UNIVERSITY OF SOUTHAMPTON COMP2212W1

SEMESTER 2 EXAMINATION 2020 - 2021

PROGRAMMING LANGUAGE CONCEPTS

DURATION 120 MINS (2 Hours)

This paper contains 5 questions

Answer **four** of the five questions.

An outline marking scheme is shown in brackets to the right of each question.

A reference for the Toy language is included in this examination booklet.

University approved calculators MAY be used. A foreign language translation dictionary (paper version) is permitted provided it contains no notes, additions or annotations

A foreign language dictionary is permitted ONLY IF it is a paper version of a direct 'Word to Word' translation dictionary AND it contains no notes, additions or annotations.

10 page examination paper.

Accompanying Reference Sheet

The syntax of expressions, values and types of the Toy language is

E	$::=$	n	Integer literals
		$\mathbf{true}$	Boolean True literal
		$\mathbf{false}$	Boolean False literal
		$E < E$	Comparison
		$E + E$	Addition
		x	Variable
		if E then E else E	Conditional
		$\lambda(x : T)E$	Abstraction
		let $(x : T) = E$ in E	Let Block
		$E\ E$	Application
V	$::=$	n	Integer value
		$\mathbf{true}$	True value
		$\mathbf{false}$	False value
		$\lambda(x : T)E$	Abstraction value
T	$::=$	Int	Type of Integers
		Bool	Type of Booleans
		$T \to T$	Type of functions

The small step reduction rules are

$$\frac{n < m}{\mathsf{n} < \mathsf{m} \;\to\; \mathbf{true}} \qquad \frac{n \not< m}{\mathsf{n} < \mathsf{m} \;\to\; \mathbf{false}}$$

$$\frac{E \;\to\; E'}{\mathsf{n} < E \;\to\; \mathsf{n} < E'} \qquad \frac{E_1 \;\to\; E_1'}{E_1 < E_2 \;\to\; E_1' < E_2}$$

$$\frac{n + m = n'}{\mathsf{n} + \mathsf{m} \;\to\; \mathsf{n}'} \qquad \frac{E \;\to\; E'}{\mathsf{n} + E \;\to\; \mathsf{n} + E'} \qquad \frac{E_1 \;\to\; E_1'}{E_1 + E_2 \;\to\; E_1' + E_2}$$

$$\frac{}{\mathsf{if}\ \mathbf{true}\ \mathsf{then}\ E_2\ \mathsf{else}\ E_3 \;\to\; E_2} \qquad \frac{}{\mathsf{if}\ \mathbf{false}\ \mathsf{then}\ E_2\ \mathsf{else}\ E_3 \;\to\; E_3}$$

$$\frac{E_1 \;\to\; E_1'}{\mathsf{if}\ E_1\ \mathsf{then}\ E_2\ \mathsf{else}\ E_3 \;\to\; \mathsf{if}\ E_1'\ \mathsf{then}\ E_2\ \mathsf{else}\ E_3}$$

$$\frac{}{\mathsf{let}\,(x : T) \;=\; V\ \mathsf{in}\ E \;\to\; E[V/x]} \qquad \frac{E_1 \;\to\; E_1'}{\mathsf{let}\,(x : T) \;=\; E_1\ \mathsf{in}\ E_2 \;\to\; \mathsf{let}\,(x : T) \;=\; E_1'\ \mathsf{in}\ E_2}$$

$$\frac{}{(\lambda(x : T)E)V \;\to\; E[V/x]} \qquad \frac{E_1 \;\to\; E_1'}{E_1 E_2 \;\to\; E_1' E_2} \qquad \frac{E_2 \;\to\; E_2'}{V E_2 \;\to\; V E_2'}$$

The type rules are

$$\frac{}{\Gamma \vdash \mathsf{n} : \mathsf{Int}} \qquad \frac{}{\Gamma \vdash \mathbf{true} : \mathsf{Bool}} \qquad \frac{}{\Gamma \vdash \mathbf{false} : \mathsf{Bool}}$$

$$\frac{\Gamma \vdash E_1 : \mathsf{Int} \quad \Gamma \vdash E_2 : \mathsf{Int}}{\Gamma \vdash E_1 < E_2 : \mathsf{Bool}} \qquad \frac{\Gamma \vdash E_1 : \mathsf{Int} \quad \Gamma \vdash E_2 : \mathsf{Int}}{\Gamma \vdash E_1 + E_2 : \mathsf{Int}} \qquad \frac{x : T \in \Gamma}{\Gamma \vdash x : T}$$

$$\frac{\Gamma \vdash E_1 : \mathsf{Bool} \quad \Gamma \vdash E_2 : T \quad \Gamma \vdash E_3 : T}{\Gamma \vdash \mathsf{if}\ E_1\ \mathsf{then}\ E_2\ \mathsf{else}\ E_3 : T} \qquad \frac{\Gamma \vdash E_1 : T_1 \quad \Gamma, x : T_1 \vdash E_2 : T_2}{\Gamma \vdash \mathsf{let}\,(x : T_1) \;=\; E_1\ \mathsf{in}\ E_2 : T_2}$$

$$\frac{\Gamma, x : T_1 \vdash E : T_2}{\Gamma \vdash \lambda(x : T_1)E : T_1 \to T_2} \qquad \frac{\Gamma \vdash E_1 : T_2 \to T_1 \quad \Gamma \vdash E_2 : T_2}{\Gamma \vdash E_1 E_2 : T_1}$$

Where Γ is a type context given by the grammar $\Gamma ::= \emptyset \mid \Gamma, x : T$

TURN OVER

Question 1.

(a) Consider the simple BNF specification below:

$$\begin{aligned} \texttt{<ifstm>} &::= \mathbf{if}\ (\texttt{<boolexp>})\ \texttt{<stmt>} \\ &\mid \mathbf{if}\ (\texttt{<boolexp>})\ \texttt{<stmt>}\ \mathbf{else}\ \texttt{<stmt>} \\ \texttt{<boolexp>} &::= \mathbf{true} \mid \mathbf{false} \\ \texttt{<stmt>} &::= \texttt{<ifstm>} \mid \mathbf{skip} \end{aligned}$$

For each of the following, state whether or not the string is generated by the grammar (assuming top level non-terminal <stmt>).

(i) **if** (**true**) **skip else false**

(ii) **if** (**true**) **if** (**false**) **else skip**

(iii) **if** (**true**) **if** (**false**) **skip else skip else skip**

(iv) **if** (**true**) **if** (**true**) **skip else if** (**true**) **if** (**true**) **if** (**true**) **skip**

[4 marks]

(b) Explain what it means for a BNF specification to be *ambiguous*. Why is ambiguity considered to be problematic in programming language design?

[5 marks]

(c) With the aid of an example, show that the BNF specification given in Part (a) is ambiguous. [5 marks]

(d) Extend the grammar to obtain a more realistic set of boolean expressions. In particular, expressions can contain boolean variables, binary logical operations $\wedge$ and $\rightarrow$, and a unary operation $\neg$. Design your grammar so that $\wedge$ and $\rightarrow$ both associate to the right (i.e. $a \wedge b \wedge c$ should mean $a \wedge (b \wedge c)$ and $a \rightarrow b \rightarrow c$ should mean $a \rightarrow (b \rightarrow c)$.

[11 marks]

Question 2.

This question is based on the Toy language described in the Accompanying Reference Sheet. We will also work with an extension of the Toy language named *ToyFix* described as follows: We extend the grammar of expressions with an expression fix E and include the following type and reduction rules to the type system and small step call-by-value semantics:

$$\frac{\Gamma \vdash E : T \to T}{\Gamma \vdash \text{fix } E : T} \qquad \frac{E \;\to\; E'}{\text{fix } E \;\to\; \text{fix } E'} \qquad \frac{}{\text{fix } \lambda(x : T)E \;\to\; E[\text{fix } \lambda(x : T)E/x]}$$

(a) The small step operational semantics provided for the Toy language are for a call-by-value semantics. Write a big step operational semantics for *ToyFix* by providing semantics for the fix operator that could be used with the conventional big-step call-by-value semantics for Toy. You may assume the big step operational semantics for Toy as provided in the lecture notes.

[4 marks]

(b) For any closed expression E of the *ToyFix* language, Is it true that this expression will evaluate to some value V in the small step semantics **if and only if** it evaluates to the same value V in the big step semantics defined in Part (a)? Justify your answer by providing a counter-example or an informal argument as to why this is the case.

[5 marks]

(c) Let B be the expression

$$\text{if } x < n \text{ then } f\ (x + x) \text{ else } x$$

and let F be the expression

$$\lambda(n : \mathsf{Int})\lambda(f : \mathsf{Int} \to \mathsf{Int})\lambda(x : \mathsf{Int})B$$

Assuming that $\Gamma \vdash B : \mathsf{Int}$ where Γ is

$$n : \mathsf{Int}, f : \mathsf{Int} \to \mathsf{Int}, x : \mathsf{Int}$$

has a valid type derivation, give a type derivation for the expression fix $(F\ 3)$ in the ToyFix language.

TURN OVER

[6 marks]

(d) Show the sequence of reductions of the expression $(\text{fix } (F\ 3))\ 2$, using the small step call-by-value operational semantics of the ToyFix language. Continue the reduction sequence until a value is reached.

[6 marks]

(e) Show the evaluation of the expression $(\text{fix } (F\ 3))\ 2$, using the big step operational semantics of the ToyFix language as given in Part (a).

[4 marks]

Question 3.

In this question we consider a typed programming language, Mat, given by the following grammar:

$$E, F ::= x \mid M \mid \text{toVec } E \mid E^T \mid E + E \mid E\ E \mid E \otimes E \mid \text{let } x\ = E \text{ in } E$$
$$M ::= (V, V, \ldots, V) \quad \text{Matrix Literals}$$
$$V ::= (v, v, \ldots, v) \quad \text{Column Vectors}$$
$$T ::= \text{Mat } n \times n \mid \text{Vec } n$$

The type Mat $n \times m$ represents a matrix with n rows and m columns and Vec n represents a column vector of dimension n. Note that x is drawn from a set of variables, n is drawn from the set of natural numbers and v ranges over integers. The intended semantics is that this is a language of matrices. The operation $+$ represents matrix addition, E^T represents matrix transpose, juxtaposition (e.g. $M_1 M_2$) represents multiplication of the matrix M_1 by the matrix M_2, and $\otimes$ represents outer product. Finally, toVec E takes a matrix containing a single column and treats it as a column vector.

(a) Explain what is meant by an *inductively defined typing relation* for a programming language.

[5 marks]

(b) Explain what is meant by *type inference* for a programming language and indicate why it is feasible to perform type inference for Mat.

[5 marks]

(c) Give an inductively defined typing relation for Mat that respects the dimensions of the matrix operations. Recall that addition is done between matrices of the same dimensions. Transpose switches the number of rows and columns. Matrix multiplication takes matrices of dimensions $n \times k$ and $k \times m$ and returns matrices of dimensions $n \times m$. Outer product takes two column vectors V and W (of possibly different dimension) and is defined as $V \otimes W = V\ W^T$. [15 marks]

TURN OVER

Question 4.

(a) Explain the main difference between *deadlock* and *livelock*. [3 marks]

(b) Explain why the following code is not thread-safe. Use the line numbers of the code to refer to specific points of execution.

```
1  public class Foo {
2
3  private Foo instance = null;
4
5  public Foo getInstance() {
6      if(instance == null) {
7      synchronized (Foo.class) {
8          if(instance == null) {
9           instance = new Foo();
10             }
11         }
12     }
13     return instance;
14     }
15 }
```

[5 marks]

(c) Suggest a modification to the above code in order to make it thread-safe and explain why it would do so.

[4 marks]

(d) Explain how `compareAndSet` works.

[3 marks]

(e) For each call to `compareAndSet` in the following code, devise a scenario where the call fails (i.e. returns false). The poll operation retrieves and removes the head of this queue, or returns null if this queue is empty. Use the line numbers of the code to refer to specific points of execution.

```
1  import java.util.concurrent.atomic.*;
2
3  public class LinkedQueue <E> {
4
5    private static class Node <E> {
6      final E item;
7      final AtomicReference<LinkedQueue.Node<E>> next;
8
9      public Node(E item, LinkedQueue.Node<E> next) {
10       this.item = item;
11       this.next = new AtomicReference<LinkedQueue.Node<E>>(next); }}
12
13   private final LinkedQueue.Node<E> dummy
14      = new LinkedQueue.Node<E>(null, null);
15   private final AtomicReference<LinkedQueue.Node<E>> head
16      = new AtomicReference<LinkedQueue.Node<E>>(dummy);
17   private final AtomicReference<LinkedQueue.Node<E>> tail
18      = new AtomicReference<LinkedQueue.Node<E>>(dummy);
19 // Code continues below
20
```

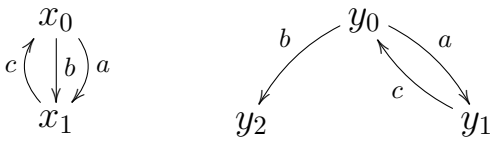
```
21
22 // Code for poll method
23       public final T poll() {
24         while (true) {
25             Node<T> currentHead = head.get();
26             Node<T> currentTail = tail.get();
27             Node<T> next = currentHead.next.get();
28             if (currentHead == head.get()) {
29                 if (currentHead == currentTail) {
30                     if (next == null) {
31                         return null;
32                     } else {
33                         tail.compareAndSet(currentTail, next);
34                     }
35                 } else {
36                     if (head.compareAndSet(currentHead, next)) {
37                         T item = next.item;
38                         next.item = null;
39                         return item;
40                         }
41                 }
42             }
43         }
44       }
45
46  // Code for put method omitted
47
48 }
```

[10 marks]

TURN OVER

Question 5.

Consider the following labelled transition systems



(a) List all the traces that originate at x_0. [2 marks]

(b) Are the nodes y_0 and x_0 trace equivalent? Justify your answer. [2 marks]

(c) Show that x_0 simulates y_0. [4 marks]

(d) Show that y_0 does not simulate x_0. [4 marks]

(e) Are x_0 and y_0 bisimilar? If yes then prove this, otherwise add a single labelled edge to one of the transition systems given and prove that x_0 and y_0 are bisimilar in the amended systems. [13 marks]

END OF PAPER