LiU-FP2016: Lecture 15

The Polymorphic Lambda Calculus (System F)

Henrik Nilsson

University of Nottingham, UK

This Lecture

- The simply typed lambda calculus.
- Limitations of the simply typed λ-calculus.
- The polymorphic lambda calculus (System F)
- Examples illustrating the power of system F

The Simply Typed λ-Calculus (1)

T	$\rightarrow$		*types:*
	$\vert$	B	*fixed set of base types*
	$\vert$	$T \rightarrow T$	*type of functions*
Γ	$\rightarrow$		*contexts:*
	$\vert$	$\emptyset$	*empty context*
	$\vert$	$\Gamma, x : T$	*context extension*

Note: Need at least ***one*** base type, or there is no way to construct a type of finite size.

The Simply Typed λ-Calculus (2)

t	$\rightarrow$		*terms:*
	$\vert$	x	*variable*
	$\vert$	c	*constant (optional)*
	$\vert$	$\lambda x : T \,.\, t$	*abstraction*
	$\vert$	$t\ t$	*application*
v	$\rightarrow$		*values:*
	$\vert$	c	*constant (optional)*
	$\vert$	$\lambda x : T \,.\, t$	*abstraction*

The Simply Typed λ-Calculus (3)

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

$$\frac{c \text{ is a constant of type } T}{\Gamma \vdash c : T} \quad \text{(T-CONST-c)}$$

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

$$\frac{\Gamma \vdash t_1 : T_{11} \to T_{12} \quad \Gamma \vdash t_2 : T_{11}}{\Gamma \vdash t_1\ t_2 : T_{12}} \quad \text{(T-APP)}$$

Example: TWICE (1)

Consider defining a function twice:

$$\mathrm{twice}(f, x) = f(f(x))$$

Easy in the untyped λ-calculus:

$$\texttt{TWICE} \equiv \lambda\texttt{f}.\lambda\texttt{x}.\texttt{f}\ (\texttt{f}\ \texttt{x})$$

What about the ***simply typed*** λ-calculus?

$$\texttt{TWICE} \equiv \lambda\texttt{f}:???.\lambda\texttt{x}:???.\texttt{f}\ (\texttt{f}\ \texttt{x})$$

What should the types of the arguments be?
Can `TWICE` be used for, say, both `Bool` and `Nat`?

Example: TWICE (2)

Suppose `Bool`, `Nat` $\in B$.

What matters is that the types would be different even if we were to encode them in the base calculus.

Thus we need a ***separate*** definition for ***each*** type at which we want to use `TWICE`:

$$\texttt{TWICEBOOL} \equiv \lambda\texttt{f}:\texttt{Bool}\to\texttt{Bool}.\lambda\texttt{x}:\texttt{Bool}.\texttt{f}\ (\texttt{f}\ \texttt{x})$$

$$\texttt{TWICENAT} \equiv \lambda\texttt{f}:\texttt{Nat}\to\texttt{Nat}.\lambda\texttt{x}:\texttt{Nat}.\texttt{f}\ (\texttt{f}\ \texttt{x})$$

$$\texttt{TWICENATFUN} \equiv \lambda\texttt{f}:(\texttt{Nat}\to\texttt{Nat})\to(\texttt{Nat}\to\texttt{Nat}).\ \lambda\texttt{x}:\texttt{Nat}\to\texttt{Nat}.\texttt{f}\ (\texttt{f}\ \texttt{x})$$

Example: TWICE (3)

We have been forced to define ***essentially the same*** function over and over.

Common CS sensibility suggests ***abstraction*** over the ***varying*** part; i.e., here ***the type***!

Thus, we would like to do something like:

$$\texttt{TWICEPOLY} \equiv \Lambda\texttt{T}.\lambda\texttt{f}:\texttt{T}\to\texttt{T}.\lambda\texttt{x}:\texttt{T}.\texttt{f}\ (\texttt{f}\ \texttt{x})$$

Now:

$$\texttt{TWICEBOOL} \equiv \texttt{TWICEPOLY}\ [\texttt{Bool}]$$

$$\texttt{TWICENAT} \equiv \texttt{TWICEPOLY}\ [\texttt{Nat}]$$

$$\texttt{TWICENATFUN} \equiv \texttt{TWICEPOLY}\ [\texttt{Nat}\to\texttt{Nat}]$$

System F: Abstract Syntax (1)

T	$\rightarrow$		*types:*
	$\mid$	$B \mid T \rightarrow T$	*[as for simply typed]*
	$\mid$	X	*type variable*
	$\mid$	$\forall X\,.\,T$	*universally quantified type*
Γ	$\rightarrow$		*contexts:*
	$\mid$	$\emptyset \mid \Gamma, x : T$	*[as for simply typed]*
	$\mid$	Γ, X	*extension with type variable*

System F: Abstract Syntax (2)

t	$\rightarrow$		*terms:*
	$\mid$	$x \mid c \mid \lambda x : T\,.\,t \mid t\ t$	*[as for simply typed]*
	$\mid$	$\Lambda X\,.\,t$	*type abstraction*
	$\mid$	$t\ [T]$	*type application*
v	$\rightarrow$		*values:*
	$\mid$	$c \mid \lambda x : T\,.\,t$	*[as for simply typed]*
	$\mid$	$\Lambda X\,.\,t$	*type abstraction value*

System F: Typing Rules

T-VAR, (T-CONST-c), T-ABS, T-APP are as before (omitted):

Additional typing rules:

$$\frac{\Gamma, X \vdash t : T}{\Gamma \vdash \Lambda X\,.\,t : \forall X\,.\,T} \quad \text{(T-TABS)}$$

$$\frac{\Gamma \vdash t_1 : \forall X\,.\,T_{12}}{\Gamma \vdash t_1\ [T_2] : [X \mapsto T_2]\ T_{12}} \quad \text{(T-TAPP)}$$

System F: Evaluation Rules

E-APP1, E-APP2, E-APPABS are as before:

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

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

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

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

$$(\Lambda X\,.\,t_{12})\ [T_2] \longrightarrow [X \mapsto T_2]\ t_{12} \quad \text{(E-TAPPABS)}$$

Exercise

Given

$$\texttt{ID} \equiv \Lambda\texttt{T}.\lambda\texttt{x}:\texttt{T}.\texttt{x}$$
$$\Gamma_1 = \emptyset, \texttt{Nat}, \texttt{5} : \texttt{Nat}$$

type check $\texttt{ID}\ [\texttt{Nat}]\ \texttt{5}$ in context Γ_1.

(On whiteboard)

System F: Church Booleans (1)

Recall untyped encoding:

$$\texttt{TRUE} \equiv \lambda\texttt{t}.\lambda\texttt{f}.\texttt{t}$$
$$\texttt{FALSE} \equiv \lambda\texttt{t}.\lambda\texttt{f}.\texttt{f}$$

We need to:

- assign a ***common*** type to these two terms;
- need to work for ***arbitrary*** argument types.

Parametrise on the type:

$$\texttt{CBOOL} \equiv \forall\texttt{X}.\texttt{X}{\rightarrow}\texttt{X}{\rightarrow}\texttt{X}$$

System F: Church Booleans (2)

$$\texttt{CBOOL} \equiv \forall\texttt{X}.\texttt{X}{\rightarrow}\texttt{X}{\rightarrow}\texttt{X}$$

$$\texttt{TRUE} : \texttt{CBOOL}$$
$$\texttt{TRUE} \equiv \Lambda\texttt{X}.\lambda\texttt{t}:\texttt{X}.\lambda\texttt{f}:\texttt{X}.\texttt{t}$$

$$\texttt{FALSE} : \texttt{CBOOL}$$
$$\texttt{FALSE} \equiv \Lambda\texttt{X}.\lambda\texttt{t}:\texttt{X}.\lambda\texttt{f}:\texttt{X}.\texttt{f}$$

$$\texttt{NOT} : \texttt{CBOOL}{\rightarrow}\texttt{CBOOL}$$
$$\texttt{NOT} \equiv \lambda\texttt{b}:\texttt{CBOOL}.\Lambda\texttt{X}.\lambda\texttt{t}:\texttt{X}.\lambda\texttt{f}:\texttt{X}.\texttt{b}\ [\texttt{X}]\ \texttt{f}\ \texttt{t}$$

Normalization

System F is strongly normalizing, like the simply typed λ-calculus.

Homework

- Given `1` : `Nat` and `2` : `Nat`, write down a type-correct application of `TRUE` to `1` and `2` such that the result is `1`.
- Evaluate the above term using the evaluation rules.
- Prove `TRUE` : `CBOOL`.
- Prove `NOT` : `CBOOL`$\longrightarrow$`CBOOL`
- Provide a suitable definition of logical conjunction, `AND`.