Semantics, WS 2011-2012: Solution for Assignment 12

Prof. Dr. Gert Smolka, Dr. Chad Brown

Exercise 12.1 Extend the given Coq development of simply typed λ-calculus to T.

Solution to Exercise 12.1 See the Coq file.

Exercise 12.2 In our development of simply typed λ-calculus we used a capturing substitution. As a consequence the following form of preservation fails:

```
Lemma preservation' Gamma t T t' :
type Gamma t T -> step t t' -> type Gamma t' T.
```

Give a counterexample showing this form of preservation fails. Use the counterexample to prove the following theorem in Coq.

```
Lemma preservation'_fails : exists Gamma, exists t, exists T, exists t',
type Gamma t T /\ step t t' /\ ~ type Gamma t' T.
```

Solution to Exercise 12.2 Let Γ be $\emptyset^{x}_{\mathsf{Nat}\to\mathsf{Nat}}$,
T be $\mathsf{Nat} \to \mathsf{Nat}$,
t be $(\lambda y : \mathsf{Nat} \to \mathsf{Nat}.\lambda x : \mathsf{Nat}.yx)(\lambda y : \mathsf{Nat}.xy)$
and t' be $\lambda x : \mathsf{Nat}.(\lambda y : \mathsf{Nat}.xy)x$.
See the Coq file for the Coq proof.

Exercise 12.3 Write the definition of the logical relation $R_T t$ from memory.

Solution to Exercise 12.3 Look in the lecture notes or in the Coq file.

Exercise 12.4 Prove the following lemma relating substitutions and typing.

```
Lemma substitution_lemma : forall Gamma t T theta,
  type Gamma t T ->
  (forall x S, Gamma x = Some S -> exists s, theta x = Some s /\ type empty s S) ->
  type empty (simsubst theta t) T.
```

You will need to prove a generalization by induction on *type Gamma t T*.

Solution to Exercise 12.4 See the Coq file.

Exercise 12.5 Prove the basic lemma.

```
Lemma Basic_lemma : forall Gamma t T theta,
  type Gamma t T -> R' Gamma theta -> R T (simsubst theta t).
```

Use the following (as yet unproven) lemma

```
Lemma R_beta : forall S T x t, type (update empty x S) t T ->
  (forall s, R S s -> R T (subst x s t)) ->
  forall s, R S s -> R T (tmA (tmL x S t) s).
```

Solution to Exercise 12.5 See the Coq file.

Exercise 12.6 Prove well-typed terms terminate using the basic lemma.

```
Definition ter (t : tm) : Prop := terminates step t.

Theorem ter_step : forall t T,
  type empty t T -> ter t.
```

Solution to Exercise 12.6 See the Coq file.

The remaining exercises are from Chapter 3 of the Introduction to Computational Logic lecture notes. We use T as syntax for a universe of types.

Exercise 12.7 Decide for each pair whether the two terms are alpha equivalent.

a) $\forall x{:}\mathsf{T}.x \to x$ and $\forall y{:}\mathsf{T}.y \to y$
b) $\lambda xy{:}\mathsf{T}.x \to y \to x$ and $\lambda yx{:}\mathsf{T}.y \to x \to y$
c) $\lambda xyz{:}\mathsf{T}.x \to (\forall u{:}x.z \to y)$ and $\lambda yxz{:}\mathsf{T}.y \to (\forall u{:}x.z \to x)$
d) $\lambda x{:}\mathsf{T}.x$ and $\forall x{:}\mathsf{T}.x$
e) $(\lambda xy{:}\mathsf{T}.y)\mathsf{T}$ and $(\lambda x{:}\mathsf{T}.\lambda z{:}\mathsf{T}.z)\mathsf{T}$

Solution to Exercise 12.7

a) These are alpha equivalent.
b) These are alpha equivalent.
c) These are not alpha equivalent.
d) These are not alpha equivalent.
e) These are alpha equivalent.

Exercise 12.8 Beta reduce the term

```
Compute fun (X : Type) (f : X -> X -> X) (y : X) => (fun x y : X => f x y) y.
```

by hand and check your result with Coq.

Solution to Exercise 12.8 `fun (X : Type) (f : X -> X -> X) (y y0 : X) => f y y0.`

Exercise 12.9 Give a beta redex where a local variable must be renamed to avoid capturing when the beta redex is reduced.

Solution to Exercise 12.9

$$(\lambda f : \mathsf{T} \to \mathsf{T}.\, \forall x \in \mathsf{T}.\, fx)(gx)$$

Exercise 12.10 Compute the normal forms of the following terms.

a) $(\lambda x : \mathsf{T}.\lambda g : \mathsf{T} \to \mathsf{T} \to \mathsf{T}.\ (\lambda f : \mathsf{T} \to \mathsf{T}.\, \forall x \in \mathsf{T}.\, fx)(gx))\,\mathsf{T}$

b) $\lambda x : \mathsf{T}.(\lambda f : x \to x \to x.\lambda yz : x.f(fyz)(fzy))(\lambda yz : x.z)$

Solution to Exercise 12.10

a) $\lambda g : \mathsf{T} \to \mathsf{T} \to \mathsf{T}.\lambda x : \mathsf{T}.g\mathsf{T}x$

b) $\lambda x : \mathsf{T}.\lambda y : x.\lambda z : x.y$