CS 520 Problem Set 9

Yarom Gabay

Nov 22, 2004

Problem 1, Ex 1, Handout 20 p. 6

```
let double = lambda f. lambda a . f (f a) in
let p = double (lambda x:Nat. succ (succ x)) 2 in
let q = double (lambda x:Bool. x) false in
0
```

Abbreviations:

t1 =	lambda f. lambda a . f (f a)
t2 =	let p = double (lambda x:Nat. succ (succ x)) 2 in let q = double (lambda x:Bool. x) false in 0
t3 =	lambda a . f (f a)
t4 =	f (f a)
t5 =	(lambda x:Nat. succ (succ x))
t6 =	(lambda x:Bool. x)
t7 =	let q = t1 t6 false in 0
Γ_1 =	$f : T \to T, a : T$

We will break this down to a few subderivations

(d1) for some type T,

$$\dfrac{\dfrac{\dfrac{\dfrac{f : T \to T \in \Gamma}{\Gamma_1 \vdash f : T \to T} \quad \dfrac{\dfrac{f : T \to T \in \Gamma}{\Gamma_1 \vdash f : T \to T}\text{(T-Var)} \quad \dfrac{a : T \in \Gamma}{\Gamma_1 \vdash a : T}\text{(T-Var)}}{\Gamma_1 \vdash f\ a : T}\text{(T-App)}}{\Gamma_1 \vdash t4 : T}\text{(T-App)}}{f : T \to T \vdash t3 : T \to T}\text{(T-Abs')}}{\emptyset \vdash t1 : (T \to T) \to T \to T}\text{(T-Abs')}$$

(d2)

$$\dfrac{\dfrac{\dfrac{(d1)\ with\ T = Nat}{\emptyset \vdash t1 : (Nat \to Nat) \to Nat \to Nat} \quad \dfrac{\dfrac{\cdots}{x : Nat \vdash succ\ (succ\ x) : Nat}}{\emptyset \vdash t5 : Nat \to Nat}\text{(T-Abs)}}{\emptyset \vdash t1\ t5 : Nat \to Nat}\text{(T-App)} \quad \emptyset \vdash 2 : Nat}{\emptyset \vdash t1\ t5\ 2 : Nat}$$

(d3)

$$\dfrac{\dfrac{\dfrac{(d1)\ with\ T = Bool}{\emptyset \vdash t1 : (Bool \to Bool) \to Bool \to Bool} \quad \dfrac{\dfrac{x : Bool \in \Gamma}{x : Bool \vdash x : Bool}\text{(T-Var)}}{\emptyset \vdash t6 : Bool \to Bool}\text{(T-Abs)}}{\emptyset \vdash t1\ t6 : Bool \to Bool}\text{(T-App)} \quad \emptyset \vdash false : Bool}{\emptyset \vdash t1\ t6\ false : Bool}$$

Derivation for the whole term: **let double = t1 in t2**

(N and B stands for Nat and Bool respectively)

$$\dfrac{\dfrac{(d1)\ with\ T = Nat}{\emptyset \vdash t1 : (N \to N) \to N \to N} \quad \dfrac{\dfrac{(d2)}{\emptyset \vdash t1\ t5\ 2 : N} \quad \dfrac{\dfrac{(d3)}{\emptyset \vdash t1\ t6\ false : B} \quad \emptyset \vdash [q \mapsto (t1\ t6\ false)]0 : N}{\emptyset \vdash [p \mapsto (t1\ t5\ 2)]t7 : N}}{\emptyset \vdash [\text{double} \mapsto t1]t2 : N}}{\emptyset \vdash \text{let double=t1 in t2} : Nat}\text{(T-LetPoly)}$$

Problem 2, Ex 2, Handout 20 p. 6

the premise "$\Gamma \vdash t_1 : T_1$" can not be omitted. Though the type T_1 in not used later in the derivation, we need that for a case where x does not occur free in the body of the *let*, for example,
let $x = t_1$ in 0. Without this premise we can type the whole term even when t_1 is not typable.

Problem 3, Ex 1, Handout 21 p. 5

```
let f = lambda x. lambda y . x y in f f succ 3
```

Abbreviations:

t1 =	lambda x. lambda y . x y
t2 =	lambda y . x y
$\Gamma_1 =$	$f : \forall X \forall Y.(Y \to X) \to Y \to X$
$S_1 =$	$\forall X \forall Y.(Y \to X) \to Y \to X$
$T_1 =$	$(Nat \to Nat) \to Nat \to Nat$

We will break this down to a few subderivations

(d1)

$$\dfrac{\dfrac{\dfrac{\dfrac{x : Y \to X \in \Gamma \quad Y \to X \preceq Y \to X}{x : Y \to X, y : Y \vdash x : Y \to X}\ (\text{T-Var}) \qquad \dfrac{y : Y \in \Gamma \quad Y \preceq Y}{x : Y \to X, y : Y \vdash y : Y}\ (\text{T-Var})}{x : Y \to X, y : Y \vdash x\ y : X}\ (\text{T-App})}{x : Y \to X \vdash t2 : Y \to X}\ (\text{T-Abs'})}{\emptyset \vdash t1 : (Y \to X) \to Y \to X}\ (\text{T-Abs'})$$

(d2)

$$\dfrac{\dfrac{\dfrac{\dfrac{f : S_1 \in \Gamma_1 \quad S_1 \preceq T_1 \to T_1}{\Gamma_1 \vdash f : T_1 \to T_1} \quad \dfrac{f : S_1 \in \Gamma_1 \quad S_1 \preceq T_1}{\Gamma_1 \vdash f : T_1}}{\Gamma_1 \vdash f\ f : T_1} \qquad \Gamma_1 \vdash succ : Nat \to Nat}{\Gamma_1 \vdash f\ f\ succ : Nat \to Nat} \qquad \Gamma_1 \vdash 3 : Nat}{f : \forall X \forall Y.(Y \to X) \to Y \to X \vdash f\ f\ succ\ 3 : Nat}$$

Derivation for the whole term: let f = t1 in f f succ 3

$$\dfrac{\dfrac{(d1)}{\emptyset \vdash t1 : (Y \to X) \to Y \to X} \quad \dfrac{(d2)}{f : \forall X \forall Y.(Y \to X) \to Y \to X \vdash f\ f\ succ\ 3 : Nat} \quad X, Y \notin \Gamma}{\emptyset \vdash let\ f = t1\ in\ f\ f\ succ\ 3 : Nat}\ (\text{T-LetPoly'})$$

Problem 4, Ex 2, Handout 21 p. 5

```
lambda x. let f = lambda y . x y in f f succ 3
```

This term has no type derivation because of the assumption
$\bar{X} = FTV(T) - FTV(\Gamma)$ in T-LetPoly rule.

In the typing of the term $f\ f\ succ\ 3$, we are using the type of x which is monomorphic and is already set in the environment. This will stop us from giving a quantified type for f and as a result different occurrences of f in the term $f\ f\ succ\ 3$ will not be given different types and the term will not be typable.

Problem 5, Ex 3, Handout 21 p. 5

```
lambda x. let f = lambda y . x y in f f succ 3
```

Abbreviations:

t1 =	lambda y . x y
$\Gamma_1 =$	$x : Y \to X, y : Y$
$\Gamma_2 =$	$x : Y \to X, f : \forall X \forall Y.Y \to X$
$S_1 =$	$\forall X \forall Y.Y \to X$
$T_1 =$	$(Nat \to Nat) \to (Nat \to Nat) \to Nat \to Nat$
$T_2 =$	$(Nat \to Nat) \to Nat \to Nat$

We will break this down to a few subderivations

(d1)

$$\dfrac{\dfrac{\dfrac{x : Y \to X \in \Gamma_1 \quad Y \to X \preceq Y \to X}{\Gamma_1 \vdash x : Y \to X}\ \text{(T-Var)} \quad \dfrac{y : Y \in \Gamma_1 \quad Y \preceq Y}{\Gamma_1 \vdash y : Y}\ \text{(T-Var)}}{\Gamma_1 \vdash x\ y : X}\ \text{(T-App)}}{x : Y \to X \vdash t1 : Y \to X}\ \text{(T-Abs')}$$

(d2)

$$\dfrac{\dfrac{\dfrac{\dfrac{f : S_1 \in \Gamma_2 \quad S_1 \preceq T_1}{\Gamma_2 \vdash f : T_1} \quad \dfrac{f : S_1 \in \Gamma_2 \quad S_1 \preceq Nat \to Nat}{\Gamma_2 \vdash f : Nat \to Nat}}{\Gamma_2 \vdash f\ f : T_2} \quad \Gamma_2 \vdash succ : Nat \to Nat}{\Gamma_2 \vdash f\ f\ succ : Nat \to Nat} \quad \Gamma_2 \vdash 3 : Nat}{\Gamma_2 \vdash f\ f\ succ\ 3 : Nat}$$

Derivation for the whole term: lambda x. let f=t1 in f f succ 3

$$\dfrac{\dfrac{\dfrac{(d1)}{x : Y \to X \vdash t1 : Y \to X} \quad \dfrac{(d2)}{\Gamma_2 \vdash f\ f\ succ\ 3 : Nat}}{x : Y \to X \vdash let\ f = t1\ in\ f\ f\ succ\ 3 : Nat}\ \text{(T-LetPoly'')}}{\emptyset \vdash \lambda x.\ let\ f = t1\ in\ f\ f\ succ\ 3 : (Y \to X) \to Nat}\ \text{(T-Abs')}$$

Problem 6, Ex 4, Handout 21 p. 5

With the type derivation we made (in the previous exercise) for the term,

```
lambda x. let f = lambda y . x y in f f succ 3
```

We can apply the term, for instance, on M where,

```
M = lambda x. succ x
(M=succ is also acceptable)
```

(lambda x. let f = lambda y . x y in f f succ 3) M
$\to$ let f = lambda y . M y in f f succ 3
$\to$ (lambda y . M y) (lambda y . M y) succ 3
$\to$ M (lambda y . M y) succ 3
$\to$ succ (lambda y . M y) succ 3

Stuck!

Problem 7

Context Structure ($\hat{\Gamma}$)

$\hat{\Gamma}$ bind term variables to two different kind of variables: type variables and scheme variables [HD 21, p. 7]. $\hat{\Gamma}$ is of the form:
$\hat{\Gamma} = x_1 : \alpha_1, ..., x_m : \alpha_m, x_{m+1} : X_1, ..., x_{m+n} : X_n$

Constraints Set Structure (the result of the algorithm)

Constraints set has three types of constraints, as stated in [HD 21, p. 8],

- $\alpha \preceq X$
- $T = T'$
- $\alpha =_\Gamma T$

Constraints Algorithm

```
constraints: contexts × terms -> (types × constraint sets)∪{error}

constraints(G,t)=
IF t is a variable x
 THEN IF G(x) is defined
  THEN IF G(x) = α                              (*scheme variable*)
       THEN LET X = fresh type variable
            IN return (X,{α ⪯ X})
  ELSE return (G(x),{})                          (*type variable*)
 ELSE 'error'
ELSE
IF t = lambda x:X. t'
THEN LET constraints((G,x:X),t') = (T,C)
     IN return (X->T,C)
ELSE
IF t = t1 t2
THEN LET constraints(G,t1) = (T1,C1) and
         constraints(G,t2) = (T2,C2) and
         X = fresh type variable
     IN return (X, C1 union C2 union {T1=T2->X})
ELSE
IF t = let x:α = t1 in t2
THEN LET constraints(G,t1) = (T1,C1) and
         constraints((G,x:α),t2) = (T2,C2)
     IN return (T2,C1 union C2 union {α =ΓT1})
ELSE 'error'
```