CIS 705 — Programming Languages — Spring 2010

Assignment 5

Due by 2:30 p.m. on Tuesday, April 20

The context for this assignment is Chapter 9 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 as follows, where variables x are as usual, and n ranges over the natural numbers. Our *types* are inductively defined by:

$T ::=$	types:
Unit	unit type
Nat	type of natural numbers
$T \to T$	type of functions

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

$t ::=$	terms:
unit	unit constant
n	natural number constant
succ	successor constant
pred	predecessor constant
$\mathsf{if}[T]$	conditional constant for type T
x	variable
$\lambda x : T.\, t$	abstraction
$t\, t$	application

And our *values* (a subset of the terms) are inductively defined by:

$v ::=$	values:
unit	unit constant
n	natural number constant
succ	successor constant
pred	predecessor constant
$\mathsf{if}[T]$	conditional constant for type T
$\mathsf{if}[T]\, v$	conditional constant for type T with one argument
$\mathsf{if}[T]\, v_1\, v_2$	conditional constant for type T with two arguments
$\lambda x : T.\, t$	abstraction

As usual, application associates to the left and abstractions extend as far as possible. (E.g., in the definition of the set of values, $\mathsf{if}[T]\, v_1\, v_2$ means $(\mathsf{if}[T]\, v_1)\, v_2$.)

E.g.,

$$(\lambda x : \mathsf{Nat}.\, \mathsf{if}[\mathsf{Nat}]\, x\, (\lambda y : \mathsf{Unit}.\, 1)\, (\lambda y : \mathsf{Unit}.\, \mathsf{succ}\, x))\, 3$$

is a term.

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

$$\begin{aligned}
\mathsf{FV}(\mathsf{unit}) &= \emptyset, \\
\mathsf{FV}(n) &= \emptyset, \\
\mathsf{FV}(\mathsf{succ}) &= \emptyset, \\
\mathsf{FV}(\mathsf{pred}) &= \emptyset, \\
\mathsf{FV}(\mathsf{if}[T]) &= \emptyset, \\
\mathsf{FV}(x) &= \{x\}, \\
\mathsf{FV}(\lambda x : T.\, t) &= \mathsf{FV}(t) \setminus \{x\}, \\
\mathsf{FV}(t_1\, t_2) &= \mathsf{FV}(t_1) \cup \mathsf{FV}(t_2).
\end{aligned}$$

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:

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

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

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

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

$$\mathsf{if}[T]\, 0\, v_1\, v_2 \to v_1\, \mathsf{unit} \qquad \text{(E-AppIfZero)}$$

$$\mathsf{if}[T]\, (n+1)\, v_1\, v_2 \to v_2\, \mathsf{unit} \qquad \text{(E-AppIfNonZero)}$$

$$\mathsf{succ}\, n \to n + 1 \qquad \text{(E-AppSucc)}$$

$$\mathsf{pred}\, 0 \to 0 \qquad \text{(E-AppPredZero)}$$

$$\mathsf{pred}(n+1) \to n \qquad \text{(E-AppPredNonZero)}$$

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

So, in the above, t_1, t_1', t_2, t_2', v, v_1 and v_2 range over *closed* terms, whereas $\mathsf{FV}(t) \subseteq \{x\}$.

E.g.,

$$(\lambda x : \mathsf{Nat}.\, \mathsf{if}[\mathsf{Nat}]\, x\, (\lambda y : \mathsf{Unit}.\, 1)\, (\lambda y : \mathsf{Unit}.\, \mathsf{succ}\, x))\, 3$$

evaluates to

$$\mathsf{if}[\mathsf{Nat}]\, 3\, (\lambda y : \mathsf{Unit}.\, 1)\, (\lambda y : \mathsf{Unit}.\, \mathsf{succ}\, 3),$$

which evaluates to

$$(\lambda y : \mathsf{Unit}.\, \mathsf{succ}\, 3)\, \mathsf{unit},$$

which evaluates to

$$\mathsf{succ}\,3,$$

which evaluates to 4.

A closed term t is a *normal form* iff there is no closed term t' such that $t \to t'$. Thus a closed term t is not a normal form iff there is a closed term t' such that $t \to t'$.

A closed term t is *stuck* iff t is a normal form but t is not a value. Thus a closed term t is not stuck iff t is a value or t is not a normal form.

A *typing context* (or just *context*) Γ 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 inductively defined by:

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

$$\Gamma \vdash n : \mathsf{Nat} \qquad \text{(T-Nat)}$$

$$\Gamma \vdash \mathsf{succ} : \mathsf{Nat} \to \mathsf{Nat} \qquad \text{(T-Succ)}$$

$$\Gamma \vdash \mathsf{pred} : \mathsf{Nat} \to \mathsf{Nat} \qquad \text{(T-Pred)}$$

$$\Gamma \vdash \mathsf{if}[T] : \mathsf{Nat} \to (\mathsf{Unit} \to T) \to (\mathsf{Unit} \to T) \to T \qquad \text{(T-If)}$$

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

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

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

E.g.,

$$\emptyset \vdash (\lambda x : \mathsf{Nat}.\, \mathsf{if}[\mathsf{Nat}]\, x\, (\lambda y : \mathsf{Unit}.\, 1)\, (\lambda y : \mathsf{Unit}.\, \mathsf{succ}\, x))\, 3 : \mathsf{Nat}.$$

The **Free Variables Lemma** says that, for all contexts Γ, terms t and types T, if $\Gamma \vdash t : T$, then $\mathsf{FV}(t) \subseteq \mathsf{dom}(\Gamma)$. (It can be proved using induction on the typing relation; see the model answers to last year's Assignment 5 for a similar proof.) Thus, if $\emptyset \vdash t : T$, then t is closed.

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

The **Weakening Lemma** says that, for all contexts Γ and Γ', terms t and types T, if $\Gamma \vdash t : T$ and $\Gamma \subseteq \Gamma'$, then $\Gamma' \vdash t : T$. (It can be proved by induction on the typing relation; see the model answers to last year's Assignment 5 for a similar proof.)

The **Typing of Substitutions Lemma** says that, for all contexts Γ, variables y, terms t and t', and types T, T', if $\Gamma[y \mapsto T'] \vdash t : T$ and $\emptyset \vdash t' : T'$, then $\Gamma \vdash [y \mapsto t']t : T$. (It can be proved by induction on the typing relation; see the model answers to last year's Assignment 5 for a similar proof.)

Exercises

When solving the following exercises, you may use—without proof—the facts and lemmas stated above, as well as the inversion lemmas and principles of induction for the inductively defined sets and relations. When doing a proof by induction on a set or relation, you *must* write down the predicate P you are using, and then do your proof using P.

Exercise 1 (10 Points)

Prove the **Normal Form Lemma**:

> For all closed values v, v is a normal form.

Exercise 2 (25 Points)

Prove 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 3 (15 Points)

Prove the **Uniqueness of Typing Proposition**:

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

Exercise 4 (25 Points)

Prove the **Progress Theorem**:

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

Exercise 5 (25 Points)

Prove the **Preservation Theorem**:

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