

Family Name: \_\_\_\_\_

Given Names: \_\_\_\_\_

Signature: \_\_\_\_\_

THE UNIVERSITY OF NEW SOUTH WALES

Sample Exam

Session 2 2007

**COMP3161/COMP9161**

**Concepts of Programming Languages**

- Time allowed: **2 hours**
- Total number of questions: **4**
- Answer **all** questions
- The questions **are** of equal value
- Answer *each* question in a **separate** answer booklet
- This paper may **not** be retained by the candidate
- **Answers must be written in ink**, with the exception of graphs
- Drawing instruments or rules may be used
- There is a 3% penalty if you do not fill in your student number and name correctly

### Question 1 [25 Marks]

Consider the following inductive definition of evaluation rules for a restricted form of boolean expressions.

**Boolean expressions:**

$$\begin{array}{c} \overline{\text{true bool}} \quad \overline{\text{false bool}} \\ \frac{b_1 \text{ bool} \quad b_2 \text{ bool}}{\text{and}(b_1, b_2) \text{ bool}} \quad \frac{b \text{ bool}}{\text{not}(b) \text{ bool}} \end{array}$$

**Evaluation rules:**

$$\begin{array}{c} \overline{\text{and}(\text{true}, b) \mapsto b} \quad \overline{\text{and}(\text{false}, b) \mapsto \text{false}} \\ \overline{\text{not}(\text{false}) \mapsto \text{true}} \quad \overline{\text{not}(\text{true}) \mapsto \text{false}} \\ \frac{b \mapsto b'}{\text{not}(b) \mapsto \text{not}(b')} \quad \frac{b_1 \mapsto b'_1}{\text{and}(b_1, b_2) \mapsto \text{and}(b'_1, b_2)} \end{array}$$

A) [7 marks]

Give the derivation of the evaluation for the following expression:

- $\text{and}(\text{not}(\text{false}), \text{and}(\text{true}, \text{not}(\text{true})))$

B) [7 marks]

Are the rules unambiguous? If so, briefly explain why. If not, give an example expression for which the set of rules allow more than a single derivation.

C) [11 marks]

The rules listed above give a small step semantics. List the inference rules which specify an equivalent big step semantics.

## Question 2 [25 Marks]

### A) [10 marks]

In the lecture, we discussed the E-machine as an example of an abstract machine which handles value bindings explicitly by maintaining a value environment. One of the possible return values of the E-machine are function closures.

- i) What is a function closure?
- ii) Give an example of an expression whose evaluation in the E-machine requires the creation of a closure.

### B) [15 marks]

We discussed two distinct methods to handle exceptions: the first method required that, when an exception is thrown, the evaluation unrolls the stack until the matching catch-expression is found. The second method made it possible to directly jump to the matching catch-expression. Describe the second method:

- i) What are the components of the state of the abstract machine?
- ii) How does the state of the machine change when a `catch`-expression is evaluated?
- iii) How does the state of the machine change when a `raise`-expression is evaluated?

For (ii) and (iii), you do not have to give the exact transition rule — it is sufficient to describe how the state is affected.

### Question 3 [25 Marks]

#### A) [6 marks]

For each of the following three pairs of type expressions determine whether the pair has a most general unifier? If so, please provide it.

- i)  $(a, b) \rightarrow (b, a)$  and  $(int, c) \rightarrow (c, c)$
- ii)  $a \rightarrow (a, a)$  and  $(b, b) \rightarrow b$
- iii)  $int \rightarrow int$  and  $float \rightarrow int$

#### B) [9 marks]

Give the principal type of the following (polymorphic) MinML expressions:

- i) `inr(inl(true))`
- ii) `fun f (x) is fst (snd (x)) end`
- iii) `fun g (x) is`  
    `case x of inl(a) -> a`  
        `inr(b) -> b`  
    `end`  
    `end`

#### C) [10 marks]

What is the difference between the function type  $\forall a.(a, a) \rightarrow a$  and the function type  $\exists a.(a, a) \rightarrow a$ ? Assume  $g : \forall a.(a, a) \rightarrow a$  and  $f : \exists a.(a, a) \rightarrow a$ . Give an example each (if it exists) for a concrete value  $v$  such that  $g(v)$  is type correct, and a value  $w$  such that  $f(w)$  is type correct.

#### **Question 4 [25 Marks]**

**A) [10 marks]**

Progress and preservation are central concepts for strongly typed languages.

- i) Give the definition of progress and of preservation in the context of a strongly typed language.
- ii) The presence of partial functions can be problematic with respect to progress. Describe how they can be handled in a strongly typed language such that both progress and preservation still hold.

**B) [5 marks]**

Give an example each for a type constructor which is covariant and a type constructor which is contravariant in at least one of its argument positions.

**C) [10 marks]**

Describe the interaction between dynamic and static typing in Java.