$\boxed{\textbf{CIS 500 Software Foundations}}$

Homework Assignment 5

Type systems, simply-typed λ-calculus

Due: Monday, October 31, 2005, by noon

Submission instructions:

You must submit your solutions electronically (in ascii, postscript, or PDF format). Electronic solutions should be submitted following the same instructions as last time; these can be found at `http://www.seas.upenn.edu/~cis500/homework.html`. Do **not** email your solutions to us.

1 Exercise As quick self-checks, you should have done exercises 8.2.3, 8.3.5, 9.2.1, 9.2.2, 9.2.3, and 9.4.1 in TAPL when you read Chapter 8 and 9. If you did not, do them now.[1] Do **not** turn in anything for this exercise.

2 Exercise Exercise 8.3.4 in TAPL.

3 Exercise Exercise 8.3.6 in TAPL.

4 Exercise Show the derivations of the following typing judgments:

1. `if false then 0 else succ 0` : `Nat`
2. `succ (if iszero 0 then succ 0 else pred 0)` : `Nat`

5 Exercise If we make the following changes to the Arith language, do the following theorems become false or remain true. If they become false, give a counterexample.

The theorems:

- Uniqueness of types (Theorem 8.2.4)
- Progress (Theorem 8.3.2)
- Preservation (Theorem 8.3.3)

The changes to Arith:

1. Say we remove the rule E-PredZero.
2. Say we add the rule: $\texttt{if}\ \texttt{t}_1\ \texttt{then}\ \texttt{t}_2\ \texttt{else}\ \texttt{t}_3 \longrightarrow \texttt{t}_2$
3. Say we add the rules

   $\texttt{pred true} \longrightarrow \texttt{false}$
   $\texttt{pred false} \longrightarrow \texttt{true}$

   and the following typing rule:

$$\frac{\texttt{t} : \texttt{Bool}}{\texttt{pred t} : \texttt{Nat}}$$

4. Say we add the rule

[1] In general, you should be doing all the one $\star$ exercises when you read TAPL. They are designed to make sure you have a minimal understanding of what you're reading.

$$\texttt{if 0 then } t_2 \texttt{ else } t_3 \longrightarrow t_2$$

and the following typing rule:

$$\frac{t_1 : \texttt{Nat} \qquad t_2 : T \qquad t_3 : T}{\texttt{if } t_1 \texttt{ then } t_2 \texttt{ else } t_3 : T}$$

5. Say we add the following typing axiom: $0 : \texttt{Bool}$

6 Exercise Prove the type soundness theorem for Arith.

Theorem: If $t : T$ and $t \longrightarrow^* t'$ and $t' \not\longrightarrow$, then t' is a value.

You should prove this theorem by induction on the following definition of multistep evaluation. In your proof, you may use the Preservation and Progress theorems from TAPL (Theorems 8.3.2 and 8.3.3).

$$\frac{t \longrightarrow t' \qquad t' \longrightarrow^* t''}{t \longrightarrow^* t''}\ \text{EV-STEP} \qquad \frac{}{t \longrightarrow^* t}\ \text{EV-REF}$$

7 Exercise The following exercise concerns the big-step semantics for the simply-typed lambda calculus. For simplicity, consider just the language with a base value 0 of type Nat and functions.

The big-step semantics of this calculus is:

$$\frac{}{v \Downarrow v}\ \text{B-VALUE}$$

$$\frac{t_1 \Downarrow \lambda x : T_1.t_{11} \qquad t_2 \Downarrow v_2 \qquad [x \mapsto v_2]t_{11} \Downarrow v}{t_1\ t_2 \Downarrow v}\ \text{B-APP}$$

1. The following theorem is true

   Theorem: If $\emptyset \vdash t : T$ then $t \Downarrow v$ and $\emptyset \vdash v : T$.

   but the following proof is flawed. Find the error in the proof below.

   Proof by induction on the structure of t.

   - Case $t = 0$. This case is trivial because $0 \Downarrow 0$ by B-VALUE and $\emptyset \vdash 0 : \texttt{Nat}$.
   - Case $t = x$, a variable. This case is trivial because it is not the case that $\emptyset \vdash x : T$ for any T.
   - Case $t = \lambda x : T_1.t_1$, an abstraction. This case is trivial because $\lambda x : T_1.t_1 \Downarrow \lambda x : T_1.t_1$ by B-VALUE and $\emptyset \vdash \lambda x : T_1.t_1 : T$ by assumption.
   - Case $t = t_1\ t_2$.
     - By inversion (Lemma 9.3.1), $\emptyset \vdash t_1 : T_1 \to T$ and $\emptyset \vdash t_2 : T_1$.
     - By induction $t_1 \Downarrow v_1$ and $\emptyset \vdash v_1 : T_1 \to T$.
     - By induction $t_2 \Downarrow v_2$ and $\emptyset \vdash v_2 : T_1$.
     - By canonical forms (Lemma 9.3.4), $v_1 = \lambda x : T_1.t_{11}$.
     - By inversion (Lemma 9.3.1), $x : T_1 \vdash t_{11} : T$.
     - By substitution (Lemma 9.3.8), $\emptyset \vdash [x \mapsto v_2]t_{11} : T$.
     - By induction $[x \mapsto v_2]t_{11} \Downarrow v$ and $\emptyset \vdash v : T$.
     - By B-APP, $t_1\ t_2 \Downarrow v$, and by the above $\emptyset \vdash v : T$.

2. An easier-to-prove statement of the theorem reads:

Theorem: If $\emptyset \vdash \mathtt{t} : \mathtt{T}$ and $\mathtt{t} \Downarrow \mathtt{v}$ then $\emptyset \vdash \mathtt{v} : \mathtt{T}$.

Prove this theorem.

8 Exercise Suppose we give an implicitly-typed presentation of the simply-typed lambda-calculus. In this case, we define typing judgments that apply to *untyped* lambda calculus terms (plus booleans).

1. What is the typing rule for abstractions?
2. This style breaks the uniqueness of types property (Theorem 9.3.3). Give a counterexample.
3. State the canonical forms lemma for this calculus (analogous to 9.3.4).

9 Debriefing

1. How many hours did each person in your group spend on this assignment, including time taken to read TAPL?
2. Would you rate it as easy, moderate, or difficult?
3. Did everyone in your study group participate?
4. How deeply do you feel you understand the material it covers (0%–100%)?

If you have any other comments, we would like to hear them; please send them to `cis500@cis.upenn.edu`.