Chapter 23: Universal Types

System F

Power of System F

Properties (Soundness, decidability, parametricity, impredicativity)

System F

- First discovered by Jean-Yves Girard (1972)
- Independently developed by John Reynolds (1974) as polymorphic lambda-calculus
- A natural extension of $\lambda\rightarrow$ with a new form of abstract and application over types:

$$(\lambda X.t_{12})\ [T_2] \longrightarrow [X \mapsto T_2]t_{12}$$

$$(\lambda x{:}T_{11}.t_{12})\ v_2 \longrightarrow [x \mapsto v_2]t_{12}$$

$$\frac{\Gamma, X \vdash t_2 : T_2}{\Gamma \vdash \lambda X.t_2 : \forall X.T_2}$$

$$\frac{\Gamma \vdash t_1 : \forall X.T_{12}}{\Gamma \vdash t_1\ [T_2] : [X \mapsto T_2]T_{12}}$$

Syntax and Evaluation

Syntax

t ::=		*terms:*
	x	*variable*
	$\lambda x{:}T.t$	*abstraction*
	$t\ t$	*application*
	$\lambda X.t$	*type abstraction*
	$t\ [T]$	*type application*
v ::=		*values:*
	$\lambda x{:}T.t$	*abstraction value*
	$\lambda X.t$	*type abstraction value*

Evaluation $\boxed{t \longrightarrow t'}$

$$\frac{t_1 \longrightarrow t_1'}{t_1\ t_2 \longrightarrow t_1'\ t_2} \quad \text{(E-APP1)}$$

$$\frac{t_2 \longrightarrow t_2'}{v_1\ t_2 \longrightarrow v_1\ t_2'} \quad \text{(E-APP2)}$$

$$(\lambda x{:}T_{11}.t_{12})\ v_2 \longrightarrow [x \mapsto v_2]t_{12} \quad \text{(E-APPABS)}$$

$$\frac{t_1 \longrightarrow t_1'}{t_1\ [T_2] \longrightarrow t_1'\ [T_2]} \quad \text{(E-TAPP)}$$

$$(\lambda X.t_{12})\ [T_2] \longrightarrow [X \mapsto T_2]t_{12} \quad \text{(E-TAPPTABS)}$$

Types and Type Context

T	$::=$		*types:*
		$\boxed{X}$	*type variable*
		$T \rightarrow T$	*type of functions*
		$\boxed{\forall X.T}$	*universal type*
Γ	$::=$		*contexts:*
		$\emptyset$	*empty context*
		$\Gamma, x{:}T$	*term variable binding*
		$\boxed{\Gamma, X}$	*type variable binding*

Typing

Typing $\boxed{\Gamma \vdash \mathtt{t} : \mathsf{T}}$

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

$$\frac{\Gamma, \mathtt{x}:\mathsf{T}_1 \vdash \mathtt{t}_2 : \mathsf{T}_2}{\Gamma \vdash \lambda \mathtt{x}:\mathsf{T}_1 . \mathtt{t}_2 : \mathsf{T}_1 \to \mathsf{T}_2} \qquad \text{(T-ABS)}$$

$$\frac{\Gamma \vdash \mathtt{t}_1 : \mathsf{T}_{11} \to \mathsf{T}_{12} \qquad \Gamma \vdash \mathtt{t}_2 : \mathsf{T}_{11}}{\Gamma \vdash \mathtt{t}_1\ \mathtt{t}_2 : \mathsf{T}_{12}} \qquad \text{(T-APP)}$$

$$\frac{\Gamma, \mathsf{X} \vdash \mathtt{t}_2 : \mathsf{T}_2}{\Gamma \vdash \lambda \mathsf{X}.\mathtt{t}_2 : \forall \mathsf{X}.\mathsf{T}_2} \qquad \text{(T-TABS)}$$

$$\frac{\Gamma \vdash \mathtt{t}_1 : \forall \mathsf{X}.\mathsf{T}_{12}}{\Gamma \vdash \mathtt{t}_1\ [\mathsf{T}_2] : [\mathsf{X} \mapsto \mathsf{T}_2]\mathsf{T}_{12}} \qquad \text{(T-TAPP)}$$

Ex.: Defining Polymorphic Functions

- $id = \lambda X.\ \lambda x{:}X.\ x$
  - $id : \forall X.\ X \to X$
  - $id\ [Nat]\ 0 \Rightarrow 0$

- $double = \lambda X.\ \lambda f{:}X\to X.\ \lambda a{:}X.\ f\ (f\ a)$
  - $double : \forall X.\ (X\to X) \to X \to X$
  - $double\ [Nat]\ (\lambda x{:}Nat.\ succ(succ(x)))\ 3 \Rightarrow 7$
  - $quadruple = \lambda X.\ double\ [X\to X]\ (double\ [X])$

- $selfApp = \lambda x{:}\forall X.X\to X.\ x\ [\forall X.X\to X]\ x$
  - $selfApp : (\forall X.\ X\to X) \to (\forall X.\ X \to X)$

Ex.: Polymorphic Lists

- nil : $\forall X.$ List X
- cons : $\forall X.\ X \to$ List $X \to$ List X
- isnil : $\forall X.$ List $X \to$ Bool
- head : $\forall X.$ List $X \to X$
- tail : $\forall X.$ List $X \to$ List X

- map : $\forall X.\ \forall Y.\ (X \to Y) \to$ List $X \to$ List Y

  map = $\lambda X.\ \lambda Y.\ \lambda f{:}\ X \to Y.$

  (fix (λm: (List X) $\to$ (List Y). λl: List X.

  if isnil [X] l then nil [Y]

  else cons [Y] (f (head [X] l)) (m (tail [X] l))))

Exercise: Can you write reverse?

Ex.: Church Encoding

- Church encodings can be carried out in System F.
- CBool = $\forall X.X \to X \to X$;
  - tru = $\lambda X.\ \lambda t{:}X.\ \lambda f{:}X.\ t$;
  - fls = $\lambda X.\ \lambda t{:}X.\ \lambda f{:}X.\ f$;
  - and = ?
- CNat = $\forall X.\ (X \to X) \to X \to X$
  - c0 = $\lambda X.\ \lambda s{:}X \to X.\ \lambda z{:}X.\ z$
  - c1= $\lambda X.\ \lambda s{:}X \to X.\ \lambda z{:}X.\ s\ z$;
  - csucc = $\lambda n{:}CNat.\ \lambda X.\ \lambda s{:}X \to X.\ \lambda z{:}X.\ s\ (n\ [X]\ s\ z)$
  - cplus = $\lambda m{:}CNat.\ \lambda n{:}CNat.\ \lambda X.\ \lambda s{:}X \to X.\ \lambda z{:}X.$
    $m\ [X]\ s\ (n\ [X]\ s\ z)$

Ex.: Encoding Lists

- List X = ∀R. (X→R→R) → R → R
  - nil = λX. (λR. λc:X→R→R. λn:R. n) as List X
  - cons = λX. λhd:X. λtl:List X.

    (λR. λc:X→R→R. λn:R. c hd (tl [R] c n)) as List X;
  - isnil = λX. λl:List X.

    l [Bool] (λhd:X. λtl:Bool. false) true
  - head = λX. λl:List X.

    l [X] (λhd:X. λtl:X. hd) (diverge [X] unit)
  - sum : List Nat → Nat

    sum = ··· definition without using fix ···?

Basic Properties of System F

Very similar to those of the simply typed λ-calculus.

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

Theorem [Progress]: If t is a closed, well-typed term, then either t is a value or there is some t' with $t \to t'$.

Theorem [Normalization]: Well-typed System F terms are normalizing.

Erasure and Type Construction

$$
\begin{aligned}
erase(\lambda \mathtt{x}{:}\mathtt{T}_1.\ \mathtt{t}_2) &= \lambda \mathtt{x}.\ erase(\mathtt{t}_2) \\
erase(\mathtt{t}_1\ \mathtt{t}_2) &= erase(\mathtt{t}_1)\ erase(\mathtt{t}_2) \\
erase(\lambda \mathtt{X}.\ \mathtt{t}_2) &= erase(\mathtt{t}_2)
\end{aligned}
$$

Theorem [Wells, 1994]: It is undecidable whether, given a closed term m of the untyped lambda-calculus, there is some well-typed term t in System F such that erase(t) = m.

Partial Erasure and Type Construction

$$
\begin{array}{lcl}
erase_p(\mathtt{x}) & = & \mathtt{x} \\
erase_p(\lambda\mathtt{x}{:}\mathtt{T}_1.\ \mathtt{t}_2) & = & \lambda\mathtt{x}{:}\mathtt{T}_1.\ erase_p(\mathtt{t}_2) \\
erase_p(\mathtt{t}_1\ \mathtt{t}_2) & = & erase_p(\mathtt{t}_1)\ erase_p(\mathtt{t}_2) \\
erase_p(\lambda\mathtt{X}.\ \mathtt{t}_2) & = & \lambda\mathtt{X}.\ erase_p(\mathtt{t}_2) \\
erase_p(\mathtt{t}_1\ [\mathtt{T}_2]) & = & erase_p(\mathtt{t}_1)\ []
\end{array}
$$

Theorem [Boehm 1985, 1989]: It is undecidable whether, given a closed term s in which type applications are marked but the arguments are omitted, there is some well-typed System F term t such that $erase_p(t) = s$.

Type reconstruction is as hard as higher-order unification.
(But many practical algorithms have been developed)

Erasure and Evaluation Order

Keep type abstraction

$$
\begin{aligned}
erase_v(\mathtt{x}) &= \mathtt{x} \\
erase_v(\lambda\mathtt{x}{:}\mathtt{T}_1.\ \mathtt{t}_2) &= \lambda\mathtt{x}.\ erase_v(\mathtt{t}_2) \\
erase_v(\mathtt{t}_1\ \mathtt{t}_2) &= erase_v(\mathtt{t}_1)\ erase_v(\mathtt{t}_2) \\
erase_v(\lambda\mathtt{X}.\ \mathtt{t}_2) &= \boxed{\lambda\_.\ erase_v(\mathtt{t}_2)} \\
erase_v(\mathtt{t}_1\ [\mathtt{T}_2]) &= erase_v(\mathtt{t}_1)\ \boxed{\mathtt{dummyv}}
\end{aligned}
$$

Theorem: If $erase_v(t) = u$, then either (1) both t and u are normal forms according to their respective evaluation relations, or (2) $t \to t'$ and $u \to u'$, with $erase_v(t') = u'$.

Fragments of System F

- Rank-1 (prenex) polymorphism
  - type variables may not be instantiated with polymorphic types

- Rank-2 polymorphism
  - A type is said to be of rank 2 if no path from its root to a $\forall$ quantifier passes to the left of 2 or more arrows.

$(\forall X.X\rightarrow X)\rightarrow Nat$	OK
$Nat\rightarrow(\forall X.X\rightarrow X)\rightarrow Nat\rightarrow Nat$	OK
$((\forall X.X\rightarrow X)\rightarrow Nat)\rightarrow Nat$	X

Type reconstruction for ranks 2 and lower is decidable, and that for rank 3 and higher of System F is undecidable.

Parametricity

- Uniform behavior of polymorphic programs

```
CBool = ∀X.X→X→X;
tru = λX. λt:X. λf:X. t;
fls = λX. λt:X. λf:X. f;
```

(1) Tru and fls are the only two basic inhabitants of Cbool.

(2) Free Theorem:
e.g., for reverse: ∀X. List X -> List X, we have

```
map f . reverse = reverse . map f
```

Impredicativity

Definition. A definition (of a set, a type, etc.) is called "impredicative" if it involves a quan- tifier whose domain includes the very thing being defined

System F is impredicative, because the type variable X in the type

$$T = \forall X. X \to X$$

ranges over all types, including T itself.

Russell's paradox: let A = { x | x is not in x }, then is "A in A"?

Homework

23.5.1 THEOREM [PRESERVATION]: If $\Gamma \vdash \mathtt{t} : \mathsf{T}$ and $\mathtt{t} \longrightarrow \mathtt{t}'$, then $\Gamma \vdash \mathtt{t}' : \mathsf{T}$. □

Proof: EXERCISE [RECOMMENDED, ★★★]. □

23.5.2 THEOREM [PROGRESS]: If $\mathtt{t}$ is a closed, well-typed term, then either $\mathtt{t}$ is a value or else there is some $\mathtt{t}'$ with $\mathtt{t} \longrightarrow \mathtt{t}'$. □

Proof: EXERCISE [RECOMMENDED, ★★★]. □