THE UNIVERSITY OF NEW SOUTH WALES

Sample Exam Solutions

Session 2 2014

COMP3161/COMP9161

Concepts of Programming Languages

Markers: answers are provided in bold font. ***Additional information which is not necessarily expected to be part of the student's answer is given in this font.***

Question 1 [25 Marks]

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

Boolean expressions:

$$\frac{}{\texttt{True}\ \textbf{Bool}} \qquad \frac{}{\texttt{False}\ \textbf{Bool}}$$

$$\frac{b_1\ \textbf{Bool} \quad b_2\ \textbf{Bool}}{\texttt{(And}\ b_1\ b_2\texttt{)}\ \textbf{Bool}} \qquad \frac{b\ \textbf{Bool}}{\texttt{(Not}\ b\texttt{)}\ \textbf{Bool}}$$

Evaluation rules:

$$\frac{}{\texttt{(And True}\ b\texttt{)} \mapsto b} \qquad \frac{}{\texttt{(And False}\ b\texttt{)} \mapsto \texttt{False}}$$

$$\frac{}{\texttt{(Not False)} \mapsto \texttt{True}} \qquad \frac{}{\texttt{(Not True)} \mapsto \texttt{False}}$$

$$\frac{b \mapsto b'}{\texttt{(Not}\ b\texttt{)} \mapsto \texttt{(Not}\ b'\texttt{)}} \qquad \frac{b_1 \mapsto b_1'}{\texttt{(And}\ b_1\ b_2\texttt{)} \mapsto \texttt{(And}\ b_1'\ b_2\texttt{)}}$$

A) **[2 marks]**

Give the derivation of the evaluation for the following expression:

- `(And (Not False) (And True (Not True)))`

B) **[3 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) **[4 marks]**

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

D) **[16 marks]**

Give a single step semantics of this language with explicit control stack, adapting the C-machine discussed in the lecture. Start by

i) (3 marks) defining a term representation for a control stack frame

ii) (3 marks) defining a term representation for a control stack

iii) (2 marks) describing what the initial and final states of the machine look like

iv) (8 marks) listing the evaluation rules.

Remember, each of the evaluation rules has to be an axiom.

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)
```
letfun f x = fst (snd x) end
```

iii)
```
letfun g x =
     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) **[5 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]**

Briefly describe the difference between parametric and ad-hoc polymorphism, and give an example function for each.

C) **[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.

D) **[5 marks]**

Why is it important what the variance of a constructor is? Give an example of what can go wrong if a language designer/implementor gets it wrong.

E) **[5 marks]**

In the lecture, we discussed the Software Transactional Memory (STM) approach to control concurrent access to shared data;

i) In contrast to semaphores, STM is said to be an optimistic programming model to control concurrent access to shared data. Why?

ii) How does the type system in Haskell ensure that STM actions are not applied outside of an atomic block?