

Polymorphism

- Outline
 - *System F (Second-order lambda calculus, Girard 1972)*
 - *Examples*
 - *Properties*
 - *Preservation*
 - *Progress*
 - *(Strong normalization)*
 - *Computability*



The polymorphic lambda calculus

e	$::=$	v	variables
		$\lambda x:t.e$	abstraction
		$e_1 e_2$	application
		$\Lambda X.e$	type abstraction
		$e[t]$	type application
t	$::=$	X	type variables
		$t_1 \rightarrow t_2$	function types
		$\forall X.t$	type quantification



Operational semantics for System F

Operational semantics for the simply-typed λ calculus.

$$(\lambda v:t.e_1) e_2 \rightarrow_{\beta} e_1[e_2/v]$$

Additional reductions for System F.

$$(\Lambda X.e) T \rightarrow_{\beta} e[T/X]$$



Identity program

Identity:

$$(\Lambda X. \lambda x: X. X) : (\forall X. X \rightarrow X)$$

Application of identity:

$$\begin{aligned} & (\Lambda X. \lambda x: X. X) [\mathbb{Z}] 1 \\ \rightarrow_{\beta} & (\lambda x: \mathbb{Z}. x) 1 \\ \rightarrow_{\beta} & 1 \end{aligned}$$



Eta-equivalence

Untyped λ calculus:

$$\lambda x.f\ x =_{\eta} f$$

Polymorphic λ calculus:

$$\text{eta} = (\Lambda X.\Lambda Y.\lambda f:(X \rightarrow Y).\lambda x:X.f\ x)$$



Eta application

eta = $(\Lambda X.\Lambda Y.\lambda f:(X \rightarrow Y).\lambda x:X.f\ x)$

eta $[\mathbb{Z}][\mathbb{Z}](\lambda i:\mathbb{Z}.s(i))\ 1$
 \rightarrow_{β} $(\lambda f:(\mathbb{Z} \rightarrow \mathbb{Z}).\lambda x:\mathbb{Z}.f\ x)\ (\lambda i:\mathbb{Z}.s(i))\ 1$
 \rightarrow_{β} $(\lambda x:\mathbb{Z}.(\lambda i:\mathbb{Z}.s(i))\ x)\ 1$
 \rightarrow_{β} $(\lambda x:\mathbb{Z}.s(x))\ 1$
 \rightarrow_{β} $s(1)$
 \rightarrow_{β} 2



Self-application

The canonical non-terminating computation was not expressible in the simply-typed λ calculus

$$\text{'} \equiv (\lambda x.x x) (\lambda x.x x)$$

Neither was the self-application fragment $\lambda x.x x$

System-F:

$$\begin{aligned} \text{self} &\equiv \lambda x: (\forall X.X \rightarrow X).x [\forall X.X \rightarrow X] x \\ \text{self} &: (\forall X.X \rightarrow X) \rightarrow (\forall X.X \rightarrow X) \end{aligned}$$



An example reduction sequence

self $\equiv \lambda x: (\forall X. X \rightarrow X). x [\forall X. X \rightarrow X] x$
self : $(\forall X. X \rightarrow X) \rightarrow (\forall X. X \rightarrow X)$

self $(\Lambda Y. \lambda y: Y. y)$
 \rightarrow_{β} $(\Lambda Y. \lambda y: Y. y) [\forall X. X \rightarrow X] (\Lambda Y. \lambda y: Y. y)$
 \rightarrow_{β} $(\lambda y: (\forall X. X \rightarrow X). y) (\Lambda Y. \lambda y: Y. y)$
 \rightarrow_{β} $(\Lambda Y. \lambda y: Y. y)$



Non-termination still fails

self \equiv self self
 $= (\lambda x: (\forall X. X \rightarrow X). x[\forall X. X \rightarrow X] x)$ self

self : $(\forall X. X \rightarrow X) \rightarrow (\forall X. X \rightarrow X)$



Typing judgments

A type judgment is a sequent $\Gamma \vdash e : t$

- e is a program
- t is a type
- Γ is a type assignment list:

Γ	$::=$	$\{\}$	empty context
		$\Gamma, x:t$	term variable
		Γ, X	type variable



Sequents

The order in Γ is significant:

$$X, y : X \rightarrow X, x : X \vdash (y \ x) : X$$

Axiom rule:

$$\frac{}{\Gamma_1, x : t, \Gamma_2 \vdash x : t} \text{ axiom}$$



Typing rules for simple fragment

Typing rules for the simply-typed λ calculus.

$$\frac{}{\Gamma, x:t \vdash x : t} \text{ var}$$

$$\frac{\Gamma \vdash e_1 : (s \rightarrow t) \quad \Gamma \vdash e_2 : s}{\Gamma \vdash (e_1 e_2) : t} \text{ app}$$

$$\frac{\Gamma, x:s \vdash t}{\Gamma \vdash (\lambda x:s.e) : (s \rightarrow t)} \text{ abs}$$



Typing rules for System F

Additional rules for System F.

$$\frac{\Gamma, X \vdash e : T}{\Gamma \vdash (\Lambda X. e) : (\forall X. T)} \quad \text{all intro}$$

$$\frac{\Gamma \vdash e : \forall X. T_2}{\Gamma \vdash e[T_1] : T_2[T_1/X]} \quad \text{all elim}$$



Typing of identity

$$\frac{\frac{\frac{\overline{X, x: X \vdash x : X} \quad 3}{X \vdash (\lambda x: X. x) : (X \rightarrow X)} \quad 2}{\vdash (\Lambda X. \lambda x: X. x) : (\forall X. X \rightarrow X)} \quad 1$$



Typing of eta

$$\frac{\frac{\frac{X, Y, f: X \rightarrow Y, x: X \vdash f : (X \rightarrow Y) \quad X, Y, f: X \rightarrow Y, x: X \vdash x : X}{X, Y, f: X \rightarrow Y, x: X \vdash (f \ x): Y} \quad 3}{X, Y \vdash (\lambda f: (X \rightarrow Y). \lambda x: X. f \ x) : ((X \rightarrow Y) \rightarrow (X \rightarrow Y))} \quad 2}{\vdash (\Lambda X. \Lambda Y. \lambda f: (X \rightarrow Y). \lambda x: X. f \ x) : (\forall X. \forall Y. (X \rightarrow Y) \rightarrow X \rightarrow Y)} \quad 1$$



Typing of self-application

$$\frac{\frac{\frac{x: \forall X. X \rightarrow X \vdash x: \forall X. X \rightarrow X}{x: \forall X. X \rightarrow X \vdash x[\forall X. X \rightarrow X]: (\forall X. X \rightarrow X) \rightarrow (\forall X. X \rightarrow X)}{x: \forall X. X \rightarrow X \vdash x[\forall X. X \rightarrow X]x: \forall X. X \rightarrow X} \quad 3}{\vdash (\lambda x: (\forall X. X \rightarrow X). x[\forall X. X \rightarrow X]x): (\forall X. X \rightarrow X) \rightarrow (\forall X. X \rightarrow X)} \quad 1} \quad 2$$



Type safety

- Preservation (types do not change during computation)
- Progress (if an expression is well-typed, then it is either a value, or it can be reduced one step more)



Term Substitution (var part)

If $\Gamma_1, x:s, \Gamma_2 \vdash e_1 \in t$
and $\Gamma_1, \Gamma_2 \vdash e_2 \in s$
then $\Gamma_1, \Gamma_2 \vdash e_1[e_2/x] \in t$.

var1 Suppose $e_1 = x$, then $s = t$

- by assumption, $\Gamma_1, x:t, \Gamma_2 \vdash e_1 \in t$
- by assumption, $\Gamma_1, \Gamma_2 \vdash e_2 \in t$
- also, $e_1[e_2/x] = e_2$,
- so $\Gamma_1, \Gamma_2 \vdash e_1[e_2/x] \in t$



Term substitution (var, app)

var2 Suppose $e_1 = y$ where $y \neq x$

- by assumption, $\Gamma_1, x:t, \Gamma_2 \vdash e_1 \in t$
- since $y \neq x$, $e_1[e_2/x] = e_1$
- so $\Gamma_1, \Gamma_2 \vdash e_1[e_2/x] \in t$

app Suppose $e_1 = e_3 e_4$, the result follows directly by induction.



Term substitution (*abs*)

abs Suppose $e_1 = \lambda y:S.e_3$, and $t = S \rightarrow T$

- then $t = S \rightarrow T$, and the last step of the proof was

$$\frac{\overline{\Gamma_1, x:s, \Gamma_2, y:S \vdash e_3 : T} \quad \dots \quad 2}{\Gamma_1, x:s, \Gamma_2 \vdash (\lambda y:S.e_3) : (S \rightarrow T)} \quad 1$$

- by induction $\Gamma_1, \Gamma_2, y:S \vdash e_3[e_2/x] : T$
- so $\Gamma_1, \Gamma_2 \vdash (\lambda y:S.e_3[e_2/x]) : (S \rightarrow T)$
- rewriting $\Gamma_1, \Gamma_2 \vdash e_1[e_2/x] \in t$



Type substitution lemma, var, app

If $\Gamma_1, X, \Gamma_2 \vdash e : t_1$,
then for any type t_2 ,
 $\Gamma_1, \Gamma_2[t_2/X] \vdash e[t_2/X] : t_1[t_2/X]$

var if e is a var, substitution does nothing.

app if $e = e_3 e_4$, the result follows by induction.



Type substitution lemma, abs

term abs: suppose $e = \lambda y:S.e_3$ and $t_1 \equiv S \rightarrow T$

- the last rule to be used was

$$\frac{\Gamma_1, X, \Gamma_2, y:S \vdash e_3 : T}{\Gamma_1, X, \Gamma_2 \vdash (\lambda y:S.e_3) : (S \rightarrow T)} \text{ abs}$$

- by induction $\Gamma_1, \Gamma_2[t_2/X], y:S[t_2/X] \vdash e_3[t_2/X] : T[t_2/X]$
- so $\Gamma_1, \Gamma_2[t_2/X] \vdash (\lambda y:S[t_2/X].e_3[t_2/X]) : (S[t_2/X] \rightarrow T[t_2/X])$
- rewriting $\Gamma_1, \Gamma_2[t_2/X] \vdash e[t_2/X] : t_1[t_2/X]$



Type substitution lemma, type abs

type abs: suppose $e = \Lambda Y.e_3$ and $t_1 \equiv \forall Y.T$

- the last rule to be used was

$$\frac{\Gamma_1, X, \Gamma_2, Y \vdash e_3 : T}{\Gamma_1, X, \Gamma_2 \vdash (\Lambda Y.e_3) : (\forall Y.T)} \text{ tabs}$$

- by induction, $\Gamma_1, \Gamma_2[t_2/X], Y \vdash e_3[t_2/X] : T[t_2/X]$
- so $\Gamma_1, \Gamma_2[t_2/X] \vdash (\Lambda Y.e_3[t_2/X]) : (\forall Y.T[t_2/X])$
- rewriting $\Gamma_1, \Gamma_2[t_2/X] \vdash e[t_2/X] : t_1[t_2/X]$



Subject-reduction

Theorem (Preservation) If $e_1 \rightarrow_{\beta}^* e_2$ and $\Gamma \vdash e_1 \in t$ then $\Gamma \vdash e_2 \in t$.

Proof

- Substitution is the key lemma
- Reduction
 - $(\lambda x:t.e_1) e_2 \rightarrow_{\beta} e_1[e_2/x]$
 - $(\Lambda X.e_1)[t] \rightarrow_{\beta} e_1[t/X]$
 - both reductions ignore types
 - can be used in any context



Subject-reduction

Use induction on the length of \rightarrow_{β}^* to reduce to a single β step. There are two cases to show:

term app If $\Gamma \vdash (\lambda x:t_1.e_1) e_2 \in t_2$, then $\Gamma \vdash e_1[e_2/x] \in t_2$

type app If $\Gamma \vdash (\Lambda X.e)[t_1] \in t_2$, then $\Gamma \vdash e[t_1/X] \in t_2$



Subject-reduction (term part)

Term application:

- Show if $\Gamma \vdash (\lambda x:t_1.e_1) e_2 \in t_2$, then $\Gamma \vdash e_1[e_2/x] \in t_2$.
- Then the last fragment of the type proof was

$$\frac{\frac{\frac{\dots}{\Gamma, x:t_1 \vdash e_1 : t_2} 3}{\Gamma \vdash (\lambda x:t_1.e_1) : t_1 \rightarrow t_2} 2 \quad \frac{\dots}{\Gamma \vdash e_2 : t_1} 4}{\Gamma \vdash (\lambda x:t_1.e_1) e_2 \in t_2} 1$$

- By the subst lemma $\Gamma \vdash e_1[e_2/x] : t_2$



Subject-reduction (type part)

Type application:

- Show if $\Gamma \vdash (\lambda X.e)[t_1] \in t_2$, then $\Gamma \vdash e[t_1/X] \in t_2$
- Then $t_2 = t_3[t_1/X]$ and the last fragment of the type proof was:

$$\frac{\frac{\frac{\dots}{\Gamma, X \vdash e : t_3} \quad 3}{\Gamma \vdash (\lambda X.e) : \forall X.t_3} \quad 2}{\Gamma \vdash (\lambda X.e)[t_1] \in t_3[t_1/X]} \quad 1$$

- By the subst lemma $\Gamma \vdash e[t_1/X] : t_3[t_1/X]$



Progress

Theorem (Progress) If $e : t$ and e is not a value (it is not in normal form), then there is an e' such that $e \rightarrow_{\beta} e'$.

Proof If e is not a value, it must either:

- Contain a redex, but then it can be reduced.
- Have weak head normal form $\lambda e_1 e_2 \cdots e_n$, where e_1, \dots, e_n are values. Not possible, because this program is not closed.



Normalization

- System-F is strong-normalizing (every reduction sequence terminates)
 - *“candidats de reducibilite” – an extreme proof*



Next time

- **Computability**
 - *Type-erasure semantics*
 - *Type checking is decidable*
 - *Type inference is not decidable*
 - *Type inference for programs with types in prenex form (like ML)*

