CS520, Handout 22, 16 Nov 2004

Universal Types (Continued)

Reading: [Pierce, Chapter 23]

What follows is a presentation of the polymorphic lambda-calculus for type-annotated lambda-terms, just as in [Pierce, Fig 23-1, p 343].

Syntax of Types (just as in [Handout 21, page 3])

Syntax of Annotated Terms (what is new is in red)

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*

Typing Contexts for Annotated Terms

(compare with [Handout 21, p. 5] and [Handout 07, p. 3])

Γ ::=		*contexts*
	$\varnothing$	*empty context*
	$\Gamma,\ x : T$	*term-variable binding*
	$\Gamma,\ X$	*type-variable binding*

Evaluation Rules for Annotated Terms

(see [Pierce, Fig 23-1, p. 343])

Typing Rules for Annotated Terms

First 3 rules are the same as in [HD 07, p. 3]:

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

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

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

Two new rules (compare with corresponding rules in [Handout 21, p. 5]):

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

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

Exercise Let t be a fully type-annotated term of the pure lambda calculus, according to the syntax on page 2 of this handout. Let $t' = \texttt{erase}(t)$ where the function $\texttt{erase}()$ is defined in [Pierce, p. 354]. Let Γ be a context in the sense of [HD 21, p.5], not in the sense of this handout on page 3 (which may contain type-variable bindings). Let T be a type according to the syntax of [HD 21, p. 3], which is the same as the syntax of types in this handout on page 2. Prove carefully the equivalence of the two following statements:

1. There are type variables $X_1, \ldots, X_k$ such that $\Gamma, X_1, \ldots, X_k \vdash t : T$ according to the rules on page 4 of this handout.
2. $\Gamma \vdash t' : T$ according to the rules in [HD 21, p. 5].

Exercise Let t be a fully type-annotated term of the pure lambda calculus, according to the syntax on page 2 of this handout. Let $t' = \texttt{erase}(t)$ where the function $\texttt{erase}()$ is defined in [Pierce, p. 354]. Let Γ be a context in the sense of [HD 21, p.5], not in the sense of this handout on page 3 (which may contain type-variable bindings). Let T be a type according to the syntax of [HD 21, p. 3], which is the same as the syntax of types in this handout on page 2. Prove carefully the equivalence of the two following statements:

1. There are type variables $X_1, \ldots, X_k$ such that $\Gamma, X_1, \ldots, X_k \vdash t : T$ according to the rules on page 4 of this handout.
2. $\Gamma \vdash t' : T$ according to the rules in [HD 21, p. 5].

What About Type Reconstruction? [Pierce, Sect 23.6]

What About Type Checking? [Handout 21, page 9]

Exercise Let t be a fully type-annotated term of the pure lambda calculus, according to the syntax on page 2 of this handout. Let $t' = \texttt{erase}(t)$ where the function $\texttt{erase}()$ is defined in [Pierce, p. 354]. Let Γ be a context in the sense of [HD 21, p.5], not in the sense of this handout on page 3 (which may contain type-variable bindings). Let T be a type according to the syntax of [HD 21, p. 3], which is the same as the syntax of types in this handout on page 2. Prove carefully the equivalence of the two following statements:

1. There are type variables $X_1, \ldots, X_k$ such that $\Gamma, X_1, \ldots, X_k \vdash t : T$ according to the rules on page 4 of this handout.
2. $\Gamma \vdash t' : T$ according to the rules in [HD 21, p. 5].

What About Type Reconstruction? [Pierce, Sect 23.6]

What About Type Checking? [Handout 21, page 9]

Exercise Prove the decidability of the type-checking problem relative to the type-annotated rules (in this handout, page 4).