## CIS 705 — Programming Languages — Spring 2009

# Assignment 5

### Due by 2:30 p.m. on Tuesday, April 21

The context for this assignment is Chapters 9 and 11 of *TAPL*.

**Re-formulating the Simply Typed Lambda Calculus**

If $f$ is a function and $x, y$ are elements of our universe, we define the function $f[x \mapsto y]$ from $\mathsf{dom}(f) \cup \{x\}$ to $\mathsf{ran}(f) \cup \{y\}$ by, for all $z \in \mathsf{dom}(f) \cup \{x\}$,

$$f[x \mapsto y](z) = \begin{cases} y, & \text{if } z = x, \\ f(z), & \text{if } z \neq x. \end{cases}$$

We reformulate the syntax of the simply typed lambda calculus with $\mathsf{Unit}$ as follows, where variables $x$ are as usual. Our *types* are defined by:

| | |
|---|---|
| $T ::=$ | types: |
| $\mathsf{Unit}$ | $\mathsf{Unit}$ type |
| $T \to T$ | type of functions |

As usual, $\to$ associates to the right. Our *terms* are defined by:

| | |
|---|---|
| $t ::=$ | terms: |
| $\mathsf{unit}$ | constant $\mathsf{unit}$ |
| $x$ | variables |
| $\lambda x.\, t$ | abstraction |
| $t\, t$ | application |

And our *values* are defined by:

| | |
|---|---|
| $v ::=$ | values: |
| $\mathsf{unit}$ | constant $\mathsf{unit}$ |
| $\lambda x.\, t$ | abstraction value |

As usual, application associates to the left and abstractions extend as far as possible.

In contrast to TAPL's approach, we do *not* identify abstractions up to the renaming of bound variables, so that, e.g., $\lambda x.x = \lambda y.y$ iff $x = y$. The *free variables* of a term $t$ ($\mathsf{FV}(t)$) is defined recursively as follows:

$$\begin{aligned}\mathsf{FV}(\mathsf{unit}) &= \emptyset,\\ \mathsf{FV}(x) &= \{x\},\\ \mathsf{FV}(\lambda x.\, t) &= \mathsf{FV}(t) \setminus \{x\},\\ \mathsf{FV}(t_1\, t_2) &= \mathsf{FV}(t_1) \cup \mathsf{FV}(t_2).\end{aligned}$$

A term is *closed* iff it has no free variables; otherwise it is *open*. The *substitution of* a *closed* term $s$ *for the free occurrences of* a variable $x$ *in* a term $t$ ($[x \mapsto s]t$) is defined recursively by:

$$\begin{aligned}[x \mapsto s]\mathsf{unit} &= \mathsf{unit},\\ [x \mapsto s]y &= \begin{cases} s, & \text{if } y = x,\\ y, & \text{if } y \neq x,\end{cases}\\ [x \mapsto s](\lambda y.\, t) &= \begin{cases} \lambda y.\, t, & \text{if } y = x,\\ \lambda y.\, [x \mapsto s]t, & \text{if } y \neq x,\end{cases}\\ [x \mapsto s](t_1\, t_2) &= [x \mapsto s]t_1\, [x \mapsto s]t_2.\end{aligned}$$

The *evaluation relation* $\boxed{t \longrightarrow t'}$ between *closed* terms is defined inductively by:

$$\frac{t_1 \longrightarrow t_1'}{t_1\, t_2 \longrightarrow t_1'\, t_2} \qquad \text{(E-App1)}$$

$$\frac{t_2 \longrightarrow t_2'}{v_1\, t_2 \longrightarrow v_1\, t_2'} \qquad \text{(E-App2)}$$

$$(\lambda x.\, t)v \longrightarrow [x \mapsto v]t \qquad \text{(E-AppAbs)}$$

So, in the above, $t_1$, $t_1'$, $t_2$, $t_2'$, $v$ and $v_1$ range over *closed* terms, whereas $\mathsf{FV}(t) \subseteq \{x\}$. A closed term $t$ is a *normal form* iff there is no closed term $t'$ such that $t \longrightarrow t'$. A closed term $t$ is *stuck* iff $t$ is a normal form but $t$ is not a value.

A *typing context* (or just *context*) $\Gamma$ is a function such that $\mathsf{dom}(\Gamma)$ is a finite subset of the variables, and $\mathsf{ran}(\Gamma)$ is a subset of the types. The *typing relation* $\boxed{\Gamma \vdash t : T}$ between typing contexts, terms and types is defined inductively by:

$$\Gamma \vdash \mathsf{unit} : \mathsf{Unit} \qquad \text{(T-Unit)}$$

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

$$\frac{\Gamma[x \mapsto T_1] \vdash t : T_2}{\Gamma \vdash \lambda x.\, t : T_1 \to T_2} \qquad \text{(T-Abs)}$$

$$\frac{\Gamma \vdash t_1 : T_1 \to T_2 \qquad \Gamma \vdash t_2 : T_1}{\Gamma \vdash t_1\, t_2 : T_2} \qquad \text{(T-App)}$$

We say that a closed term $t$ is *well-typed* iff $\emptyset \vdash t : T$ for some type $T$.

**Exercise 1 (5 Points)**

State the Inversion Lemma for the Evaluation Relation. (You don't have to prove that it's correct.)

A consequence of this lemma will be that closed values are normal forms, and you may use this fact without proof in what follows.

**Exercise 2 (5 Points)**

State the Principle of Induction on the Evaluation Relation. (You don't have to prove that it's correct.)

**Exercise 3 (5 Points)**

State the Inversion Lemma for the Typing Relation. (You don't have to prove that it's correct.)

**Exercise 4 (5 Points)**

State the Principle of Induction on the Typing Relation. (You don't have to prove that it's correct.)

**Exercise 5 (10 Points)**

Prove the Free Variables Lemma:

For all contexts $\Gamma$, terms $t$ and types $T$, if $\Gamma \vdash t : T$, then $\mathsf{FV}(t) \subseteq \mathsf{dom}(\Gamma)$.

**Exercise 6 (5 Points)**

Prove the Canonical Forms Lemma:

> For all closed values $v$:
>
> - if $\emptyset \vdash v : \mathsf{Unit}$, then $v = \mathsf{unit}$.
> - if $\emptyset \vdash v : T_1 \to T_2$, for some types $T_1$ and $T_2$, then $v$ is an abstraction.

**Exercise 7 (10 Points)**

Either prove or disprove the Determinacy of Evaluation Proposition:

> For all closed terms $t$, $t'$ and $t''$, if $t \to t'$ and $t \to t''$, then $t' = t''$.

**Exercise 8 (10 Points)**

Either prove or disprove the Uniqueness of Typing Proposition:

> For all contexts $\Gamma$, terms $t$ and types $T$ and $T'$, if $\Gamma \vdash t : T$ and $\Gamma \vdash t : T'$, then $T = T'$.

**Exercise 9 (20 Points)**

Either prove or disprove the Progress Theorem:

> For all closed terms $t$, if $t$ is well-typed, then $t$ is not stuck.

**Exercise 10 (25 Points)**

Either prove or disprove the Preservation Theorem:

> For all closed terms $t$ and $t'$ and types $T$, if $\emptyset \vdash t : T$ and $t \to t'$, then $\emptyset \vdash t' : T$.