Candidate Number

G5029

THE UNIVERSITY OF SUSSEX

BSc and MComp FINAL YEAR EXAMINATION
May/June 2018 (A2)

Limits of Computation

Assessment Period: May/June 2018 (A2)

DO NOT TURN OVER UNTIL INSTRUCTED TO BY THE LEAD INVIGILATOR

Candidates should answer TWO questions out of THREE. If all three questions are attempted only the first two answers will be marked.

The time allowed is TWO hours.

Each question is worth 50 marks.

At the end of the examination the question paper and any answer books/answer sheets, used or unused, will be collected from you before you leave the examination room.

1. This question is about `SRAM`, `WHILE` and other notions of effective computabili[illegible]

   (a) Descri[illegible]he judgment

   $$\texttt{p} \vdash (\ell, \sigma) \to (\ell', \sigma')$$

   means [illegible]m $\texttt{p}$. Recall that ℓ and ℓ' are program labels and that σ [illegible]hat map natural numbers to natural numbers. [5 marks]

   (b) Let a specific `SRAM` program $\texttt{p}$ and a label ℓ be given such that $\texttt{p}(\ell) =$ `Xi:=<Xj>`. Let σ be a store. Describe precisely what ℓ' and σ' must be if

   $$\texttt{p} \vdash (\ell, \sigma) \to (\ell', \sigma')$$

   holds for $\texttt{p}$, ℓ and σ as described above. [5 marks]

   (c) Do the computability results we proved for the `WHILE`-language also hold for `SRAM`? Explain your answer in one sentence. [4 marks]

   (d) The semantics of a `WHILE` command C is expressed by the judgment

   $$C \vdash \sigma \to \sigma'$$

   where σ and σ' are stores for `WHILE`. For instance, $[X \mapsto \ulcorner 2 \urcorner, Y \mapsto \texttt{nil}]$ describes a store that maps program variable X to value $\ulcorner 2 \urcorner$ and variable Y to value `nil`. Recall that $\ulcorner n \urcorner$ is the encoding of number n as `WHILE`-data.

      i. Give an example of a *single* `WHILE` statement C such that

      $$C \vdash [X \mapsto \texttt{nil}] \to [X \mapsto \ulcorner 2 \urcorner]$$

      [4 marks]

      ii. Give an example of a *single* `WHILE` statement C such that there is no store σ with $C \vdash [X \mapsto \texttt{nil}] \to \sigma$ [4 marks]

      iii. Compute the value of $\mathcal{E}[\![\texttt{cons tl X nil}]\!][X \mapsto \ulcorner 3 \urcorner]$. Present the final result as a list that contains only natural numbers. *Show your working.* [6 marks]

   (e) Assuming that we start counting variables from 0, give the program-as-data representation of the following `WHILE` program:

```
prog read X {
       X:= tl X;
       if Y { Y:= cons X Y }
       else { X:= tl X ; Y:= nil }
       }
write Y
```

   [10 marks]

(f) Consider the language `REPEAT` which has the same program structure and ex[illegible] `WHILE`-language but whose commands do not contai[illegible]t but instead a statement of the form:

$$\texttt{repeat}\ E\ \texttt{times}\ B$$

where [illegible]sents expressions and B is a block of the form $\{C_1;..$[illegible] and expressions in `REPEAT` are exactly as in `WHILE`, and therefore the stores used in the semantics of `REPEAT` are like the ones used in `WHILE`. Also the semantics of assignment and sequential composition in `REPEAT` are exactly as they are in `WHILE`.

Assuming that `E` evaluates to $\ulcorner n \urcorner$, executing `repeat` E `times` B means executing its body B exactly n times (if $n = 0$, B is not executed at all). If, however, E does not evaluate to a natural number, the repeat statement terminates immediately.

Consider the `REPEAT` program `test`:

```
test read X {
  Y := nil;
  repeat X times { Y:= cons nil Y }
  }
write Y
```


For input $\ulcorner 0 \urcorner$ the program `test` returns $\ulcorner 0 \urcorner$, for input $\ulcorner 1 \urcorner$ it returns $\ulcorner 1 \urcorner$, for input $\ulcorner 2 \urcorner$ it returns $\ulcorner 2 \urcorner$, and so on.

Here is the formal semantics of the repeat statement for the case that the expression argument does not evaluate to a natural number:

$$\texttt{repeat E times C} \vdash \sigma \to \sigma \text{ if there is no } k \in \mathbb{N} \text{ such that } \mathcal{E}\,[\![\mathtt{E}]\!]\,\sigma = \ulcorner k \urcorner$$

Complete the formal semantics of the `repeat` statement by giving the missing clause(s). [8 marks]

(g) Can we program a self-interpreter for language `REPEAT` like we have done for `WHILE`? Explain your answer *briefly*. [4 marks]

2. This question is about semidecidability and decidability.

In this ques[illegible]rogramming language L. For any L-program p, the semant[illegible] the semantic function for L:

$[\![\mathtt{p}]\!]^{\mathtt{L}} : \mathtt{L}$–da[illegible]

(a) Refer[illegible] function for L, explain what it means for a set $A \subseteq$ L-[illegible]decidable". [6 marks]

(b) Give [illegible] (without explanation) of a problem that is semi-decidable but not decidable. [2 marks]

(c) Without having to know what L is exactly (we assume only it is not trivial), we can always give <u>TWO</u> examples of L-decidable problems. Which are they? (No explanation required) [4 marks]

(d) Assume f is a function of type L-data $\rightarrow$ L-data$_\perp$. Which minimal assumptions about f do we have to make to ensure that the following statement is correct:
"$\{x \in$ L-data $\mid f(x) = x\}$ is L-decidable."
With your chosen assumptions argue briefly why the statement holds. [7 marks]

(e) Assume now L is WHILE. Which of the following problems are WHILE-undecidable? (No explanation needed.)

i. Halting problem for WHILE-programs
ii. Complement of the Travelling Salesman problem
iii. The problem whether two given WHILE program have the same semantics
iv. Shortest Path problem
v. Tiling problem
vi. Matching Problem

[6 marks]

(f) Decide for the following sets $A \subseteq$ WHILE-data whether they are WHILE-decidable. For each case, explain your answer. In cases where A is decidable, this explanation should consist of a description of the decision procedure. Recall that $\ulcorner n \urcorner$ is the encoding of number n as WHILE-data and $\ulcorner p \urcorner$ the encoding of a WHILE-program as WHILE-data.

i. $A = \{\ulcorner p \urcorner \mid p$ is a WHILE-program and $\ulcorner p \urcorner$ is a list of length $13\ \}$ [6 marks]

ii. $A = \{d \mid$ WHILE-program Prog halts when run on input $d\}$

where Prog is a WHILE-program with semantics

$$[\![\mathtt{Prog}]\!]^{\mathtt{WHILE}}(d) = \begin{cases} \ulcorner m - n \urcorner & \text{if } d = (\ulcorner m \urcorner . \ulcorner n \urcorner) \text{ and } m \geq n, \\ \ulcorner 0 \urcorner & \text{if } d = (\ulcorner m \urcorner . \ulcorner n \urcorner) \text{ and } m < n, \\ \text{undefined} & \text{otherwise} \end{cases}$$

[6 marks]

iii. $A = \{ \ulcorner p \urcorner \mid \texttt{WHILE}\text{-program } p \text{ returns } \ulcorner p \urcorner \text{ for any input } \}$ [6 marks]

(g) Briefly [illegible]act Rice's Theorem has on programming langua[illegible]opment environments (tools) for programmers. [7 marks]

3. This question is about complexity.

   (a) Define [illegible]ss $\mathbf{P}^{\texttt{WHILE}}$. [4 marks]

   (b) For ea[illegible]ts below state whether it is known to be true, known [illegible]ether it is unknown whether it is true or false. You d[illegible]easons for your answer.

      1 The [illegible]P-complete.

      2 SAT (Cook's Satisfiability Problem) can be reduced to TSP (the Traveling Salesman Problem) in polynomial time.

      3 $\mathbf{P}^{\text{TM}} = \mathbf{P}^{\text{RAM}}$

      4 TSP (the Traveling Salesman Problem) can be reduced to SAT (Cook's Satisfiability Problem) in polynomial time.

      5 TSP (the Traveling Salesman Problem) can be reduced to the Postman problem in polynomial time.

      6 $\mathbf{P} \subseteq \mathbf{NP}$.

      7 The Postman problem is in $\mathbf{P}$.

      8 TSP (the Traveling Salesman Problem) is in $\mathbf{P}$.

      9 Quantum Computers can solve **NP**-complete problems in polynomial time.

   [9 marks]

   (c) Explain what the *Cook-Karp* (also known as *Cobham-Edmonds*) thesis says. What are the arguments for and against this thesis? [8 marks]

   (d) Consider the following optimisation problem which is a minimisation problem:

   Bin Packing Opt: Assume we have n items, called $1, \ldots, n$, each of which has a nonnegative integer size, called a_i. Assume further we have an arbitrary number of 'bins' of capacity k, which means that each bin can hold items only if the sum of the size of those items is less or equal k. Minimise the number of bins that can hold all items $1, \ldots, n$.

   For instance, if $n = 3$ with $a_1 = 4$, $a_2 = 3$ and $a_3 = 9$ and $k = 10$ then the minimal number of bins is 2. The first bin, for example, can hold item 3 but nothing else as $a_3 = 9 \leq 10$, and the second bin can hold items 1 and 2, as $a_1 + a_2 = 7 \leq 10$.

   All numbers a_i and k are represented in binary format.

   i. In general, what condition must an algorithm for a minimisation problem satisfy to be called an α-approximation algorithm ? Your explanation should include a definition of the *approximation factor* of such an algorithm. [4 marks]

   ii. Sketch a simple polynomial approximation algorithm for the optimisation problem **Bin Packing Opt** above with an approximation factor of 2. *Hint: it's not difficult to get this factor (it's almost automatic) and you don't have to prove it. To make sure your* *algorithm achieves this factor just ensure that it is never the case th[illegible]at are less than half full.* [6 marks]

iii. In [illegible]n problem class must **Bin Packing Opt** be ac[illegible]e? Briefly explain why. [4 marks]

(e) Consi[illegible]*n problem* version of **Bin Packing**:

Bin Pa[illegible][illegible]e have n items, called $1, \ldots, n$, each of which has a [illegible]r size, called a_i. Is it possible to place all items $1, \ldots, n$ in M bins that all have the same capacity k, i.e. each bin can only hold items the sum of their size is less or equal k? We can assume here that $M < n$ otherwise the problem is trivial.

All numbers a_i, M and k are represented in binary format.

Explain why **Bin Packing** (the decision problem!) is in **NP**. Any algorithm to show this can be sketched, but any runtime information must be discussed thoroughly. [9 marks]

(f) What important result would follow if we could prove that $\mathbf{NP}^{\texttt{WHILE}}$ was *not* closed under complement? Explain your answer briefly. [6 marks]