The University of Nottingham

SCHOOL OF COMPUTER SCIENCE

A LEVEL 2 MODULE, SPRING SEMESTER 2015–2016

MACHINES AND THEIR LANGUAGES

ANSWERS

Time allowed TWO hours

Candidates may complete the front cover of their answer book and sign their desk card but must NOT write anything else until the start of the examination period is announced.

Answer ALL THREE questions

No calculators are permitted in this examination.

Dictionaries are not allowed with one exception. Those whose first language is not English may use a standard translation dictionary to translate between that language and English provided that neither language is the subject of this examination. Subject-specific translation directories are not permitted.

No electronic devices capable of storing and retrieving text, including electronic dictionaries, may be used.

Note: ANSWERS

Question 1
The following questions are multiple choice. There is at least one correct answer, but there may be several. To get all the marks you have to list all correct answers and none of the incorrect ones. 1 mistake results in 3 marks, 2 mistakes result in 1 mark, 3 or more mistakes result in zero marks.

Answer: *Note that the answer that should be provided is just a list of the correct alternative(s). The additional explanations below are just for your information.*

(a) Which of the following statements are correct?

(i) The empty word ϵ is the only word in the empty language $\emptyset$.

(ii) $\epsilon \in \Sigma^*$, where $\Sigma = \{0, 1\}$

(iii) $\epsilon \notin \emptyset^*$

(iv) The empty word ϵ does not belong to any non-empty language.

(v) $\epsilon \in \{a^i b^j \mid i, j \in \mathbb{N}, i + j \leq 42\}$ (where $\mathbb{N} = \{0, 1, 2, \ldots\}$) (5)

Answer: *Correct: ii, v*

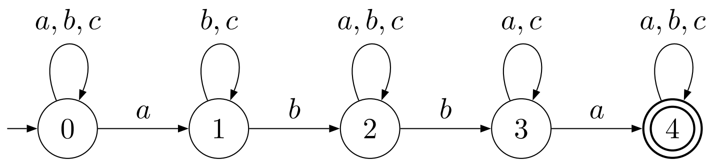
Incorrect:

i The empty language $\emptyset$; i.e., the empty set, does not contain any *words, not even the empty one.*

iii By definition, $\epsilon \in L^*$ for any *language L, including the empty language $\emptyset$.*

iv $\{\epsilon\}$ is an example of a non-empty language that does contain the empty word ϵ.

(b) Consider the following finite automaton A over $\Sigma = \{a, b, c\}$:



Which of the following statements about A are correct?

(i) The automaton A is a Deterministic Finite Automaton (DFA).

(ii) $\epsilon \in L(A)$

(iii) $aaacabca \in L(A)$

(iv) $bbaacbabcac \in L(A)$

(v) The language accepted by the automaton A is all words over Σ that contains the letters a, b, b, a in that order. (5)

Answer: *Correct: iv, v*

Incorrect:

i The automaton is an NFA since more than one transition sometimes are possible for some states and alphabet symbols.

ii $\epsilon \notin L(A)$ since no start state is accepting.

iii $aaacabca \notin L(A)$ since it is not possible to reach any accepting state on this word.

(c) Consider the following regular expression:

$$\mathbf{a}^*(\mathbf{ab})^*(\mathbf{abc})^*$$

Which of the following regular expressions denote the *same* language as the above regular expression?

(i) $(\mathbf{a} + \mathbf{ab} + \mathbf{abc})^*$

(ii) $\mathbf{a}^*(\mathbf{a} + \mathbf{b})^*(\mathbf{a} + \mathbf{b} + \mathbf{c})^*$

(iii) $\mathbf{a}^*(\epsilon + \mathbf{ab})^*(\emptyset + \mathbf{abc})^*$

(iv) $\mathbf{a}^* + (\mathbf{ab})^* + (\mathbf{abc})^*$

(v) $\mathbf{a}^*(\mathbf{a}^*\mathbf{b}^*)^*(\mathbf{a}^*\mathbf{b}^*\mathbf{c}^*)^*$ (5)

Answer: *Correct: iii*

Incorrect: i, ii, iv, v

(d) Consider the following Context-Free Grammar (CFG) G:

$$\begin{array}{rcl} S & \to & XX \mid Y \\ X & \to & aXc \mid aYc \\ Y & \to & Yb \mid \epsilon \end{array}$$

where S, X, Y are nonterminal symbols, S is the start symbol, and a, b, c are terminal symbols.

Which of the following statements about the language $L(G)$ generated by G are correct?

(i) $\epsilon \in L(G)$

(ii) $aabbbccac \in L(G)$

(iii) $aabbbccbb \in L(G)$

(iv) $L(G) = L_1L_1 \cup L_2$ where $L_1 = \{a^ib^jc^i \mid i, j \in \mathbb{N},\ i > 0\}$ and $L_2 = \{b^i \mid i \in \mathbb{N}\}$ (where $\mathbb{N} = \{0, 1, 2, \ldots\}$)

(v) The following CFG G' is equivalent to G above; i.e., $L(G') = L(G)$:

$$\begin{array}{rcl} S & \to & XX \\ X & \to & aXc \mid Y \\ Y & \to & Yb \mid \epsilon \end{array}$$

(5)

Answer: *Correct: i, ii, iv*

Incorrect:

iii *Since the word contains a's and c's, the derivation must begin $S \Rightarrow XX$. The only possibility to derive the word from XX is to split it into two parts after the last c, and derive the first part from the first X and the last part from the second X. But while aabbbcc can be derived from the first X, bb cannot be derived from the second.*

v *Not equivalent because ϵ can now be derived from X, meaning that a word like $ac \in L(G')$. However, $ac \notin L(G)$.*

(e) Which of the following statements about the Halting Problem are correct?

(i) The Halting Problem is undecidable.

(ii) The Halting Problem is semi-decidable.

(iii) The Halting Problem is the problem that Turing Machines can get stuck.

(iv) There is no Turing Machine that decides the Halting Problem.

(v) A Turing machine that never halts has got the Halting Problem.

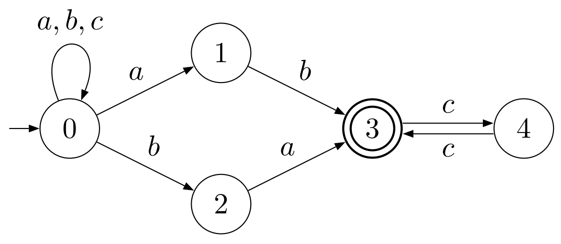
(5)

Answer: *Correct: i, ii, iv*
(The semi-decidable languages are a proper subset of the undecidable ones.)

Incorrect: iii, v

Question 2

(a) Given the following Nondeterministic Finite Automaton (NFA) N over the alphabet $\Sigma = \{a,\ b,\ c\}$, construct a Deterministic Finite Automaton (DFA) $D(N)$ that accepts the same language as N by applying the *subset construction*:

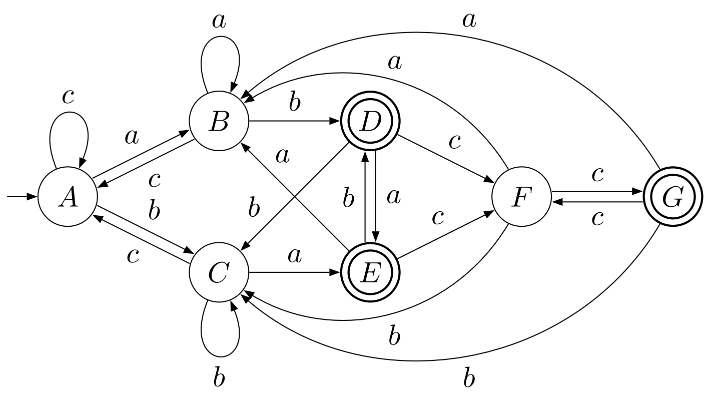


To save work, consider only the *reachable* part of $D(N)$. Clearly show your calculations in a state-transition *table*. Then draw the transition *diagram* for the resulting DFA $D(N)$. Do not forget to indicate the initial state and the final states both in the transition table and the final transition diagram. (12)

Answer:

	$\delta_{D(N)}$	a	b	c
$\rightarrow$	$\{0\} = A$	$\{0,1\} = B$	$\{0,2\} = C$	$\{0\} = A$
	$\{0,1\} = B$	$\{0,1\} = B$	$\{0,2,3\} = D$	$\{0\} = A$
	$\{0,2\} = C$	$\{0,1,3\} = E$	$\{0,2\} = C$	$\{0\} = A$
$*$	$\{0,2,3\} = D$	$\{0,1,3\} = E$	$\{0,2\} = C$	$\{0,4\} = F$
$*$	$\{0,1,3\} = E$	$\{0,1\} = B$	$\{0,2,3\} = D$	$\{0,4\} = F$
	$\{0,4\} = F$	$\{0,1\} = B$	$\{0,2\} = C$	$\{0,3\} = G$
$*$	$\{0,3\} = G$	$\{0,1\} = B$	$\{0,2\} = C$	$\{0,4\} = F$

We can now draw the transition diagram for $D(N)$:



(b) What language does the following Turing Machine (TM) M accept?

$$M = (Q,\, \Sigma,\, \Gamma,\, \delta, q_0,\, B,\, F)$$

where

$$\begin{array}{rcl@{\qquad}rcl}
Q & = & \{q_0, q_1, q_2, q_3, q_4\} & \delta(q_0, a) & = & \{(q_1, x, R)\} \\
\Sigma & = & \{a, b, c\} & \delta(q_1, a) & = & \{(q_1, x, R)\} \\
\Gamma & = & \{a, b, c, x, B\} & \delta(q_1, b) & = & \{(q_2, x, R)\} \\
F & = & \{q_4\} & \delta(q_1, c) & = & \{(q_3, x, R)\} \\
 & & & \delta(q_2, b) & = & \{(q_2, x, R)\} \\
 & & & \delta(q_2, c) & = & \{(q_3, x, R)\} \\
 & & & \delta(q_3, c) & = & \{(q_3, x, R)\} \\
 & & & \delta(q_3, B) & = & \{(q_4, B, R)\} \\
 & & & \delta(q, x) & = & \text{stop} \qquad \text{elsewhere}
\end{array}$$

Give a precise, mathematical characterisation of the accepted language along with a brief explanation of why the machine accepts this language. (8)

Answer: *$L(M) = \{a^i b^j c^k \mid i, j, k \in \mathbb{N}, i \geq 1, k \geq 1\}$. Alternatively, we can observe that the language actually is regular and given by e.g. the regular expression* $\mathbf{aa}^*\mathbf{b}^*\mathbf{cc}^*$.

The machine only ever move right and either stays in a state or moves to a higher numbered state. The machine starts in state q_0, where only 'a' is accepted, causing the machine to move to state q_1. In q_1 the machine can loop on 'a', accepting a further arbitrary number of 'a's, or it can accept a 'b' or a 'c' moving to q_2 or q_3, respectively. Both q_2 and q_3 are looping states, allowing an arbitrary number of 'b's and 'c's respectively to be accepted. A single 'c' is required to move form q_2 to q_3, but note that q_3 also can be reached directly from q_1 on a single 'c', meaning that the number of 'b's in an accepted word can be 0. The words accepted are thus those consisting of 1 or more 'a's, followed by 0 or more 'b's, followed by 1 or more 'c's.

(c) In the context of Turing Machines, explain what it means for a language to be: *recursive*, *recursively-enumerable*, *decidable*, *undecidable*. (5)

Answer: *A recursive language is a language that is accepted by a Turing Machine that always halts. This is the same as saying that the language is decidable (or that the problem represented by the language is decidable).*

A recursively enumerable language is a language that is accepted by a Turing Machine (but not by any Turing machine that necessarily halts).

A language that is not recursive is undecidable. The undecidable languages thus includes the recursively enumerable (or semi-decidable) languages and those languages which are not even recursively-enumerable.

Question 3

(a) The following is a context-free grammar (CFG) for simple arithmetic expressions. For simplicity, we only consider the numbers 0, 1, and 2:

$$\begin{aligned} E &\rightarrow E + E \mid E * E \mid E \uparrow E \mid (E) \mid N \\ N &\rightarrow 0 \mid 1 \mid 2 \end{aligned}$$

E and N are nonterminals, E is the start symbol, $+$, $*$, $\uparrow$, (,), 0, 1, 2 are terminals.

Show that this grammar is ambiguous. (5)

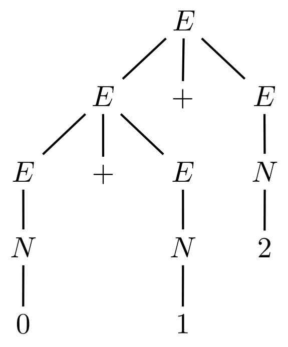
Answer: *To demonstrate ambiguity, demonstrate that there is at least one derivable word for which there are two different leftmost or rightmost derivations, or two different parse tress. For example, here are two different leftmost derivations of the word* $0 + 1 + 2$*:*

$$\begin{aligned} E &\underset{\text{lm}}{\Rightarrow} E + E \\ &\underset{\text{lm}}{\Rightarrow} E + E + E \\ &\underset{\text{lm}}{\Rightarrow} N + E + E \\ &\underset{\text{lm}}{\Rightarrow} 0 + E + E \\ &\underset{\text{lm}}{\Rightarrow} 0 + N + E \\ &\underset{\text{lm}}{\Rightarrow} 0 + 1 + E \\ &\underset{\text{lm}}{\Rightarrow} 0 + 1 + N \\ &\underset{\text{lm}}{\Rightarrow} 0 + 1 + 2 \end{aligned}$$

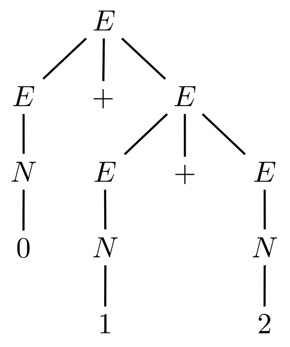
and

$$\begin{aligned} E &\underset{\text{lm}}{\Rightarrow} E + E \\ &\underset{\text{lm}}{\Rightarrow} N + E \\ &\underset{\text{lm}}{\Rightarrow} 0 + E \\ &\underset{\text{lm}}{\Rightarrow} 0 + E + E \\ &\underset{\text{lm}}{\Rightarrow} 0 + N + E \\ &\underset{\text{lm}}{\Rightarrow} 0 + 1 + E \\ &\underset{\text{lm}}{\Rightarrow} 0 + 1 + N \\ &\underset{\text{lm}}{\Rightarrow} 0 + 1 + 2 \end{aligned}$$

Alternatively, here are the two parse trees corresponding to the derivations above:



and



(b) Construct an equivalent but *unambiguous* version of the context-free grammar for arithmetic expressions given in (a) above by making it reflect the following conventions regarding *operator precedence* and *associativity*:

Operators	Precedence	Associativity
$\uparrow$	highest	right
$*$	medium	left
$+$	lowest	left

It should further be possible to use parentheses for grouping in the usual way. (8)

Answer: *Stratify the grammar so that productions for operators refer to either productions at the same level or productions at the next level up, and use left-recursive productions for left-associative operators and right-recursive productions for right-associative operators.*

$$\begin{array}{rcl} E & \to & E + E_1 \mid E_1 \\ E_1 & \to & E_1 * E_2 \mid E_2 \\ E_2 & \to & E_3 \uparrow E_2 \mid E_3 \\ E_3 & \to & (E) \mid N \\ N & \to & 0 \mid 1 \mid 2 \end{array}$$

(c) The following Context-free grammar (CFG) is immediately left-recursive:

$$\begin{array}{lcl} S & \to & aS \mid bX \\ X & \to & XXc \mid YXd \mid Y \\ Y & \to & Ye \mid Yf \mid g \end{array}$$

S, X, and Y are nonterminals, a, b, c, d, e, f, and g are terminals, and S is the start symbol.

Transform this grammar into an equivalent right-recursive CFG. State the general transformation rule you are using and show the main transformation steps. (12)

Answer: *First identify the immediately left-recursive non-terminals. Then group the productions for each such non-terminal into two groups: one where each RHS starts with the non-terminal in question, and one where they don't:*

$$\begin{array}{lcl} A & \to & A\alpha_1 \mid \ldots \mid A\alpha_m \\ A & \to & \beta_1 \mid \ldots \mid \beta_n \end{array}$$

Then replace those productions with new productions for A and productions for A', where A' is a new name, as follows:

$$\begin{array}{lcl} A & \to & \beta_1 A' \mid \ldots \mid \beta_n A' \\ A' & \to & \alpha_1 A' \mid \ldots \mid \alpha_m A' \mid \epsilon \end{array}$$

There are two immediately left-recursive non-terminals in the given grammar: X and Y. The grammar is essentially already grouped as required. Applying the above transformation rule to both the X and Y productions yields:

$$\begin{array}{lcl} S & \to & aS \mid bX \\ X & \to & YXdX' \mid YX' \\ X' & \to & XcX' \mid \epsilon \\ Y & \to & gY' \\ Y' & \to & eY' \mid fY' \mid \epsilon \end{array}$$