Theories of Computation
Solutions

Main Summer Examinations 2024

Note

Answer ALL questions. Each question will be marked out of 20. The paper will be marked out of 60, which will be rescaled to a mark out of 100.

Question 1

(a) Consider the following context-free grammar for simple Java methods:

⇒	Method	::= Visibility Modifier Type Word (Type Word) { Operation ; }
	Visibility	::= `public` | `private`
	Modifier	::= ε | `static` | `abstract`
	Type	::= `void` | `int` | `Boolean` | `String` | Type`[]`
	Word	::= [a-z] | Word Word
	Number	::= [0-9] | Number Number
	Value	::= "Word" | Number | Number + Number
	Operation	::= `print(`Value`)` | `return` Value

(i) For both of the following two simple Java methods, state whether or not they are generated by the above grammar (ignoring whitespace such as spaces).

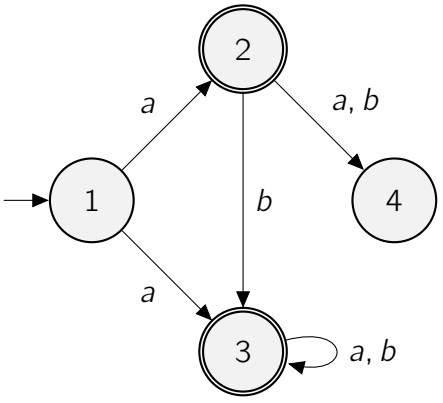
i. `private abstract int add5(int x) { return x + 5; }` **[1 mark]**

ii. `public static void main(String[] args) { print("hi"); }` **[1 mark]**

(ii) Write the leftmost derivation of the following simple Java method. At each stage, write a dot over the variable that will be replaced in the next step.

`public String ln(Boolean x) { return 0; }` **[4 marks]**

(b) Consider the following nondeterministic finite automaton (NFA), which is defined on the alphabet $\Sigma = \{a, b\}$:



(i) Write a regular expression for the language of this NFA. **[3 marks]**

(ii) Draw the partial DFA that results from determinising the NFA. **[3 marks]**

(iii) Under what conditions is a total DFA said to be *minimal*? **[3 marks]**

(c) Adira has written a sorting program A, which sorts any list of natural numbers. Let $T(n)$ be the time in milliseconds that A takes to sort a list of length $n \in \mathbb{N}$. Adira finds that:

- Sorting the empty list always takes 1 millisecond,
- Sorting any list of one element always takes 2 milliseconds,
- For any $n \in \mathbb{N}$, sorting any list of length $n+2$ always takes $2T(n) + T(n+1)$ milliseconds.

Using course-of-values induction, prove that when asked to sort any list of length $n \in \mathbb{N}$ the program A always takes 2^n milliseconds. **[5 marks]**

Model answers:

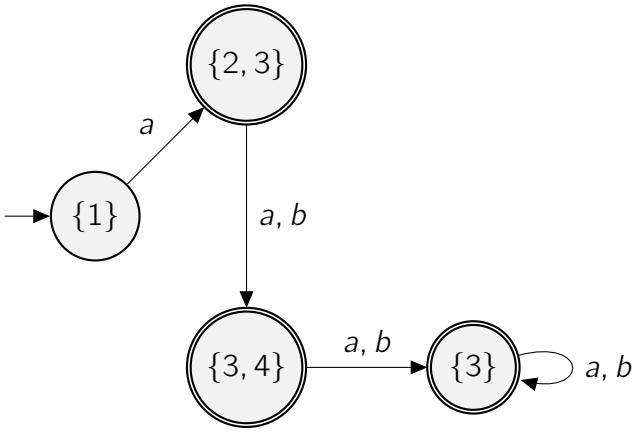
(a) (i) i. No.
ii. Yes.

(ii)

$$
\begin{aligned}
\text{Method} &\rightsquigarrow \text{Visibility Modifier Type Word (Type Word) \{ Operation ; \}} \\
&\rightsquigarrow \texttt{public Modifier Type Word (Type Word) \{ Operation ; \}} \\
&\rightsquigarrow \texttt{public Type Word (Type Word) \{ Operation ; \}} \\
&\rightsquigarrow \texttt{public String Word (Type Word) \{ Operation ; \}} \\
&\rightsquigarrow \texttt{public String Word Word (Type Word) \{ Operation ; \}} \\
&\rightsquigarrow \texttt{public String lWord (Type Word) \{ Operation ; \}} \\
&\rightsquigarrow \texttt{public String ln (Type Word) \{ Operation ; \}} \\
&\rightsquigarrow \texttt{public String ln (Boolean Word) \{ Operation ; \}} \\
&\rightsquigarrow \texttt{public String ln (Boolean x) \{ Operation ; \}} \\
&\rightsquigarrow \texttt{public String ln (Boolean x) \{ return Value ; \}} \\
&\rightsquigarrow \texttt{public String ln (Boolean x) \{ return Number ; \}} \\
&\rightsquigarrow \texttt{public String ln (Boolean x) \{ return 0 ; \}}
\end{aligned}
$$

(b) (i) $\texttt{a}(\varepsilon \mid \texttt{b})(\texttt{a} \mid \texttt{b})^*$

(ii)



(iii) A total DFA is said to be *minimal* when it has the following two properties:

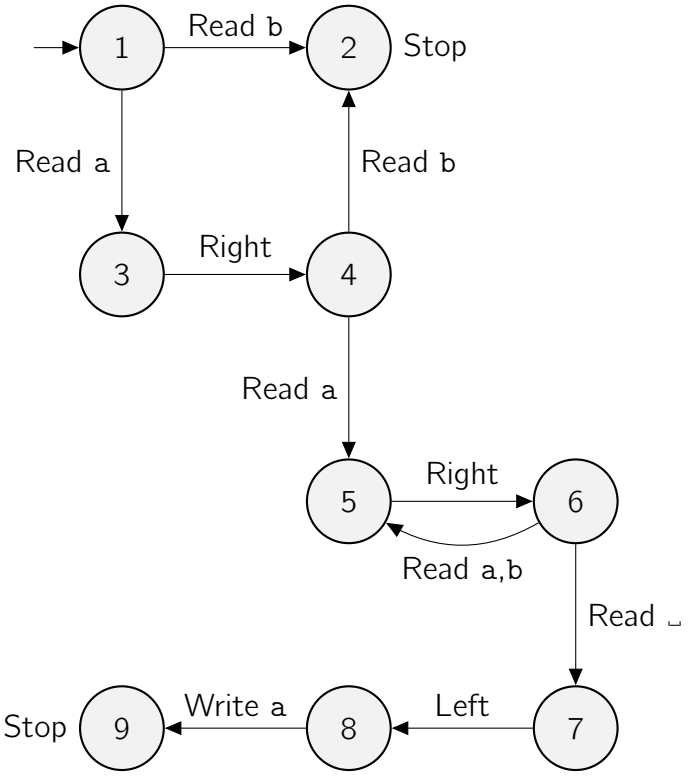
- Each of its states x is *reachable*. This means that there's a path from the initial state to x.
- Any two distinct states x, y are *inequivalent*. This means that there's a word that x accepts (i.e., that leads from x to an accepting state) but y rejects, or vice versa.

(c) Students should recognise that $T(0) = 1$ and $T(1) = 2$. From these recognitions, they can conclude the two base cases of this inductive proof: $T(0) = 1 = 2^0$ and $T(1) = 2 = 2^1$.

The inductive case requires showing that $T(n+2) = 2^{n+2}$. We already have that $T(n+2) = 2T(n) + T(n+1)$. By course-of-values induction, because $n < n+2$ and $n+1 < n+2$, we therefore have $T(n+2) = 2(2^n) + 2^{n+1} = 2^{n+1} + 2^{n+1} = 2(2^{n+1}) = 2^{n+2}$, as required.

Question 2

Consider the following Turing machine $\mathcal{M}$ on alphabet $\Omega = \{\mathtt{a}, \mathtt{b}, \sqcup\}$ with return value set $V = \{\mathtt{Done}\}$.



(a) Give the complete run of machine $\mathcal{M}$ above on the word aab, with the head positioned on the leftmost a.

At each step, indicate the tape contents, the position of the head, the current state and the instruction (including the result if it is a Read). **[6 marks]**

Hint: No more than 12 steps are required.

(b) Now suppose the tape initially contains a block of a's and b's of length $n \geq 2$, on an otherwise blank tape, with the head on the leftmost non-blank character. State, with justification, the number of steps (including Stop) required in:

(i) The best case. **[2 marks]**

(ii) The worst case. **[2 marks]**

(iii) The average case. **[4 marks]**

Note: Assume a and b are equally likely and the characters of the input are independent.

(c) Given two programs A and B, we build the program AB which first runs A and then B on the same input of length n. The worst case running time of A is given by a

function $T(n)$, which is $n^7 + 5$ for $n < 1000$ and $n^3 + 5$ for $n \geq 1000$. The worst case running time of B is given by a function $R(n)$, which is $O(n \log_{10} n)$. Show carefully that the worst case running time of AB is $O(n^3)$. **[6 marks]**

Note: You may assume that $\log_{10} n < n$ for all $n > 0$.

Model answers:

(a)

$\dot{a}ab$	1	Read a
$\dot{a}ab$	3	Right
$a\dot{a}b$	4	Read a
$a\dot{a}b$	5	Right
$aa\dot{b}$	6	Read b
$aa\dot{b}$	5	Right
$aab\dot{\sqcup}$	6	Read $\sqcup$
$aab\dot{\sqcup}$	7	Left
$aa\dot{b}$	8	Write a
$aa\dot{a}$	9	Stop

(b) (i) The best case, realised by b^n, is 2 steps (Read b, Stop).

(ii) The worst case, realised by $aa(a|b)^{n-2}$, consists of 4 steps (Read a, Right, Read a, Right), then $n-2$ cycles of 2 steps (Read (a, b), Right), then 4 steps (Read $\sqcup$, Left, Write a, Stop). In total $4 + 2(n-2) + 4 = 2n + 4$.

(iii) If the input begins ba or bb, the time is 2 steps. If the input begins ab, the time is 4 steps (Read a, Right, Read b, Stop). If the input begins aa, the time is $2n+4$ steps. So the average number of steps is $\frac{1}{4}(2+2+4+2n+4) = \frac{1}{2}n+3$.

(c) There is C and M such that for all $n \geq M$, we have $R(n) \leqslant Cn\log_{10} n$. Let M' be the maximum of M and 1000. For all $n \geq M'$, the running time of AB on an input of length n is

$$\begin{aligned} T(n) + R(n) &\leq n^3 + 5 + Cn\log_{10} n \\ &\leq n^3 + 5n^3 + Cn^3 \\ &= (6 + C)n^3 \end{aligned}$$

so $T(n) + R(n)$ is $O(n^3)$.

Question 3

(a) What does it mean for a property of words to be in NP? **[5 marks]**

(b) The *factorisation* decision problem is as follows: given two integers $a, b > 1$, written in binary, say whether a has a factor less than b. Explain informally why this problem is in NP. **[5 marks]**

(c) Andrew is the manager of GrammarMe Ltd. Every Monday, he receives two context free grammars from the software team and has to check that precisely one of them is ambiguous (he does not need to specify which one).

(i) Can he write a Java program to do this? Explain your answer. (You may use the fact that ambiguity of context free grammars is an undecidable property.) **[6 marks]**

(ii) What if Andrew uses a different programming language? **[4 marks]**

Model answers:

(a) There is a nondeterministic Turing machine that runs in polynomial time (i.e., there is M, C, k such that for any input of length $n \geq M$, all runs take at most Cn^k steps) and an input is acceptable iff it has the property.

(b) Consider a nondeterministic Turing machine which first chooses any number x less than b, and then checks whether x is a factor of a. Thus acceptance is possible iff a has a factor less than b. Writing n for the total length of the two numbers written in binary, then the first task takes $O(n)$ steps and the second takes $O(n^2)$, so the total is $O(n^2)$, which is polynomial.

(c) (i) No it cannot. To see this, let G_0 be the unambiguous grammar

$$\Rightarrow S ::= \varepsilon$$

whose language is empty. If the task in the question could be done by a program P, we could decide ambiguity of a context free gramma G by applying P to G and G_0, as this would return True iff G is ambiguous.

(ii) Church's thesis says that no programming language can decide more properties on words than a Turing machine, or equivalently, than Java. So the answer is still no.