G54FOP: Lecture 14

The Polymorphic Lambda Calculus (System F)

Henrik Nilsson

University of Nottingham, UK

This Lecture

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

Rcp: The Simply Typed λ-Calculus (1)

$T \quad \rightarrow$		*types:*
$\mid$	B	*fixed set of base types*
$\mid$	$T{\rightarrow}T$	*type of functions*

$\Gamma \quad \rightarrow$		*contexts:*
$\mid$	$\emptyset$	*empty context*
$\mid$	$\Gamma, x : T$	*context extension*

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

Rcp: The Simply Typed λ-Calculus (2)

t	$\rightarrow$		*terms:*
	$\mid$	x	*variable*
	$\mid$	c	*constant (optional)*
	$\mid$	$\lambda x{:}T.t$	*abstraction*
	$\mid$	$t\ t$	*application*

v	$\rightarrow$		*values:*
	$\mid$	c	*constant (optional)*
	$\mid$	$\lambda x{:}T.t$	*abstraction*

Rcp: The Simply Typed λ-Calculus (3)

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

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

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

$$\frac{\Gamma \vdash t_1 : T_{11} \rightarrow T_{12} \qquad \Gamma \vdash t_2 : T_{11}}{\Gamma \vdash t_1 \; t_2 : T_{12}} \qquad \text{(T-APP)}$$

Example: TWICE (1)

Consider defining a function twice:

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

Example: TWICE (1)

Consider defining a function twice:

$$\text{twice}(f, x) \;=\; f(f(x))$$

Easy in the untyped λ-calculus:

$$\mathtt{TWICE} \;\equiv\; \lambda\mathtt{f}.\lambda\mathtt{x}.\mathtt{f}\,(\mathtt{f}\;\mathtt{x})$$

Example: TWICE (1)

Consider defining a function twice:

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

Easy in the untyped λ-calculus:

$$\mathtt{TWICE} \;\equiv\; \lambda\mathtt{f}.\lambda\mathtt{x}.\mathtt{f}\,(\mathtt{f}\;\mathtt{x})$$

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

$$\mathtt{TWICE} \;\equiv\; \lambda\mathtt{f}{:}???.\lambda\mathtt{x}{:}???.\mathtt{f}\,(\mathtt{f}\;\mathtt{x})$$

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?

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

What should the types of the arguments be?

Example: TWICE (1)

Consider defining a function twice:

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

Easy in the untyped λ-calculus:

$$\mathtt{TWICE} \;\equiv\; \lambda\mathtt{f}.\lambda\mathtt{x}.\mathtt{f}\,(\mathtt{f}\;\mathtt{x})$$

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

$$\mathtt{TWICE} \;\equiv\; \lambda\mathtt{f}{:}???.\lambda\mathtt{x}{:}???.\mathtt{f}\,(\mathtt{f}\;\mathtt{x})$$

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

Example: TWICE (2)

Suppose $\texttt{Bool}, \texttt{Nat} \in B$.

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.

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`**:

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**:

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

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`**:

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

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

Example: TWICE (2)

Suppose $\texttt{Bool}, \texttt{Nat} \in B$.

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**:

$$
\begin{aligned}
\texttt{TWICEBOOL} &\equiv \lambda\texttt{f:Bool}\to\texttt{Bool}.\lambda\texttt{x:Bool}.\texttt{f}\,(\texttt{f}\,\texttt{x})\\
\texttt{TWICENAT} &\equiv \lambda\texttt{f:Nat}\to\texttt{Nat}.\lambda\texttt{x:Nat}.\texttt{f}\,(\texttt{f}\,\texttt{x})\\
\texttt{TWICENATFUN} &\equiv \lambda\texttt{f:}(\texttt{Nat}\to\texttt{Nat})\to(\texttt{Nat}\to\texttt{Nat}).\\
&\qquad \lambda\texttt{x:Nat}\to\texttt{Nat}.\texttt{f}\,(\texttt{f}\,\texttt{x})
\end{aligned}
$$

Example: TWICE (3)

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

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***!

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:

$$\mathtt{TWICEPOLY} \equiv \Lambda\mathtt{T}.\lambda\mathtt{f{:}T}\rightarrow\mathtt{T}.\lambda\mathtt{x{:}T.f}\,(\mathtt{f}\ \mathtt{x})$$

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:

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

Now:

$$\begin{aligned}\mathtt{TWICEBOOL} &\equiv \mathtt{TWICEPOLY}\,[\mathtt{Bool}]\\ \mathtt{TWICENAT} &\equiv \mathtt{TWICEPOLY}\,[\mathtt{Nat}]\\ \mathtt{TWICENATFUN} &\equiv \mathtt{TWICEPOLY}\,[\mathtt{Nat}\to\mathtt{Nat}]\end{aligned}$$

System F: Abstract Syntax (1)

T	$\rightarrow$		*types:*
	$\mid$	$B \mid T \rightarrow T$	*[as for simply typed]*
	$\mid$	$\forall X \,.\, T$	*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).

System F: Typing Rules

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

Additional typing rules:

System F: Typing Rules

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

Additional typing rules:

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

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:

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

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

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

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

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

Exercise

Given

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

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

(On whiteboard)

System F: Church Booleans (1)

Recall untyped encoding:

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

We need to:

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

Any ideas?

$$\mathtt{CBOOL} \equiv ???$$

System F: Church Booleans (1)

Recall untyped encoding:

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

We need to:

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

Parametrise on the type:

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

System F: Church Booleans (2)

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

$$\mathtt{TRUE} : \mathtt{CBOOL}$$

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

$$\mathtt{FALSE} : \mathtt{CBOOL}$$

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

$$\mathtt{NOT} : \mathtt{CBOOL} \rightarrow \mathtt{CBOOL}$$

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

Normalization

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

Homework

- Given $\mathtt{1} : \mathtt{Nat}$ and $\mathtt{2} : \mathtt{Nat}$, 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 $\mathtt{TRUE} : \mathtt{CBOOL}$.
- Prove $\mathtt{NOT} : \mathtt{CBOOL} \rightarrow \mathtt{CBOOL}$
- Provide a suitable definition of logical conjunction, **AND**.