

---

# Mid-term Exam

Foundations of Software

November 14, 2008

---

Last Name : \_\_\_\_\_

First Name : \_\_\_\_\_

Section : \_\_\_\_\_

<b>Exercise</b>	<b>Points</b>	<b>Achieved Points</b>
<b>Total</b>	0	

## 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  $t \rightarrow t'$ .
- *Preservation*: If  $\Gamma \vdash t : T$  and  $t \rightarrow t'$ , 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`

2. Suppose we add the following typing rule:

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

3. Suppose we add the following typing rule:

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

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

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

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

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

Show using a suitable inductive proof that preservation remains true.

## 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.  $\lambda x. \lambda y. \lambda z. (x\ y)\ z$
2.  $\lambda x. \lambda y. \lambda z. x\ (y\ z)$
3.  $\lambda x. \lambda y. x\ (y\ x)$
4.  $\lambda x. x\ (\lambda y. y\ x)$
5.  $\lambda x. \lambda y. \lambda z. \text{if } (x\ y) \text{ then } y \text{ else } z$
6.  $\lambda x. \lambda y. \lambda z. x\ (\text{if } (y\ z) \text{ then } (z\ x) \text{ else true})$

### 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:

$$(\lambda x:T. t \ x) \rightarrow t \quad \text{if } x \text{ not free in } t$$

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.

## 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.

For reference: typed arithmetic expressions

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

$t ::=$   true   false   if $t$ then $t$ else $t$   0   succ $t$   pred $t$   iszero $t$	<b>terms :</b> <i>constant true</i> <i>constant false</i> <i>condition</i> <i>constant zero</i> <i>successor</i> <i>predecessor</i> <i>zero test</i>		$v ::=$   true   false   $nv$  $nv ::=$   0   succ $nv$	<b>values :</b> <i>true value</i> <i>false value</i> <i>numeric value</i>  <b>numeric values :</b> <i>zero value</i> <i>successor value</i>
---	---	--	--	--

Evaluation rules (TAPL, p.41):

(E-PREDZERO) $\text{pred } 0 \longrightarrow 0$	(E-PREDSUCC) $\text{pred } (\text{succ } nv_1) \longrightarrow nv_1$
(E-SUCC) $\frac{t_1 \longrightarrow t'_1}{\text{succ } t_1 \longrightarrow \text{succ } t'_1}$	(E-PRED) $\frac{t_1 \longrightarrow t'_1}{\text{pred } t_1 \longrightarrow \text{pred } t'_1}$
(E-ISZEROZERO) $\text{iszero } 0 \longrightarrow \text{true}$	(E-ISZEROSUCC) $\text{iszero } (\text{succ } nv_1) \longrightarrow \text{false}$
(E-ISZERO) $\frac{t_1 \longrightarrow t'_1}{\text{iszero } t_1 \longrightarrow \text{iszero } t'_1}$	(E-IF) $\frac{t_1 \longrightarrow t'_1}{\text{if } t_1 \text{ then } t_2 \text{ else } t_3 \longrightarrow \text{if } t'_1 \text{ then } t_2 \text{ else } t_3}$
(E-IFTRUE) $\text{if true then } t_2 \text{ else } t_3 \longrightarrow t_2$	(E-IFFALSE) $\text{if false then } t_2 \text{ else } t_3 \longrightarrow t_3$

Typing rules (TAPL, p.93):

(T-TRUE) $\text{true} : \text{Bool}$	(T-FALSE) $\text{false} : \text{Bool}$
(T-IF) $\frac{t_1 : \text{Bool} \quad t_2 : \text{T} \quad t_3 : \text{T}}{\text{if } t_1 \text{ then } t_2 \text{ else } t_3 : \text{T}}$	(T-ZERO) $0 : \text{Nat}$
(T-SUCC) $\frac{t_1 : \text{Nat}}{\text{succ } t_1 : \text{Nat}}$	(T-PRED) $\frac{t_1 : \text{Nat}}{\text{pred } t_1 : \text{Nat}}$
(T-ISZERO) $\frac{t_1 : \text{Nat}}{\text{iszero } t_1 : \text{Bool}}$	