CS520, Handout 21, 14 Nov 2004

Universal Types

Reading: [Pierce, Chapter 23]

Motivation

Read [Pierce, Sect 23.1]

Varieties of Polymorphism

Read [Pierce, Sect 23.2]

Syntax of Types

(compare with [Handout 20, page 2])

X	$::=$	$\mathtt{A} \mid \mathtt{B} \mid \cdots \mid \mathtt{X} \mid \mathtt{Y} \mid \cdots$	*type variables*
T	$::=$		*types*
		$\mathtt{Nat} \mid \mathtt{Bool} \mid \cdots$	*base type*
		X	*type variable*
		$T \to T$	*function type*
		$\forall X.T$	*universal type*

There is no longer a distinction between types and schemes as in [Handout 20].

Notation: Write $\bar{X}$ for a set of types variables $\{X_1, \ldots, X_n\}$ for some $n \geqslant 0$.
Write $\forall \bar{X}.T$ as an abbreviation for $\forall X_1.\cdots.\forall X_n.T$.

Convention: $\forall X_1.\forall X_2.T = \forall X_2.\forall X_1.T$,
i.e., adjacent type-variable bindings commute.

What follows is a presentation of the polymorphic lambda-calculus, also called System F, for unannotated lambda-terms. Compare with the presentation in [Pierce, Fig 23-1, p 343] which is for type-annotated lambda-terms.

What follows is a presentation of the polymorphic lambda-calculus, also called System F, for unannotated lambda-terms. Compare with the presentation in [Pierce, Fig 23-1, p 343] which is for type-annotated lambda-terms.

For simplicity, we restrict attention for a language without arithmetic expressions and other extensions.

Syntax of Unannotated Terms

t ::=		*terms*
	x	*variable*
	$\lambda x.\, t$	*abstraction*
	$t\, t$	*application*
v ::=		*values*
	$\lambda x.\, t$	*abstraction value*

Typing Contexts

(just as in [Handout 07, p. 3])

Typing Contexts

(just as in [Handout 07, p. 3])

Typing Rules for Unannotated Terms

First 3 rules are the same as in [HD 07, p. 3] with type annotation added in (T-ABS):

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

$$\frac{\Gamma, x : T \vdash t : T'}{\Gamma \vdash (\lambda x.\, t) : T \to T'} \qquad \text{(T-ABS)}$$

$$\frac{\Gamma \vdash t_1 : T \to T' \qquad \Gamma \vdash t_2 : T}{\Gamma \vdash t_1\, t_2 : T'} \qquad \text{(T-APP)}$$

Typing Contexts

(just as in [Handout 07, p. 3])

Typing Rules for Unannotated Terms

First 3 rules are the same as in [HD 07, p. 3] with type annotation added in (T-ABS):

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

$$\frac{\Gamma, x : T \vdash t : T'}{\Gamma \vdash (\lambda x.\, t) : T \to T'} \qquad \text{(T-ABS)}$$

$$\frac{\Gamma \vdash t_1 : T \to T' \qquad \Gamma \vdash t_2 : T}{\Gamma \vdash t_1\, t_2 : T'} \qquad \text{(T-APP)}$$

Two new rules, note the side condition in the first below:

$$\frac{\Gamma \vdash t : T}{\Gamma \vdash t : \forall X.T} \quad X \notin FTV(\Gamma) \qquad \text{(T-TABS) or (T-GEN)}$$

$$\frac{\Gamma \vdash t : \forall X.T}{\Gamma \vdash t : [X \mapsto T']\,T} \qquad \text{(T-TAPP) or (T-INST)}$$

Exercise 1 Consider the term t defined by:

$$t = (\lambda \mathtt{f}.\, \mathtt{f}\ \mathtt{f}\,)\,(\lambda \mathtt{x}.\, \mathtt{x}\,)$$

(1) Show that t is not typable according to the typing rules in [Handout 20, page 4]. (In fact, t is not typable according to all the typing rules in all preceding Handouts.)
(2) Show that t is typable according to the typing rules in this handout, page 5.

Exercise 2 The term $\mathtt{omega} = (\lambda \mathtt{x}.\, \mathtt{x}\ \mathtt{x}\,)\,(\lambda \mathtt{x}.\, \mathtt{x}\ \mathtt{x}\,)$ was introduced in [Pierce, p. 65]. Give a precise reason why $\mathtt{omega}$ is not typable according to the typing rules in this handout, page 5.

Exercise 3 We introduce a new typing rule

$$\frac{\Gamma \vdash t : T}{\Gamma \vdash t : \forall X.T} \qquad (\text{T-Gen}')$$

The rule T-Gen$'$ is obtained from T-Gen on page 5 by omitting the side condition $X \notin FTV(\Gamma)$. Show that $\mathtt{omega}$, as defined in the preceding exercise, is typable in the presence of T-Gen$'$.

Exercise 4 Show that every term of the (pure) lambda-calculus is typable according to the rules on page 5 when we add the new rule T-Gen$'$.

Evaluation Rules (just as in [Handout 07, p. 4])

Basic Properties [Pierce, Sect 23.5]

The statements of the Progress Theorem and Preservation Theorem are identical to those of the simply-typed lambda-calculus [Handout 07, p. 7].

Theorem [Progress]
Let t be a closed term. If t is typable, i.e., $\vdash t : T$ is derivable for some type T, then either t is a value or else $t \Rightarrow t'$ for some term t'.

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

Basic Properties [Pierce, Sect 23.5]

The statements of the Progress Theorem and Preservation Theorem are identical to those of the simply-typed lambda-calculus [Handout 07, p. 7].

Theorem [Progress]
Let t be a closed term. If t is typable, i.e., $\vdash t : T$ is derivable for some type T, then either t is a value or else $t \Rightarrow t'$ for some term t'.

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

See [Handout 11] for the difference between Normalization and Strong Normalization. The theorem below is stronger than [Pierce, Theorem 23.5.3, p. 353].

Theorem [Strong Normalization]
Every well-typed term is strongly normalizing.

What About Type Reconstruction? [Pierce, Sect 23.6]

What About Type Reconstruction? [Pierce, Sect 23.6]

Theorem [Undecidability of Type Reconstruction]
([Pierce, Theorem 23.6.1])
It is undecidable, given an arbitrary unannotated lambda-term t, whether there is a context Γ and a type T such that $\Gamma \vdash t : T$ is derivable.

What About Type Reconstruction? [Pierce, Sect 23.6]

Theorem [Undecidability of Type Reconstruction]
([Pierce, Theorem 23.6.1])
It is undecidable, given an arbitrary unannotated lambda-term t, whether there is a context Γ and a type T such that $\Gamma \vdash t : T$ is derivable.

Theorem [Undecidability of Type Checking for unannotated terms]
(not in [Pierce])
It is undecidable, given an arbitrary unannotated lambda-term t, a context Γ and a type T, whether $\Gamma \vdash t : T$ is derivable.

Conjecture

Consider the following variation of the rule T-INST on page 5 of this handout:

$$\frac{\Gamma \vdash t : \forall X.T}{\Gamma \vdash t : [X \mapsto T']T} \quad T' \text{ is quantifier-free} \qquad (\text{T-I}{\small\text{NST}}_0)$$

It is decidable, given an arbitrary unannotated lambda-term t, whether there is a context Γ and a type T such that $\Gamma \vdash t : T$ is derivable according to the rules on page 5 of this handout when **T-INST is replaced by T-INST$_0$.**