Mid-term Exam

Foundations of Software

November 14, 2008

Last Name : ______________________________

First Name : ______________________________

Section : ______________________________

Exercise	**Points**	**Achieved Points**
1	10	
2	10	
3	10	
4	6	
Total	36	

Exercise 1 : Progress and Preservation (10 points)

Recall the following properties of the language of numbers and booleans (typed arithmetic expressions, see reference page at the end of the assignment):

- *Progress:* If $\Gamma \vdash$ `t : T`, then either `t` is a value or else `t` $\to$ `t'`.
- *Preservation:* If $\Gamma \vdash$ `t : T` and `t` $\to$ `t'`, then $\Gamma \vdash$ `t' : T`.

Each part of this exercise suggests a different way of changing the language of typed arithmetic and boolean expressions. Note that these changes are not cumulative: each part starts from the original language. In each part, for each property, indicate whether the property remains true or becomes false after the suggested change. If a property becomes false, give a counterexample. In the last part (and only in the last part), show using a suitable inductive proof that preservation remains true.

1. Suppose we add the following typing axiom: `pred 0 : Bool`

   Solution: Progress holds. Preservation does not hold: `pred 0 : Bool` $\to$ `0 : Nat`

2. Suppose we add the following typing rule:

   $$\text{T-FUNNY1}\ \frac{\texttt{t1 : Nat}}{\texttt{if true then t1 else t2 : Nat}}$$

   Solution: Progress holds. Preservation holds.

3. Suppose we add the following typing rule:

   $$\text{T-FUNNY2}\ \frac{\texttt{t : Bool}}{\texttt{pred t : Bool}}$$

   Solution: Progress does not hold: `pred true : Bool`, but it is not a value and cannot take a step. Preservation holds.

4. Suppose we add a type `Foo` and the following two typing rules:

   $$\text{T-FOO1}\ \frac{\texttt{t : Nat}}{\texttt{pred t : Foo}} \qquad \text{T-FOO2}\ \frac{\texttt{t : Foo}}{\texttt{succ t : Nat}}$$

   Solution: Progress holds. Preservation does not hold: `pred 0 : Foo`, but `pred 0` $\to$ `0 : Nat`

5. Suppose we add a type `Bar` and the following two typing rules:

   $$\text{T-BAR1}\ \frac{\texttt{t : Nat}}{\texttt{succ t : Bar}} \qquad \text{T-BAR2}\ \frac{\texttt{t : Bar}}{\texttt{pred t : Nat}}$$

   Show using a suitable inductive proof that preservation remains true.

   Solution:

   (a) case $\texttt{t} = \texttt{succ t}_1$ and T-Bar1 is the last rule used in the typing derivation. Since $\texttt{t} \to \texttt{t}'$ we have $\texttt{t}' = \texttt{succ t}_1'$ with $\texttt{t}_1 \to \texttt{t}_1'$ (by E-Succ) and by IH $\texttt{t}_1'$ `: Nat`. By T-Bar1, $\texttt{succ t}_1'$ `: Bar`.

   (b) case $\texttt{t = pred t}_1$ and T-Bar2 is the last rule used in the typing derivation. We have two sub cases for $\texttt{t} \to \texttt{t}'$:

      i. $\texttt{pred t}_1 \to \texttt{pred t}_1'$ (E-Pred) and $\texttt{t}_1 \to \texttt{t}_1'$. By IH, $\texttt{t}_1'$ `: Bar`. By T-Bar2, $\texttt{pred t}_1'$ `: Nat`.

      ii. `pred (succ nv)` $\to$ `nv`. By inversion of T-Bar1, `nv : Nat`

Exercise 2 : Typings (10 points)

For each of the following lambda terms either find a possible type or indicate that the term is not typable.
Note that the lambda calculus that we consider here contains a primitive type `Bool` as well as a conditional term `if` t_1 `then` t_2 `else` t_3 with the usual typing.

1. λx. λy. λz. `(x y) z`

   Solution: $(a \to b \to c) \to a \to b \to c$

2. λx. λy. λz. `x (y z)`

   Solution: $(a \to b) \to (c \to a) \to (c \to b)$

3. λx. λy. `x (y x)`

   Solution: $(a \to b) \to ((a \to b) \to a) \to b$

4. λx. `x` (λy. `y x`)

   Solution: not typable

5. λx. λy. λz. `if (x y) then y else z`

   Solution: $(a \to \texttt{Bool}) \to a \to a \to a$

6. λx. λy. λz. `x (if (y z) then (z x) else true)`

   Solution: $(\texttt{Bool} \to a) \to (((\texttt{Bool} \to a) \to \texttt{Bool}) \to \texttt{Bool}) \to ((\texttt{Bool} \to a) \to \texttt{Bool}) \to a$

Exercise 3 : Simply Typed Lambda Calculus (10 points)

Consider the Simply Typed Lambda Calculus (STLC) you have seen in the course. We add the following evaluation rule:

$(\lambda\texttt{x:T. t x}) \to \texttt{t}$ if $\texttt{x}$ not free in $\texttt{t}$

Show that progress and preservation still hold. You don't need to repeat the whole proof for STLC, but you should mention which cases are not treated by your proof because their proof holds unchanged.

Solution:

- Progress. If t is a well-typed term, it can either take a step or it is a value.

  Progress still holds because we add one more evaluation rule, and don't remove any of the old ones.

- Preservation. If $\Gamma \vdash t : T$ and $t \to t'$ then $\Gamma \vdash t' : T$.

  Proof. We prove this by induction on typing derivations. The original proof needs to be adjusted in case `T-ABS`

  $$\text{T-ABS}\ \frac{\Gamma, x : T \vdash t_1 : T_1}{\Gamma \vdash \lambda x : T.t_1 : T \to T_1}$$

  We need to treat the additional case $t_1 = t\ x$, which triggers the new evaluation rule.

  $$\text{T-ABS}\ \frac{\text{T-APP}\ \dfrac{\Gamma, x : T \vdash t : T_2 \to T_1 \quad \Gamma, x : T \vdash x : T}{\Gamma, x : T \vdash (t\ x) : T_1}}{\Gamma \vdash \lambda x : T.(t\ x) : T \to T_1}\ x\ \textit{free in}\ t$$

  Since t is well-typed, we conclude that $T_2 = T$, and since x is free in t

  $\Gamma, x : T \vdash t : T_2 \to T_1$ implies $\Gamma \vdash t : T \to T_1$.

  $(\lambda x : T.t\ x : T \to T_1) \to t$ and we just showed that $\Gamma \vdash t : T \to T_1$. $\square$

Exercise 4 : Behavioral equivalence (6 points)

Find two terms in untyped lambda calculus which are:

- behaviorally equivalent when using call-by-value but not when using call-by-name.
- behaviorally equivalent when using call-by-name but not when using call-by-value.

Solution: Let
$\Omega = \lambda x.(x\ x)\ \lambda x.(x\ x)$
$tru = \lambda x.\lambda y.x$
$fls = \lambda x.\lambda y.y$

1. Equivalent in CBV but not in CBN
   $tru\ \Omega$
   $fls\ \Omega$

   Under CBV, both term diverge, trying to reduce Ω to a value before applying it. Under CBN, the first one reduces to Ω and then diverges, while the second reduces to $\lambda y.y$.

2. Equivalent in CBN, but not in CBV

   $fls\ \lambda x.\Omega$
   $fls\ (\lambda x.\Omega\ x)$

   Under CBN, both reduce to $\lambda y.y$. Under CBV, the first one reduces to $\lambda y.y$, while the second one diverges evaluating the argument to fls.

For reference: typed arithmetic expressions

Syntax for arithmetic expressions (TAPL, p.91):

$$
\begin{array}{lll|lll}
t & ::= & \textbf{terms}: & v & ::= & \textbf{values}: \\
& \mid \texttt{true} & \textit{constant true} & & \mid \texttt{true} & \textit{true value} \\
& \mid \texttt{false} & \textit{constant false} & & \mid \texttt{false} & \textit{false value} \\
& \mid \texttt{if}\ t\ \texttt{then}\ t\ \texttt{else}\ t & \textit{condition} & & \mid nv & \textit{numeric value} \\
& \mid \texttt{0} & \textit{constant zero} & & & \\
& \mid \texttt{succ}\ t & \textit{successor} & nv & ::= & \textbf{numeric values}: \\
& \mid \texttt{pred}\ t & \textit{predecessor} & & \mid \texttt{0} & \textit{zero value} \\
& \mid \texttt{iszero}\ t & \textit{zero test} & & \mid \texttt{succ}\ nv & \textit{successor value}
\end{array}
$$

Evaluation rules (TAPL, p.41):

$$(\text{E-PredZero})\ \texttt{pred 0} \longrightarrow \texttt{0} \qquad (\text{E-PredSucc})\ \texttt{pred (succ}\ nv_1\texttt{)} \longrightarrow nv_1$$

$$(\text{E-Succ})\ \frac{t_1 \longrightarrow t_1'}{\texttt{succ}\ t_1 \longrightarrow \texttt{succ}\ t_1'} \qquad (\text{E-Pred})\ \frac{t_1 \longrightarrow t_1'}{\texttt{pred}\ t_1 \longrightarrow \texttt{pred}\ t_1'}$$

$$(\text{E-IsZeroZero})\ \texttt{iszero}\ 0 \longrightarrow \texttt{true} \qquad (\text{E-IsZeroSucc})\ \texttt{iszero (succ}\ nv_1\texttt{)} \longrightarrow \texttt{false}$$

$$(\text{E-IsZero})\ \frac{t_1 \longrightarrow t_1'}{\texttt{iszero}\ t_1 \longrightarrow \texttt{iszero}\ t_1'} \qquad (\text{E-If})\ \frac{t_1 \longrightarrow t_1'}{\texttt{if}\ t_1\ \texttt{then}\ t_2\ \texttt{else}\ t_3 \longrightarrow \texttt{if}\ t_1'\ \texttt{then}\ t_2\ \texttt{else}\ t_3}$$

$$(\text{E-IfTrue})\ \texttt{if true then}\ t_2\ \texttt{else}\ t_3 \longrightarrow t_2 \qquad (\text{E-IfFalse})\ \texttt{if false then}\ t_2\ \texttt{else}\ t_3 \longrightarrow t_3$$

Typing rules (TAPL, p.93):

$$(\text{T-True})\ \texttt{true} : \texttt{Bool} \qquad (\text{T-False})\ \texttt{false} : \texttt{Bool}$$

$$(\text{T-If})\ \frac{t_1 : \texttt{Bool} \quad t_2 : \texttt{T} \quad t_3 : \texttt{T}}{\texttt{if}\ t_1\ \texttt{then}\ t_2\ \texttt{else}\ t_3 : \texttt{T}} \qquad (\text{T-Zero})\ 0 : \texttt{Nat}$$

$$(\text{T-Succ})\ \frac{t_1 : \texttt{Nat}}{\texttt{succ}\ t_1 : \texttt{Nat}} \qquad (\text{T-Pred})\ \frac{t_1 : \texttt{Nat}}{\texttt{pred}\ t_1 : \texttt{Nat}}$$

$$(\text{T-IsZero})\ \frac{t_1 : \texttt{Nat}}{\texttt{iszero}\ t_1 : \texttt{Bool}}$$