The University of Nottingham

SCHOOL OF COMPUTER SCIENCE

A LEVEL 2 MODULE, AUTUMN SEMESTER 2008–2009

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 QUESTION ONE and any TWO other 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 (Compulsory)
The following questions are multiple choice. There is at least one correct choice, but there may be several. To get all the marks you have to list all the correct answers and none of the wrong ones.

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) An alphabet is a finite sequence of distinct symbols.
(ii) A word is a finite sequence of symbols over a given alphabet.
(iii) A language is a possibly infinite set of words over a given alphabet.
(iv) A regular language is always infinite.
(v) An infinite language can be regular.

(5)

Answer: *Correct: ii, iii, v*

Incorrect:

i An alphabet is a set *of symbols, not a sequence.*
iv All finite languages are regular.

(b) 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 belongs to all non-empty languages.
(v) $\epsilon \in \{a^i b^j \mid i, j \in \mathbb{N}, i + j \leq 42\}$

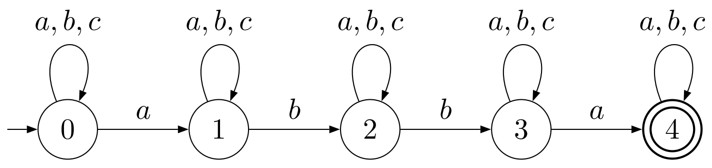
(5)

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 $\{a\}$ is an example of a non-empty language that does not contain the empty word ϵ.

(c) 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) $bacabca \in L(A)$

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

(v) The language accepted by the automaton A is all words over Σ that contains each of the letters a, b, b, a at least once 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 $bacabca \notin L(A)$ since it is not possible to reach any accepting state on this word.

(d) 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

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

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

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 $L_2 = \{b^i \mid i \in \mathbb{N}\}$

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

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

(5)

Answer: *Correct: i, ii, iv*

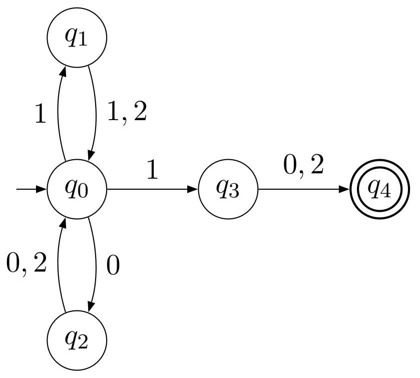
Incorrect:

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, $ac \notin L(G)$.

Question 2

(a) Given the following NFA N over the alphabet $\Sigma = \{0,\ 1,\ 2\}$, construct a DFA $D(N)$ that accepts the same language as N by applying the *subset construction*:

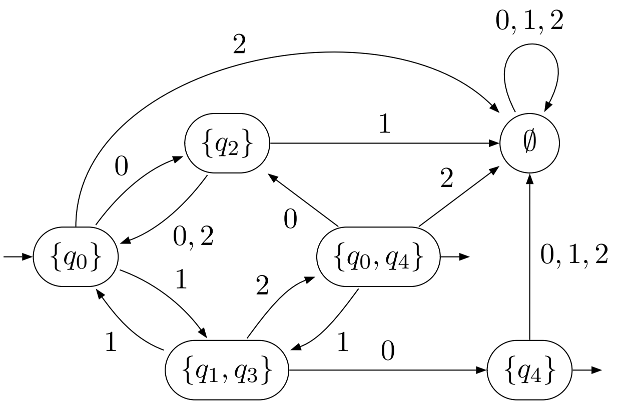


To save work, consider only the *reachable* part of $D(N)$. Clearly show your calculations in a state-transition table. Then draw the transition diagram for the resulting DFA $D(N)$. Do not forget to indicate the initial state and the final states both in the transition table and the final transition diagram. (15)

Answer: *Starting from $S = \{q_0\}$, the set of start states of N and thus the start state of $D(N)$, we compute $\hat{\delta}_N(S, x)$ for each $x \in \Sigma$. Whenever we encounter a state $P \subseteq Q$ of $D(N)$ that has not been considered before, we add P to the table and proceed to tabulate $\hat{\delta}_N(P, x)$ for each $x \in \Sigma$. We repeat the process until no new states are encountered. Finally, we identify the initial state ($\rightarrow$ to the left of the state) and all accepting states ($*$ to the left of the state). Note that a DFA state is accepting if it contains at least one accepting NFA state (as this means it is* possible *to reach at least one accepting state on a given word, which means that word is considered to be in the language of the NFA).*

	$\delta_{D(N)}$	0	1	2
$\rightarrow$	$\{q_0\}$	$\{q_2\}$	$\{q_1, q_3\}$	$\emptyset$
	$\{q_2\}$	$\{q_0\}$	$\emptyset$	$\{q_0\}$
	$\{q_1, q_3\}$	$\emptyset \cup \{q_4\} = \{q_4\}$	$\{q_0\} \cup \emptyset = \{q_0\}$	$\{q_0\} \cup \{q_4\} = \{q_0, q_4\}$
	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$*$	$\{q_4\}$	$\emptyset$	$\emptyset$	$\emptyset$
$*$	$\{q_0, q_4\}$	$\{q_2\} \cup \emptyset = \{q_2\}$	$\{q_1, q_3\} \cup \emptyset = \{q_1, q_3\}$	$\emptyset \cup \emptyset = \emptyset$

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



Accepting states have been marked by outgoing arrows in this case. That is an alternative to the double circle.

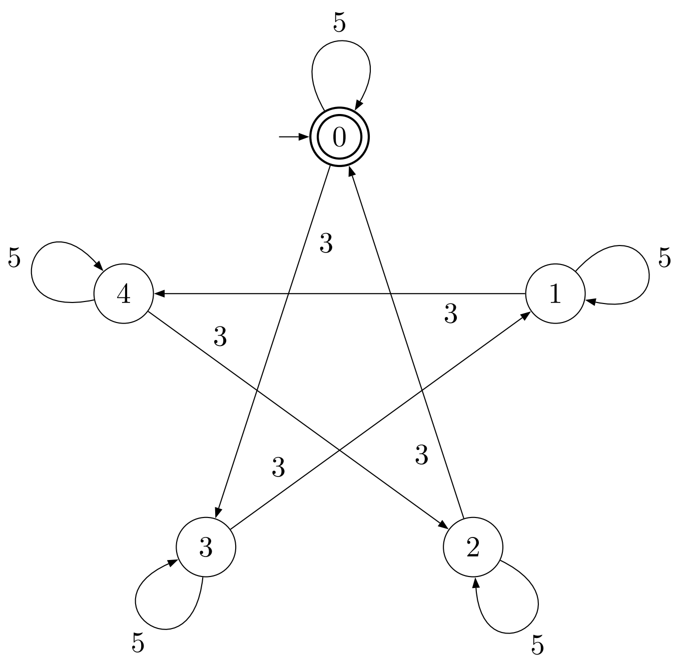
(b) Consider the language L over the alphabet $\Sigma = \{3, 5\}$ of all words for which the arithmetic sum of the constituent symbols is divisible by 5. For example, $\epsilon \in L$ (there are no symbols in the empty string, the sum is thus 0 which is divisible by 5), $555 \in L$ $(5 + 5 + 5 = 15$ which is divisible by 5), and $335333 \in L$ $(3 + 3 + 5 + 3 + 3 + 3 = 20$ which is divisible by 5), but $333 \notin L$ $(3 + 3 + 3 = 9$ which is not divisible by 5 (the reminder is 4)).

Is L a regular language or not? If it is, construct a DFA A such that $L(A) = L$. Your answer should consist of the transition diagram for A, with initial and final state(s) clearly identified, along with a *brief* justification (in plain English) that makes the idea behind the construction clear and thus explains why the given automaton accepts the language in question.

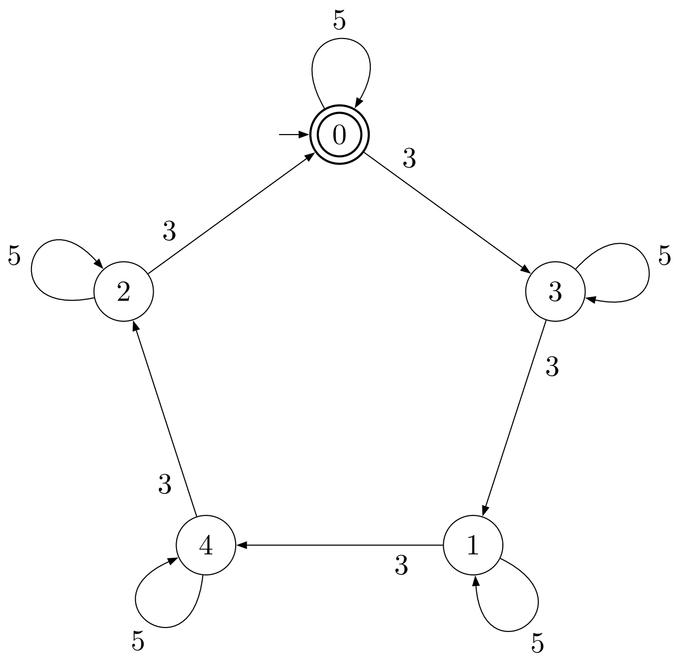
If L is not a regular language, prove this by using the pumping lemma for regular languages. (10)

Answer: *The language L is regular. It is just a variation of the type of language exemplified by "all strings with an odd number of a particular symbol", and can thus be recognised by a variation of the DFAs for recognising that type of language.*

We need one state for each possible reminder when dividing by 5, i.e. 5 states. Let us label each state with the reminder in question. State 0 is thus both the initial and the only final state. We then just note that if the reminder when dividing the sum n of the symbols seen so far by 5 is r, and the next symbol is i, then the reminder of $n + i$ divided by 5 is just the reminder of $r + i$ divided by 5.



Or, if you prefer, can be drawn like this:



Question 3

(a) Give regular expressions defining the following languages over the alphabet $\Sigma = \{a, b, c\}$:

(i) All words that contain an a followed by a b (possibly with other symbols in between).

(ii) All words that contain at least one a and at most one b.

You only need to provide the regular expressions as an answer, but they should not be unnecessarily complicated. (5)

Answer: *Note: these are not the only possibilities, nor necessarily the "simplest" in any formal sense. But they are all fairly simple, and your answers should not be much more complicated.*

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

An alternative. (It is essential that the first parentheses is a choice between **a**, **b**, *and* **c**, *as opposed to just* **b** *and* **c**. *Why?)*

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

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

Another alternative. (It is essential that the first parentheses is a choice between **a** *and* **c**, *as opposed to starting with just an iteration of* **c**. *Why?)*

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

And a slight variation:

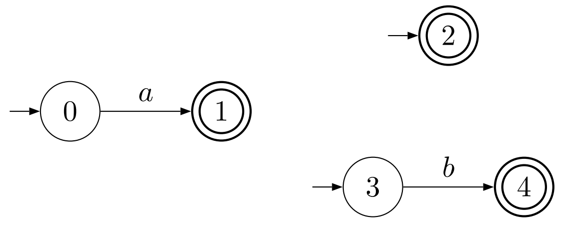
$(\mathbf{a}+\mathbf{c})^*(\mathbf{a}+\mathbf{a}\mathbf{c}^*\mathbf{b}+\mathbf{b}\mathbf{c}^*\mathbf{a})(\mathbf{a}+\mathbf{c})^*$

(b) Systematically construct an NFA for the regular expression

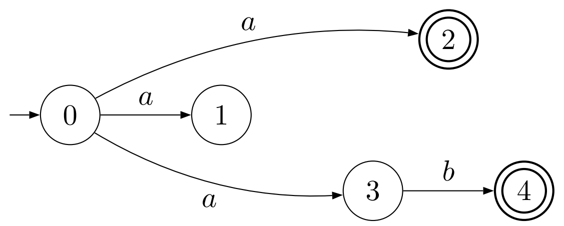
$$(\mathbf{a}(\epsilon+\mathbf{b}))^*(\mathbf{c}+\mathbf{d}+\emptyset)$$

by following the graphical construction from the lecture notes. Make sure it is clear how you undertake the construction by showing the major steps. Eliminate "dead ends" (states from which no final state can be reached) when they appear. The states in the final NFA should be named, but as long as it is clear what you are doing, you can leave the states of intermediate NFAs unnamed. (10)

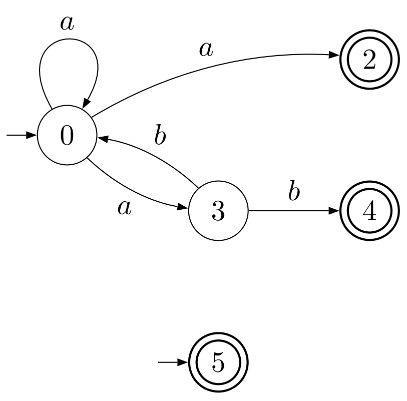
Answer: *First construct an NFA A for the subexpression* $(\mathbf{a}(\epsilon+\mathbf{b}))^*$ *according to the lecture notes. (I have named the states according to how they will be named in the final NFA to make it easier to follow the derivation. It is OK to leave states unnamed to the end. Also, it is not necessary to show all intermediate stages of the construction: here for clarity only.) NFA for* **a** *(to the left) and for* $\epsilon+\mathbf{b}$ *(to the right):*



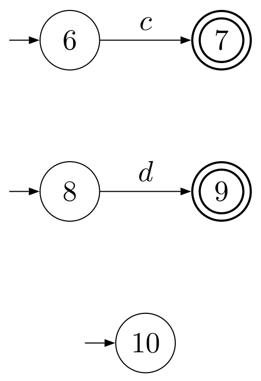
Join the above two NFAs to obtain an NFA for $\mathbf{a}(\epsilon + \mathbf{b})$*, keeping in mind that the left DFA does not accept* ϵ*:*



It is now clear state 1 is a dead end, so it can be removed prior to carrying out the construction for the Kleene star (not forgetting one extra state for accepting ϵ*) leaving us with the following NFA A for* $(\mathbf{a}(\epsilon + \mathbf{b}))^*$*:*

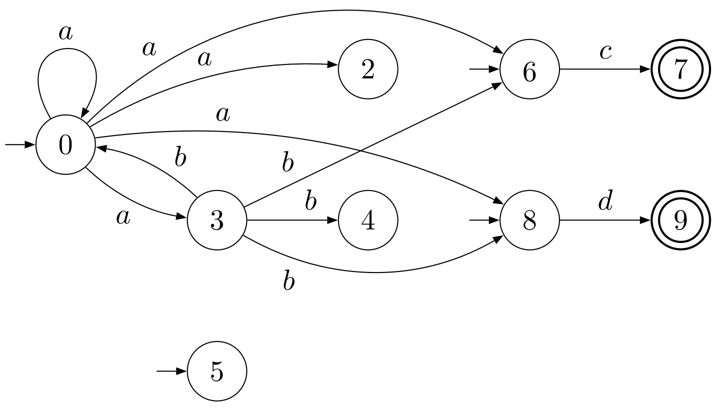


The NFA B for $(\mathbf{c} + \mathbf{d} + \emptyset)$ *is simply:*

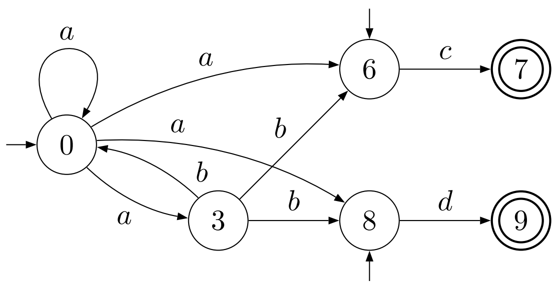


It's clear that state 10 will become a dead end, so it can be removed.

Now, joining A with B (less state 10), while keeping in mind that A can *accept ϵ which means states 6 and 8 will remain initial states, gives:*



It's now clear that states 2, 4, and 5 are all dead ends, so we can simplify and obtain the final NFA for $(\mathbf{a}(\epsilon + \mathbf{b}))^*(\mathbf{c} + \mathbf{d} + \emptyset)$*:*



Note that there are three *initial states: 0, 6, and 8.*

(c) Construct an *unambiguous* Context-Free Grammar (CFG) for regular expressions over the alphabet $\Sigma = \{a, b, c\}$ (with the syntax defined in the lecture notes). To ensure your grammar is unambiguous, it should reflect the precedence and associativity for the regular expression constructs as specified by the following table:

Operators	Precedence	Associativity
$*$	highest	n/a
concatenation	medium	left
$+$	lowest	left

For example,

$$(\mathbf{a}(\epsilon + \mathbf{b}))^* + \emptyset$$

is a valid regular expression, whereas both

$$(\mathbf{a}$$

(because the parentheses are not balanced) and

$$\mathbf{a}+$$

(because + is a binary operator) are not. (10)

Answer: *The following is one possible grammar. It has been stratified to capture the desired precedence levels, and left recursion is used to impart left associativity on the relevant constructs according to the specification:*

$$\begin{array}{rcl} E & \rightarrow & E + E_1 \mid E_1 \\ E_1 & \rightarrow & E_1 E_2 \mid E_2 \\ E_2 & \rightarrow & E_2 * \mid E_3 \\ E_3 & \rightarrow & (E) \mid E_P \\ E_P & \rightarrow & \mathbf{a} \mid \mathbf{b} \mid \mathbf{c} \mid \epsilon \mid \emptyset \end{array}$$

Here, E, E_1, E_2, and E_P are nonterminals with E being the start symbol, and $+$, $*$, (,), **a**, **b**, **c**, *ϵ, $\emptyset$ are terminals. (Note in particular that $E_P \rightarrow \epsilon$ is* not *an epsilon production in this case!)*

Question 4

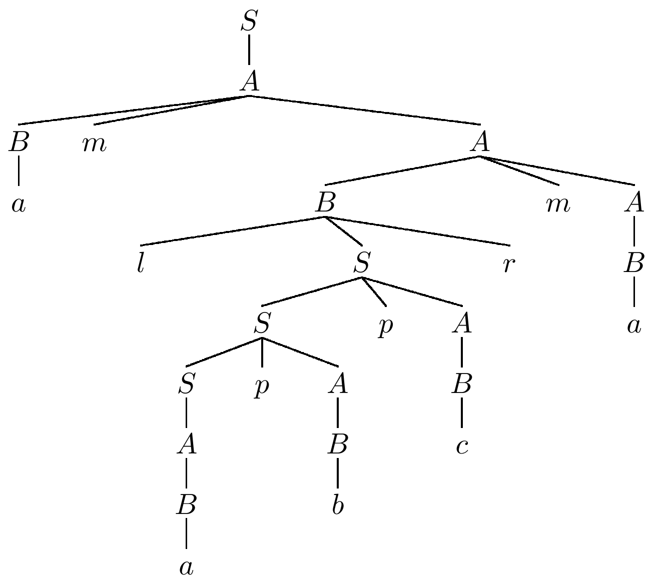
(a) Consider the following Context-Free Grammar (CFG):

$$\begin{array}{lcl} S & \rightarrow & SpA \mid A \\ A & \rightarrow & BmA \mid B \\ B & \rightarrow & a \mid b \mid c \mid lSr \end{array}$$

S, A, and B are nonterminals, a, b, c, l, m, p, and r are terminals, and S is the start symbol.

Draw the derivation tree according to this grammar for the word *amlapbpcrma*. (5)

Answer: *Derivation tree for amlapbpcrma:*



(b) Explain what it means for a Context-Free Grammar (CFG) to be *ambiguous*. (5)

Answer: *A context-free grammar is ambiguous if there exists at least one word in the language generated by the grammar for which which there is more than one derivation tree, or, equivalently, for which there is more than one leftmost or more than one rightmost derivation.*

(c) Is the following CFG ambiguous? If yes, show this. If no, explain why.

$$\begin{array}{rcl} A & \rightarrow & AaA \mid AbA \mid B \\ B & \rightarrow & c \end{array}$$

A and B are nonterminals, A is the start symbol, a, b, and c are terminals. (5)

Answer: *Yes, the grammar is ambiguous. Two different leftmost derivations for the word cacac:*

$$\begin{array}{rcl} A & \underset{\text{lm}}{\Rightarrow} & AaA \\ & \underset{\text{lm}}{\Rightarrow} & BaA \\ & \underset{\text{lm}}{\Rightarrow} & caA \\ & \underset{\text{lm}}{\Rightarrow} & caAaA \\ & \underset{\text{lm}}{\Rightarrow} & caBaA \\ & \underset{\text{lm}}{\Rightarrow} & cacaA \\ & \underset{\text{lm}}{\Rightarrow} & cacaB \\ & \underset{\text{lm}}{\Rightarrow} & cacac \end{array}$$

and

$$\begin{array}{rcl} A & \underset{\text{lm}}{\Rightarrow} & AaA \\ & \underset{\text{lm}}{\Rightarrow} & AaAaA \\ & \underset{\text{lm}}{\Rightarrow} & BaAaA \\ & \underset{\text{lm}}{\Rightarrow} & caAaA \\ & \underset{\text{lm}}{\Rightarrow} & caBaA \\ & \underset{\text{lm}}{\Rightarrow} & cacaA \\ & \underset{\text{lm}}{\Rightarrow} & cacaB \\ & \underset{\text{lm}}{\Rightarrow} & cacac \end{array}$$

(d) The following Context-Free Grammar (CFG) is immediately *left-recursive*:

$$\begin{aligned} S &\rightarrow aS \mid bX \\ X &\rightarrow XXc \mid Xd \mid Y \\ Y &\rightarrow Ye \mid f \mid g \end{aligned}$$

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. (10)

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:*

$$\begin{aligned} A &\rightarrow A\alpha_1 \mid \ldots \mid A\alpha_m \\ A &\rightarrow \beta_1 \mid \ldots \mid \beta_n \end{aligned}$$

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

$$\begin{aligned} A &\rightarrow \beta_1 A' \mid \ldots \mid \beta_n A' \\ A' &\rightarrow \alpha_1 A' \mid \ldots \mid \alpha_m A' \mid \epsilon \end{aligned}$$

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:

$$\begin{aligned} S &\rightarrow aS \mid bX \\ X &\rightarrow YX' \\ X' &\rightarrow XcX' \mid dX' \mid \epsilon \\ Y &\rightarrow fY' \mid gY' \\ Y' &\rightarrow eY' \mid \epsilon \end{aligned}$$

Question 5
Consider the following Pushdown Automaton (PDA) P:

$$P = (Q = \{q_0, q_1, q_2\},\ \Sigma = \{a, b, c\},\ \Gamma = \{a, \#\},\ \delta,\ q_0,\ Z_0 = \#,\ F = \{q_2\})$$

where the transition function δ is given by

$$\begin{array}{rcll}
\delta(q_0, a, \#) & = & \{(q_0, a\#)\} & \\
\delta(q_0, c, \#) & = & \{(q_0, \#)\} & \\
\delta(q_0, a, a) & = & \{(q_0, aa)\} & \\
\delta(q_0, b, a) & = & \{(q_1, \epsilon)\} & \\
\delta(q_0, c, a) & = & \{(q_0, a)\} & \\
\delta(q_1, c, \#) & = & \{(q_1, \#)\} & \\
\delta(q_1, b, a) & = & \{(q_1, \epsilon)\} & \\
\delta(q_1, c, a) & = & \{(q_1, a)\} & \\
\delta(q_1, \epsilon, \#) & = & \{(q_2, \#)\} & \\
\delta(q, w, z) & = & \emptyset & \text{everywhere else}
\end{array}$$

Acceptance is by *final state.*

(a) Which of the following words are accepted by the PDA P?

(i) *acabbc*
(ii) *abcabc*

For those words that are accepted, provide a sequence of Instantaneous Descriptions (IDs) leading to an accepting configuration as evidence. For those words that are not accepted, explain why there is no sequence of IDs leading to an accepting configuration. (10)

Answer:

(i) The word acabbc is accepted. ID sequence:

$$\begin{array}{rcl}
(q_0, acabbc, \#) & \vdash & (q_0, cabbc, a\#) \\
& \vdash & (q_0, abbc, a\#) \\
& \vdash & (q_0, bbc, aa\#) \\
& \vdash & (q_1, bc, a\#) \\
& \vdash & (q_1, c, \#) \\
& \vdash & (q_1, \epsilon, \#) \\
& \vdash & (q_2, \epsilon, \#)
\end{array}$$

Accepting configuration since q_2 is an accepting state and since all input has been read.
[Marking: 5 points]

(ii) The string abcabc is not accepted. For the first two moves, there is no choice:

$$\begin{array}{rcl}(q_0, abcabc, \#) & \vdash & (q_0, bcabc, a\#) \\ & \vdash & (q_1, cabc, \#)\end{array}$$

Here there is a choice: consume c and stay in q_1 or an epsilon move to q_2. But the latter cannot lead to acceptance at this point since not all input has been read.

$$\vdash \quad (q_1, abc, \#)$$

Now the only possibility is an epsilon move to q_2:

$$\vdash \quad (q_2, abc, \#)$$

This is not an accepting configuration as not all input has been read.

[Marking: 5 points]

(b) Consider the following Context-Free Grammar (CFG):

$$\begin{array}{rcl} S & \rightarrow & ABC \mid BC \\ A & \rightarrow & aA \mid a \\ B & \rightarrow & b \mid \epsilon \\ C & \rightarrow & c \mid d \mid \epsilon \end{array}$$

S, A, B, and C are nonterminals, a, b, c, and d are terminals, and S is the start symbol.

(i) What is the set N_ϵ of *nullable* nonterminals? Provide a brief justification. (2)

Answer: *$N_\epsilon = \{S, B, C\}$. B is nullable because $B \rightarrow \epsilon$ is a production. C is nullable because $C \rightarrow \epsilon$ is a production. S is nullable because $S \rightarrow BC$ is a production and both B and C are nullable. A is not nullable since the RHS of all productions for A start with a terminal (a).*

(ii) Systematically compute the *first sets* for all nonterminals, i.e. first(S), first(A), first(B), and first(C), by setting up and solving the equations according to the definitions of first sets for nonterminals and strings of grammar symbols. Show your calculations. (4)

Answer:

$$\begin{array}{rcl} \text{first}(A) & = & \text{first}(aA) \cup \text{first}(a) \\ & = & \{a\} \cup \{a\} \\ & = & \{a\} \end{array}$$

$$\begin{array}{rcl} \text{first}(B) & = & \text{first}(b) \cup \text{first}(\epsilon) \\ & = & \{b\} \cup \emptyset \\ & = & \{b\} \end{array}$$

$$\begin{array}{rcl} \text{first}(C) & = & \text{first}(c) \cup \text{first}(d) \cup \text{first}(\epsilon) \\ & = & \{c\} \cup \{d\} \cup \emptyset \\ & = & \{c, d\} \end{array}$$

$$\begin{array}{rcl} \text{first}(S) & = & \text{first}(ABC) \cup \text{first}(BC) \\ & = & (\text{first}(A) \cup \emptyset) \cup (\text{first}(B) \cup \text{first}(C) \cup \emptyset) \\ & = & \{a\} \cup \{b\} \cup \{c, d\} \\ & = & \{a, b, c, d\} \end{array}$$

(iii) Set up the subset constraint system that defines the *follow sets* for all nonterminals, i.e. follow(S), follow(A), follow(B), and follow(C). Simplify where possible using the law

$$A \subseteq C \ \land \ B \subseteq C \quad \Longleftrightarrow \quad A \cup B \subseteq C$$

and the fact that constraints like $A \subseteq A$ are trivially satisfied and can be omitted. (7)

Answer: *Note: detailed account below for clarity. It is sufficient to just state the constraints according to the definitions and then simplify.*

Constraints for follow(S)*:*

$$\{\$\} \subseteq \text{follow}(S)$$

Constraints for follow(A) *from the productions where A occurs in the RHS, i.e.*

$$\begin{aligned} S &\rightarrow ABC \\ A &\rightarrow aA \end{aligned}$$

(note: nullable(BC) *and* nullable(ϵ)*):*

$$\begin{aligned} \text{first}(BC) &\subseteq \text{follow}(A) \\ \text{follow}(S) &\subseteq \text{follow}(A) \\ \text{first}(\epsilon) &\subseteq \text{follow}(A) \\ \text{follow}(A) &\subseteq \text{follow}(A) \end{aligned}$$

Constraints for follow(B) *from the productions where B occurs in the RHS, i.e.*

$$\begin{aligned} S &\rightarrow ABC \\ S &\rightarrow BC \end{aligned}$$

(note: nullable(C)*):*

$$\begin{aligned} \text{first}(C) &\subseteq \text{follow}(B) \\ \text{follow}(S) &\subseteq \text{follow}(B) \\ \text{first}(C) &\subseteq \text{follow}(B) \\ \text{follow}(S) &\subseteq \text{follow}(B) \end{aligned}$$

Constraints for follow(C) *from the productions where C occurs in the RHS, i.e.*

$$\begin{aligned} S &\rightarrow ABC \\ S &\rightarrow BC \end{aligned}$$

(note: nullable(ϵ)*):*

$$\begin{aligned} \text{first}(\epsilon) &\subseteq \text{follow}(C) \\ \text{follow}(S) &\subseteq \text{follow}(C) \end{aligned}$$

Using

$$\begin{aligned} \text{first}(\epsilon) &= \emptyset \\ \text{first}(C) &= \{c, d\} \\ \text{first}(BC) &= \text{first}(B) \cup \text{first}(C) \cup \emptyset \\ &= \{b\} \cup \{c, d\} = \{b, c, d\} \end{aligned}$$

and eliminating trivial constraints yields:

$$\begin{aligned} \{\$\} &\subseteq \text{follow}(S) \\ \\ \{b, c, d\} &\subseteq \text{follow}(A) \\ \text{follow}(S) &\subseteq \text{follow}(A) \\ \emptyset &\subseteq \text{follow}(A) \\ \\ \{c, d\} &\subseteq \text{follow}(B) \\ \text{follow}(S) &\subseteq \text{follow}(B) \\ \\ \emptyset &\subseteq \text{follow}(C) \\ \text{follow}(S) &\subseteq \text{follow}(C) \end{aligned}$$

This is equivalent to

$$\begin{aligned} \{\$\} &\subseteq \text{follow}(S) \\ \{b, c, d\} \cup \text{follow}(S) \cup \emptyset &\subseteq \text{follow}(A) \\ \{c, d\} \cup \text{follow}(S) &\subseteq \text{follow}(B) \\ \emptyset \cup \text{follow}(S) &\subseteq \text{follow}(C) \end{aligned}$$

which can be further simplified to the final constraints:

$$\begin{aligned} \{\$\} &\subseteq \text{follow}(S) \\ \{b, c, d\} \cup \text{follow}(S) &\subseteq \text{follow}(A) \\ \{c, d\} \cup \text{follow}(S) &\subseteq \text{follow}(B) \\ \emptyset \cup \text{follow}(S) &\subseteq \text{follow}(C) \end{aligned}$$

(iv) Solve the subset constraint system for the follow sets from the previous question by finding the smallest sets satisfying the constraints. (2)

Answer: *The smallest set satisfying the constraint for* follow(S) *is obviously just* $\{\$\}$. *Substituting this into the remaining constraints makes the smallest sets satisfying those obvious too. Thus:*

$$\begin{aligned}
\text{follow}(S) &= \{\$\} \\
\text{follow}(A) &= \{b, c, d\} \cup \{\$\} = \{b, c, d, \$\} \\
\text{follow}(B) &= \{c, d\} \cup \{\$\} = \{c, d, \$\} \\
\text{follow}(C) &= \{\$\}
\end{aligned}$$