

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)

Assignment Project Exam Help

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 computability.

- (a) Describe precisely what the judgment

$$p \vdash (\ell, \sigma) \rightarrow (\ell', \sigma')$$

means for an SRAM program  $p$ . Recall that  $\ell$  and  $\ell'$  are program labels and that  $\sigma$  and  $\sigma'$  are stores that map natural numbers to natural numbers.

[5 marks]

- (b) Let a specific SRAM program  $p$  and a label  $\ell$  be given such that  $p(\ell) = X_i := \langle X_j \rangle$ . Let  $\sigma$  be a store. Describe precisely what  $\ell'$  and  $\sigma'$  must be if

$$p \vdash (\ell, \sigma) \rightarrow (\ell', \sigma')$$

holds for  $p$ ,  $\ell$  and  $\sigma$  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 judgement

$$C \vdash \sigma \rightarrow \sigma'$$

where  $\sigma$  and  $\sigma'$  are stores for WHILE. For instance,  $[X \mapsto \ulcorner 2 \urcorner, Y \mapsto \text{nil}]$  describes a store that maps program variable  $X$  to value  $\ulcorner 2 \urcorner$  and variable  $Y$  to value  $\text{nil}$ . 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

$$C \vdash [X \mapsto \text{nil}] \rightarrow [X \mapsto \ulcorner 2 \urcorner]$$

[4 marks]

- ii. Give an example of a *single* WHILE statement  $C$  such that there is no store  $\sigma$  with  $C \vdash [X \mapsto \text{nil}] \rightarrow \sigma$

[4 marks]

- iii. Compute the value of  $\mathcal{E} \llbracket \text{cons } t1 \ X \ \text{nil} \rrbracket [X \mapsto \ulcorner 3 \urcorner]$ . 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 := t1 X;  
  if Y { Y := cons X Y }  
  else { X := t1 X ; Y := nil }  
}  
write Y
```

[10 marks]

- (f) Consider the language REPEAT which has the same program structure and expressions  $E$  as the WHILE-language but whose commands do not contain a `while` statement but instead a statement of the form:

`repeat  $E$  times  $B$`

where, as usual,  $E$  represents expressions and  $B$  is a block of the form  $\{C_1; \dots; C_n\}$ . Variables 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  $\lceil n \rceil$ , 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  $\lceil 0 \rceil$  the program `test` returns  $\lceil 0 \rceil$ , for input  $\lceil 1 \rceil$  it returns  $\lceil 1 \rceil$ , for input  $\lceil 2 \rceil$  it returns  $\lceil 2 \rceil$ , 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:

`repeat  $E$  times  $C \vdash \sigma \rightarrow \sigma$  if there is no  $k \in \mathbb{N}$  such that  $\mathcal{E} \llbracket E \rrbracket \sigma = \lceil k \rceil$`

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 question we refer to a programming language  $L$ . For any  $L$ -program  $p$ , the semantics of  $p$  is given by the semantic function for  $L$ :

$$\llbracket p \rrbracket^L : L\text{-data} \rightarrow L\text{-data}_\perp$$

- (a) Referring to the semantic function for  $L$ , explain what it means for a set  $A \subseteq L\text{-data}$  to be “ $L$ -semidecidable”. [6 marks]
- (b) Give an example (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 TWO examples of  $L$ -decidable problems. Which are they? (No explanation required) [4 marks]

- (d) Assume  $f$  is a function of type  $L\text{-data} \rightarrow L\text{-data}_\perp$ . Which minimal assumptions about  $f$  do we have to make to ensure that the following statement is correct:

“ $\{x \in L\text{-data} \mid f(x) = x\}$  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 programs have the same semantics
- iv. Shortest Path problem
- v. Tiling problem
- vi. Matching Problem

[6 marks]

- (f) Decide for the following sets  $A \subseteq \text{WHILE-data}$  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.  $A = \{\ulcorner p \urcorner \mid p \text{ is a WHILE-program and } \ulcorner p \urcorner \text{ is a list of length } 13\}$

[6 marks]

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

where Prog is a WHILE-program with semantics

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

[6 marks]

- iii.  $A = \{\lceil p \rceil \mid \text{WHILE-program } p \text{ returns } \lceil p \rceil \text{ for any input } \}$  [6 marks]
- (g) Briefly explain the impact Rice's Theorem has on programming languages and their development environments (tools) for programmers. [7 marks]

Assignment Project Exam Help

<https://tutorcs.com>

WeChat: cstutorcs

3. This question is about complexity.

(a) Define the complexity class  $\mathbf{P}^{\text{WHILE}}$ . [4 marks]

(b) For each of the statements below state whether it is known to be true, known to be false, or whether it is unknown whether it is true or false. You do *not* have to give reasons for your answer.

1 The Tiling problem is **NP**-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.

<https://tutorcs.com>

[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  $1, \dots, n$ , 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  $1, \dots, n$ .

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  $\alpha$ -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 that two bins exist that are less than half full.* [6 marks]

- iii. In which optimisation problem class must **Bin Packing Opt** be according to the above? Briefly explain why. [4 marks]

- (e) Consider now the *decision problem* version of **Bin Packing**:

**Bin Packing**: Assume we have  $n$  items, called  $1, \dots, n$ , each of which has a nonnegative integer size, called  $a_i$ . Is it possible to place all items  $1, \dots, n$  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 **NP<sup>WHILE</sup>** was *not* closed under complement? Explain your answer briefly. [6 marks]

Assignment Project Exam Help

<https://tutorcs.com>

WeChat: cstutorcs