sigma}$$

$$(\exists E) \qquad \frac{\Gamma \vdash M : \exists\alpha.\sigma \qquad \Gamma, x : \sigma \vdash N : \rho}{\Gamma \vdash \text{abstype } \alpha \text{ with } x : \sigma \text{ is } M \text{ in } N : \rho} \quad (\alpha \notin FV(\Gamma, \rho))$$

Booleans in System F

Boolean Type

$$\mathit{Bool} \;:=\; \forall\alpha.\,(\alpha \to \alpha \to \alpha)$$

Boolean Constants

$$\mathit{true} \;:=\; \Lambda\alpha.\,\lambda x^{\alpha}.\,\lambda y^{\alpha}.\,x \qquad \mathit{false} \;:=\; \Lambda\alpha.\,\lambda x^{\alpha}.\,\lambda y^{\alpha}.\,y$$

Church Numerals in System F

Integer Type

$$\omega \;:=\; \forall\alpha.\,((\alpha \to \alpha) \to \alpha \to \alpha)$$

Church Numerals

$$c_n \;:=\; \Lambda\alpha.\,\lambda f^{\alpha\to\alpha}.\,\lambda x^{\alpha}.\;\underbrace{f(\cdots f}_{n \text{ times}}(x)\cdots)$$

Definable Functions in System F

Successor

$$\mathsf{succ} \;:=\; \lambda n^{\omega}.\, \Lambda\alpha.\, \lambda f^{\alpha\to\alpha}.\, \lambda x^{\alpha}.\; f\,(n\,\alpha\,f\,x)$$

$n \mapsto n^n$

$$\mathsf{Exp} \;:=\; \lambda n^{\omega}.\, \Lambda\alpha.\, n\,(\alpha\to\alpha)\,(n\,\alpha)$$

(This uses polymorphism in an essential way!)

Recursor in System F

Embedding System T

System T can be embedded into System F.

β-Equations for r_σ

$$r_\sigma\, M\, N\, (c_0) \;=_\beta\; M$$

$$r_\sigma\, M\, N\, (\mathsf{succ}(n)) \;=_\beta\; N\, (r_\sigma\, M\, N\, n)\, n$$

Proposition (Recursor)

For any type σ, define

$$r_\sigma : \sigma \to \big(\sigma \to \omega \to \sigma\big) \to \omega \to \sigma$$

by

$$\begin{aligned} r_\sigma \;:=\; \lambda y^\sigma.\lambda f^{\sigma\to\omega\to\sigma}.\lambda n^\omega.\; \pi_1\Big(& n\,(\sigma\wedge\omega) \\ & (\lambda v^{\sigma\wedge\omega}.\,\langle f(\pi_1(v))(\pi_2(v)),\; \mathsf{succ}(\pi_2(v))\rangle) \\ & \langle y, c_0\rangle\Big). \end{aligned}$$

More Data Structures: Lists, Trees, ...

Lists of Natural Numbers

A list is either empty (`nil`) or of the form $n :: \ell$.

Encoding in System F

Define the type of lists of naturals as

$$\mathsf{List}_\omega \;:=\; \forall p.\,((\omega \to p \to p) \to p \to p).$$

Constructors

$$\mathsf{nil} \;:=\; \Lambda p.\,\lambda f^{\omega \to p \to p}.\,\lambda x^p.\,x$$

$$n :: \ell \;:=\; \Lambda p.\,\lambda f^{\omega \to p \to p}.\,\lambda x^p.\,f\ c_n\ (\ell\ p\ f\ x)$$

Example

$$1 :: 2 :: 3 :: \mathsf{nil} \;=\; \Lambda p.\,\lambda f.\,\lambda x.\; f\ \bar{3}\big(f\ \bar{2}(f\ \bar{1}x)\big)$$

Outline

Theorems for Free

Idea (Wadler, 1989)

We can derive theorems for free from the type of a polymorphic function.

Example

Let r be a function of type

$$r : \forall\alpha.\, [\alpha] \to [\alpha].$$

Let f be a (total) function of type

$$f : A \to A'.$$

Then we have:

$$f^* \circ r_A \;=\; r_{A'} \circ f^*,$$

where $f^* : [A] \to [A']$ is the standard map f acting elementwise on lists.

Intuition

Since r is polymorphic, it cannot inspect elements of the list. It can only transform the *structure* of the list.

Naturality

The free theorem

$$f^* \circ r_A \;=\; r_{A'} \circ f^*$$

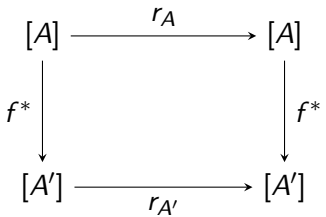
Naturality

The free theorem

$$f^* \circ r_A \;=\; r_{A'} \circ f^*$$

is exactly the **naturality condition** if we view r as a natural transformation

$$r : \mathsf{List} \Rightarrow \mathsf{List}.$$



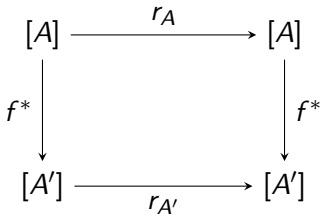
Naturality

The free theorem

$$f^* \circ r_A \;=\; r_{A'} \circ f^*$$

is exactly the **naturality condition** if we view r as a natural transformation

$$r : \mathsf{List} \Rightarrow \mathsf{List}.$$



Remark

Actually, parametric polymorphism imposes a stronger condition than mere naturality: the family of morphisms must arise from a *single uniform definition*.

Main References

[1] Sørensen, M. H., & Urzyczyn, P. (2006). Lectures on the Curry-Howard isomorphism (Vol. 149). Elsevier.

More References and Links on Free Theorems

- Wadler, P. (1989). Theorems for free!. In Proceedings of the fourth international conference on Functional programming languages and computer architecture (pp. 347-359).
- De Bruin, P. J. (1989). Naturalness of polymorphism. University of Groningen, Department of Mathematics and Computing Science.
- Bartosz Milewski's blogpost on Parametricity (2014). `https://bartoszmilewski.com/2014/09/22/parametricity-money-for-nothing-and-theorems-for-free/`
- Bartosz Milewski's blogpost on Natural Transformations (2014). `https://bartoszmilewski.com/2015/04/07/natural-transformations/`
- Math StackExchange discussions on Natural transformation = parametric polymorphic function in "structure categories"? `https://math.stackexchange.com/questions/3142222/`