CS 520 Problem Set 10

Yarom Gabay

December 16, 2004

Problem 1, Ex 1, Handout 22 p. 6

Part 1

```
t = (lambda f. f f) (lambda x. x)
```

Since there is no *let* in this term, then there is no polymorphic type to any of the subterms. When we try to type the subterm `f f`, we will have some monomorphic type bound to `f` and so we will not be able to apply *T-App*.

$$\dfrac{\dfrac{\dfrac{f : T_1 \in f : T_1}{f : T_1 \vdash f : T_1}\ \text{T-Var} \qquad \dfrac{f : T_1 \in f : T_1}{f : T_1 \vdash f : T_1}\ \text{T-Var}}{f : T_1 \vdash f\ f :?}\ \text{T-App}}{\emptyset \vdash \lambda f.f\ f :?}\ \text{T-Abs}$$

Part 2

```
t = (lambda f. f f) (lambda x. x)
```

Abbreviations:

t1 = lambda f. f f
t2 = lambda x. x
$T_1 =$ $X \to X$

We will break this down to a few subderivations

(d1)

$$\dfrac{\dfrac{\dfrac{\dfrac{f : \forall X.X \to X \in \Gamma}{f : \forall X.X \to X \vdash f : \forall X.X \to X}\ \text{T-Var}}{f : \forall X.X \to X \vdash f : T_1 \to T_1}\ \text{T-Inst} \qquad \dfrac{\dfrac{f : \forall X.X \to X \in \Gamma}{f : \forall X.X \to X \vdash f : \forall X.X \to X}\ \text{T-Var}}{f : \forall X.X \to X \vdash f : T_1}\ \text{T-Inst}}{f : \forall X.X \to X \vdash (f\ f) : T_1}\ \text{T-App}}{\emptyset \vdash t1 : (\forall X.X \to X) \to T_1}$$

(d2)

$$\dfrac{\dfrac{\dfrac{x : X \in \Gamma}{x : X \vdash x : X}\;\text{T-Var}}{\emptyset \vdash t2 : X \to X}\;\text{T-Abs}}{\emptyset \vdash t2 : \forall X.X \to X}\;\text{T-Gen}$$

Derivation for the whole term:

$$\dfrac{(d1) \quad (d2)}{\emptyset \vdash t : T1}\;\text{T-App}$$

Problem 2, Ex 2, Handout 22 p. 6

We know that under these rules, if t is a well-typed term then t is normalizing. Since *omega* is a diverging term it implies that *omega* is not well-typed.

Problem 3, Ex 3, Handout 22 p. 6

`omega = (lambda x. x x) (lambda x. x x)`

Abbreviations:

t1 =	lambda x. x x
$T_1 =$	$\forall X.X$
$T_2 =$	$(\forall X.X) \to (\forall X.X)$

We will break this down to a few subderivations

(d1)

$$\dfrac{\dfrac{\dfrac{\dfrac{x : T_1 \in x : T_1}{x : T_1 \vdash x : T_1}\;\text{T-Var}}{x : T_1 \vdash x : T_2}\;\text{T-Inst} \qquad \dfrac{x : T_1 \in x : T_1}{x : T_1 \vdash x : T_1}\;\text{T-Var}}{x : T_1 \vdash x\ x : T_1}\;\text{T-App}}{\emptyset \vdash t1 : T_2}\;\text{T-Abs}$$

(d2)

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{x : Y \in x : Y}{x : Y \vdash x : Y}\;\text{T-Var}}{x : Y \vdash x : \forall Y.Y}\;\text{T-Gen'}}{x : Y \vdash x : Y \to Y}\;\text{T-Inst} \qquad \dfrac{x : Y \in x : Y}{x : Y \vdash x : Y}\;\text{T-Var}}{x : Y \vdash x\ x : Y}\;\text{T-App}}{\emptyset \vdash t1 : Y \to Y}\;\text{T-Abs}}{\emptyset \vdash t1 : \forall Y.(Y \to Y)}\;\text{T-Gen}}{\emptyset \vdash t1 : T_2 \to T_2}\;\text{T-Inst}$$

Derivation for the whole term:

$$\frac{(d2) \quad (d1)}{\emptyset \vdash omega : T_2}$$

Problem 4, Ex 4, Handout 22 p. 6

We need to show that every term of the pure lambda calculus is typable using the rules: T-Var, T-Abs, T-App, T-Gen', and T-Inst.

For the purpose of this question let's define the following,

basic type: type T is a basic type if it is either a type variable , or has the form $T = X \to T'$ where X is a type variable and T' is a basic type.

basic context: context Γ is a basic context if every binding in Γ is of the form $x : X$ where X is a type variable..

We will make the following assumption. w.n.l.o.g. we could rename variables in the term in such a way that we have the following

1. All type variables are of the form $x_1, x_2, x3, ..$
2. No two lambda bindings have the same variable name.
3. The set of free variables and the set of bound variables are disjoint.

For example the term $\lambda x.(x\ \lambda y.(\lambda x.(x\ y)))$ will become $\lambda x_1.(x_1\ \lambda x_2.(\lambda x_3.(x_3\ x_2)))$

We will also keep the convention that every binding we introduce, will have the form $x_i : X_i$ where X_i's index follows x_i's index.

claim: if t is a term of the pure lambda calculus, then $\Gamma \vdash t : T$ is derivable where T is a basic type and Γ is a basic context.

proof: by induction on the structure of t.

t=x$_i$ (some variable with index i)
according to (T-Var) we can make the following derivation,
$x_i : X_i \vdash x_i : X_i$, we can see X_i is a basic type and $x_i : X_i$ is a basic context.

t=lambda x$_i$**. t1**
by IH we have,

(s1) $\Gamma \vdash t1 : T1$ for a basic context Γ and for a basic type T1.

It might be that $x_i \in FV(t1)$. In this case we define Γ' to be Γ with $x_i : X_i$ binding removed. if $x_i \notin FV(t1)$ then we define Γ' to be Γ. According to the weakening lemma (we need it only in one case) and according to (s1) we can derive,

(s2) $\Gamma', x_i : X_i \vdash t1 : T1$

from (s2) and (T-Abs) we can derive,

(s3) $\Gamma' \vdash$ lambda x. t1 : $X_i \rightarrow T1$

$X_i \rightarrow T1$ is a basic type and Γ' is a basic context.

t=t1 t2
by applying the IH twice we have,

(s1) $\Gamma 1 \vdash t1 : T1$ for a basic context $\Gamma 1$ and for a basic type T1.

(s2) $\Gamma 2 \vdash t2 : T2$ for a basic context $\Gamma 2$ and for a basic type T2.

Let Γ be $\Gamma 1 \cup \Gamma 2$. Γ is well-defined, since if there is a variable x_i such that $x_i \in dom(\Gamma_1)$ and $x_i \in dom(\Gamma_2)$ than we know that x_i binding will be $x_i : X_i$ on both. Γ is of course a basic context.

By the weakening lemma and (s1),(s2) we can derive,

(s3) $\Gamma \vdash t1 : T1$

(s4) $\Gamma \vdash t2 : T2$

Since $T1$ is a basic type let's consider the possible structure of $T1$,

- **case:** $T1 = X_i$, where X_i is a type variable.
  We can make the following derivation,

  (d1)
  $$\dfrac{\dfrac{\Gamma \vdash t1 : X_i}{\Gamma \vdash t1 : \forall X_i.X_i}\ (T-Gen')}{\Gamma \vdash t1 : T2 \rightarrow T2}\ (T-Inst)$$

  According to (d1), (s4) and (T-App) we can derive, $\Gamma \vdash t1\ t2 : T2$ where $T2$ is a basic type.

- **case:** $T1 = X_i \rightarrow T'$, where X_i is a type variable and T' is basic type.
  We can make the following derivation,

  (d2)
  $$\dfrac{\dfrac{\Gamma \vdash t1 : X_i \rightarrow T'}{\Gamma \vdash t1 : \forall X_i.(X_i \rightarrow T')}\ (T-Gen')}{\Gamma \vdash t1 : T2 \rightarrow T'}\ (T-Inst)$$

  According to (d2), (s4) and (T-App) we can derive, $\Gamma \vdash t1\ t2 : T'$ where T' is a basic type.

Problem 5, Ex 1, Handout 23 p. 5

- $\vdash_{22}$ denote derivation according to the rules in HD22
- $\vdash_{23}$ denote derivation according to the rules in HD23

part 1

if there are type variables $X_1, ..., X_k$ such that $\Gamma, X_1, ..., X_k \vdash_{23} t : T$
then $\Gamma \vdash_{22} t' : T$

proof: by induction on the structure of t.

- **case t=x** (t'=x).
  By the assumption and rule T-Var$_{23}$ we have,

  (s1) $x : T \in \Gamma$

  According to (s1) and T-Var$_{22}$ we derive, $\Gamma \vdash_{22} t' : T$

- **case t=t1 t2** (t'=erase(t1) erase(t2)).
  By the assumption and rule T-App$_{23}$ we have,

  (s1) $\Gamma, X_1, ..., X_k \vdash_{23} t1 : T' \to T$ for some type T'
  (s2) $\Gamma, X_1, ..., X_k \vdash_{23} t2 : T'$

  The IH with (s1) imply,

  (s3) $\Gamma \vdash_{22} erase(t1) : T' \to T$

  The IH with (s2) imply,

  (s4) $\Gamma \vdash_{22} erase(t2) : T'$

  ¿From (s3), (s4) and T-App$_{22}$ we conclude,
  $\Gamma \vdash_{22} t' : T$

- **case t=**$\lambda X.t_1 : T$
  By the assumption we have,

  (s1) $T = \forall X.T_1$
  (s2) $\Gamma, X \vdash_{23} t_1 : T_1$ (here $X \notin FTV(\Gamma)$)

  The IH with (s2) imply,

  (s3) $\Gamma \vdash_{22} t_1' : T_1$ where $t_1' = erase(t_1)$

  Using the fact that $X \notin FTV(\Gamma)$, with T-Gen$_{22}$ and (s3) we derive,

  (s4) $\Gamma \vdash_{22} t_1' : \forall X.T_1$ and so we have, $t' = t_1'$ where $erase(t) = erase(\lambda X.t_1) = erase(t_1) = t_1'$

- **case t=**$t_1[T'] : T$
  By the assumption we have,

  (s1) $T = [X \mapsto T']T''$
  (s2) $\Gamma \vdash_{23} t_1 : \forall X.T''$

  The IH with (s2) imply,

  (s3) $\Gamma \vdash_{22} t'_1 : \forall X.T''$ where $t'_1 = erase(t_1)$

  Using T-Inst$_{22}$ and (s3) we derive,

  (s4) $\Gamma \vdash_{22} t'_1 : [X \mapsto T']T''$ and so we have, $t' = t'_1$ where $erase(t) = erase(t_1[T']) = erase(t_1) = t'_1$

part 2

if $\Gamma \vdash_{22} t' : T$
then there are type variables $X_1, ..., X_k$ such that $\Gamma, X_1, ..., X_k \vdash_{23} t : T$

proof: by induction on the type derivation of t'.

- **case (T-Var$_{22}$)**
  ¿From the assumption we have $t' : T \in \Gamma$ and since $t' = erase(t) = t$ using T-Var$_{23}$ we can derive $\Gamma \vdash_{23} t : T$.

- **case (T-Abs$_{22}$)**
  By the assumption and rule T-Abs$_{22}$ we have,

  (s1) t' is of the form $\lambda x : T.t'_2$
  (s2) $T = T' \to T''$
  (s3) $\Gamma, x : T' \vdash_{22} t'_2 : T''$

  The IH with (s1),(s2),(s3) imply,

  (s4) $\Gamma, x : T', X_1, .., X_k \vdash_{23} t_2 : T''$ for type variable $X_1, .., X_k$ where $t_2 = erase(t'_2)$

  Using T-Abs$_{23}$ and (s4) we derive,

  (s5) $\Gamma, X_1, .., X_k \vdash_{23} (\lambda x : T'.t_2) : T' \to T''$ where $erase(\lambda x : T'.t_2) = t'$

- **case (T-App$_{22}$)**
  By the assumption and rule T-App$_{22}$ we have,

  (s1) t' is of the form $t'_1\ t'_2$
  (s2) $\Gamma \vdash_{22} t'_1 : T' \to T$
  (s3) $\Gamma \vdash_{22} t'_2 : T'$

  By applying the IH twice with (s2),(s3) we have, (s4) $\Gamma, X'_1, .., X'_m \vdash_{23} t_1 : T' \to T$ where $erase(t_1) = t'_1$

(s5) $\Gamma, X''_1, .., X''_l \vdash_{23} t_2 : T'$ where $erase(t_2) = t'_2$

Let $X_1, .., X_k$ be the union of $X'_1, .., X'_m$ and $X''_1, .., X''_l$. We can now say,

(s6) $\Gamma, X_1, .., X_k \vdash_{23} t_1 : T' \to T$

(s7) $\Gamma, X_1, .., X_k \vdash_{23} t_2 : T'$

¿From (s6),(s7) and T-App$_{23}$ we can derive,

(s8) $\Gamma, X_1, .., X_k \vdash_{23} t_1\ t_2 : T$ and so we have, $t = t_1\ t_2$ where $erase(t) = erase(t_1)\ erase(t_2) = t'_1\ t'_2 = t'$

- **case (T-Gen$_{22}$)**
  By the assumption and rule T-Gen$_{22}$ we have,

  (s1) $T = \forall X.T'$

  (s2) $X \notin FTV(\Gamma)$

  (s3) $\Gamma \vdash_{22} t' : T'$

  The IH with (s1),(s3) imply,

  (s4) $\Gamma, X_1, .., X_k \vdash_{23} t_1 : T'$ where $erase(t_1) = t'$

  Since (s2) holds, we can say

  (s5) $\Gamma, X_1, .., X_k, X \vdash_{23} t_1 : T'$ (it could be that X is already in the set $X_1, .., X_k$ in that case we don't need to add X. The important thing is that $X \notin FTV(\Gamma)$)

  Using T-Gen$_{23}$ and (s2),(s4)

  (s6) $\Gamma, X_1, .., X_k \vdash_{23} \lambda X.t_1 : \forall X.T'$, and so we have, $t = \lambda X.t_1$ where $erase(t) = erase(\lambda X.t_1) = erase(t_1) = t'$

- **case (T-Inst$_{22}$)**
  By the assumption and rule T-Inst$_{22}$ we have,

  (s1) $T = [X \mapsto T']T''$

  (s2) $\Gamma \vdash_{22} t' : \forall X.T''$

  The IH with (s2) imply,

  (s3) $\Gamma, X_1, .., X_k \vdash_{23} t_1 : \forall X.T''$ where $erase(t_1) = t'$

  By T-Inst$_{23}$ and (s3) we conclude,

  (s3) $\Gamma, X_1, .., X_k \vdash_{23} t_1[T'] : [X \mapsto T']T''$ and so we have, $t = t_1[T']$ where $erase(t) = erase(t_1) = t'$

Problem 6, Ex 2, Handout 23 p. 5

Type-checking with the type-annotated rules is decidable. The type-annotated rules are syntax directed which means that for every term there is just one rule that we can apply. This means we can implement a deterministic algorithm for type checking. This algorithm terminates since in every typing rule we always proceed with a term which is a subtype of the term we begin with.

In order to make this question complete we really have to show an algorithm (which is a straight forward implementation of the rules) but we skip this part.