The University of Nottingham

SCHOOL OF COMPUTER SCIENCE

A LEVEL 2 MODULE, SPRING SEMESTER 2016–2017

LANGUAGES AND COMPUTATION

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.

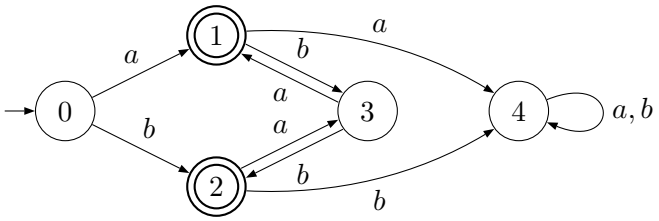
No electronic devices capable of storing and retrieving text, including electronic dictionaries, may be used.

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). Any further explanations below are just for clarification.*

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



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) $baaba \in L(A)$

(iv) All words accepted by A contain one more a than b or one more b than a.

(v) The automaton A accepts all words over Σ that contain one more a than b or one more b than a.

(5)

Answer: *Correct: i, iii, iv*

Incorrect:

ii The initial state is not accepting.

v E.g. $aab \notin L(A)$.

(b) Consider the following set W of words:

$$W = \{\epsilon,\ ab,\ cab,\ abab\}$$

Which of the following regular expressions denote a language that contains *all* words in W? (But not necessarily *only* the words in W: the language denoted by the regular expression is allowed to contain *more* words.)

(i) $(\epsilon + \mathbf{ab} + \mathbf{c})(\epsilon + \mathbf{ab})$

(ii) $(\epsilon + \mathbf{ab} + \mathbf{c})(\emptyset + \mathbf{ab})$

(iii) $(\epsilon + \mathbf{ab} + \mathbf{c})^*$

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

(v) $(\mathbf{ab})^* + \mathbf{c}^*$

(5)

Answer: *Correct: i, iii, iv*

Incorrect:

ii $\epsilon \notin L(\emptyset + \mathbf{ab})$ *and therefore* $\epsilon \notin L((\epsilon + \mathbf{ab} + \mathbf{c})(\emptyset + \mathbf{ab}))$.

v $cab \notin L((\mathbf{ab})^*)$ *and* $cab \notin L(\mathbf{c}^*)$*, and therefore* $cab \notin L((\mathbf{ab})^* + \mathbf{c}^*)$.

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

$$\begin{array}{lcl} S & \rightarrow & XY \\ X & \rightarrow & aXb \mid \epsilon \\ Y & \rightarrow & bYc \mid \epsilon \end{array}$$

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) $abbc \in L(G)$

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

(iv) $aaabbbbbbccc \in L(G)$

(v) $L(G) = \{a^n b^{2n} c^n \mid n \in \mathbb{N}\}$ (where $\mathbb{N} = \{0, 1, 2, \ldots\}$)

(5)

Answer: *Correct: i, ii, iv;*
Incorrect: iii, v ($\{a^n b^{2n} c^n \mid n \in \mathbb{N}\} \subset L(G)$)

(d) Which of the following properties of a problem P imply that P is undecidable?

(i) P is reducible to the Halting Problem.

(ii) The Halting Problem is reducible to P.

(iii) P is recursively enumerable but not recursive.

(iv) The complement of P is recursively enumerable.

(v) There is no Turing Machine that solves P.

(5)

Answer: *Correct: ii, iii, v*

Incorrect: i, iv

When a problem Q is reducible to another problem P, this means that any instance of Q can be transformed into an instance of P with the same solution. If we had an algorithm to solve P, we could then also solve Q. Therefore, if we already know that P is undecidable, this reduction shows that there cannot be an algorithm to solve P. We can conclude that P is also undecidable. This is why (iii) is correct. However if we know that P is undecidable, the fact that instances of Q are reducible to instances of P is not informative: there may still be an algorithm to solve Q, independently of the fact that we don't have one for P. That's why (ii) does not imply that P is undecidable.

Answer (v) is incorrect because it merely say that there is an algorithm that enumerates the instances of P which have a negative solution. This still leaves open the possibility that there may be another algorithm that enumerates the instances of P with a positive solution. In that case, by combining the two algorithms, we may be able to decide P.

(e) Which of the following statements about the λ-calculus are true?

(i) Every λ-term has a normal form.

(ii) Every λ-term has a fixed point.

(iii) Every computable function can be represented by a λ-term.

(iv) The normalization property of λ-terms is a decidable problem.

(v) The set of functions representable by λ-terms is the same as those computable by Turing Machines.

(5)

Answer: *Correct: ii,iii,v*

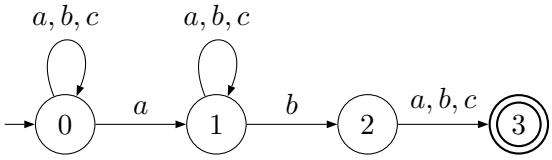
Incorrect: i,iv

Answer (i) is incorrect because there are λ-terms whose reduction sequences go on forever (independently of what reduction strategy we use). One example is $(\lambda x.x\,x)\,(\lambda x.x\,x)$.

Answer (iv) is incorrect because normalization of λ-terms is equivalent to the Halting Problem (it is in fact the formulation of the Halting Problem in the λ-calculous). Therefore it is undecidable.

Question 2

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



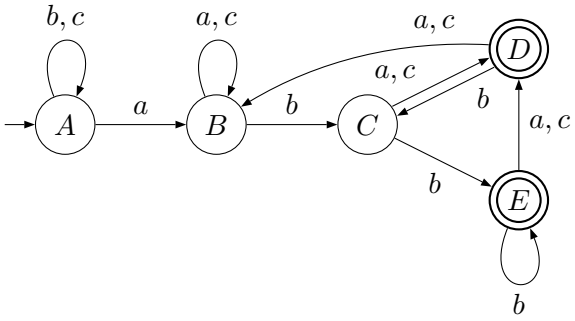
Show your calculations in a state-transition *table*. Consider only the *reachable* part of $D(N)$. Then draw the transition *diagram* for the resulting DFA $D(N)$. Indicate the initial state and the final states both in the transition table and the final transition diagram. (12)

Answer:

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

The states have been named (A, B, ..., E) to facilitate drawing the transition diagram.

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



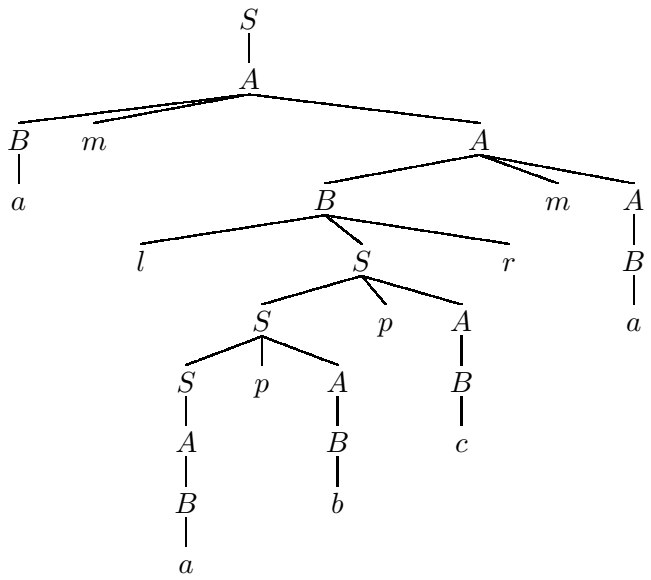
(b) Consider the following Context-Free Grammar (CFG):

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

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:*



(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. (8)

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:*

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

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

Question 3

(a) Write the λ-terms that represent the following values:

- The Boolean values true and false;
- The *exclusive or* function xor, such that

$$\mathsf{xor}\,\mathsf{true}\,\mathsf{true} \rightsquigarrow^* \mathsf{false}, \qquad \mathsf{xor}\,\mathsf{false}\,\mathsf{true} \rightsquigarrow^* \mathsf{true},$$
$$\mathsf{xor}\,\mathsf{true}\,\mathsf{false} \rightsquigarrow^* \mathsf{true}, \qquad \mathsf{xor}\,\mathsf{false}\,\mathsf{false} \rightsquigarrow^* \mathsf{false};$$

- For every couple of terms a and b, a term $\langle a, b\rangle$ representing the pair, such that

$$\langle a, b\rangle\,\mathsf{true} \rightsquigarrow^* a, \qquad \langle a, b\rangle\,\mathsf{false} \rightsquigarrow^* b;$$

- The Church Numeral $\overline{3}$.

(8)

Answer:

$$\mathsf{true} = \lambda x.\lambda y.x \qquad \mathsf{false} = \lambda x.\lambda y.y$$
$$\mathsf{xor} = \lambda\, u.\lambda v.u\,(v\,\mathsf{false}\,\mathsf{true})\,v$$
$$\textit{where } (v\,\mathsf{false}\,\mathsf{true}) \textit{ is the encoding of } (\mathsf{not}\,v)$$
$$\langle a, b\rangle = \lambda x.x\,a\,b$$
$$\overline{3} = \lambda f.\lambda x.f\,(f\,(f\,x))$$

(b) Consider the following λ-terms: $\mathsf{xor\text{-}pair}$ is a function on pairs of Booleans, $\mathsf{xor\text{-}fun}$ a function from Church Numerals to pairs of Booleans:

$$\mathsf{xor\text{-}pair} = \lambda p.\langle p\,\mathsf{false}, \mathsf{xor}\,(p\,\mathsf{true})\,(p\,\mathsf{false})\rangle$$
$$\mathsf{xor\text{-}fun} = \lambda n.n\,\mathsf{xor\text{-}pair}\,\langle\mathsf{true}, \mathsf{true}\rangle$$

What values do the following terms reduce to?

$$\mathsf{xor\text{-}pair}\,\langle\mathsf{true}, \mathsf{true}\rangle \rightsquigarrow^* ? \qquad \mathsf{xor\text{-}pair}\,\langle\mathsf{false}, \mathsf{true}\rangle \rightsquigarrow^* ?$$
$$\mathsf{xor\text{-}pair}\,\langle\mathsf{true}, \mathsf{false}\rangle \rightsquigarrow^* ? \qquad \mathsf{xor\text{-}pair}\,\langle\mathsf{false}, \mathsf{false}\rangle \rightsquigarrow^* ?$$

Show the steps of reduction in the computation of $(\mathsf{xor\text{-}fun}\,\overline{3})$. [You can use the previous reductions and those from part (a) as single steps.]

Give an informal definition of what $\mathsf{xor\text{-}fun}$ does: for which numbers n does $(\mathsf{xor\text{-}fun}\,\overline{n}) \rightsquigarrow^* \langle\mathsf{true}, \mathsf{true}\rangle$? (10)

Answer:

$$\mathsf{xor\text{-}pair}\,\langle\mathsf{true}, \mathsf{true}\rangle \rightsquigarrow^* \langle\mathsf{true}, \mathsf{false}\rangle$$
$$\mathsf{xor\text{-}pair}\,\langle\mathsf{true}, \mathsf{false}\rangle \rightsquigarrow^* \langle\mathsf{false}, \mathsf{true}\rangle$$
$$\mathsf{xor\text{-}pair}\,\langle\mathsf{false}, \mathsf{true}\rangle \rightsquigarrow^* \langle\mathsf{true}, \mathsf{true}\rangle$$
$$\mathsf{xor\text{-}pair}\,\langle\mathsf{false}, \mathsf{false}\rangle \rightsquigarrow^* \langle\mathsf{false}, \mathsf{false}\rangle$$

$$\begin{aligned}\mathsf{xor\text{-}fun}\,\overline{3} &\rightsquigarrow^* \overline{3}\,\mathsf{xor\text{-}pair}\,\langle \mathsf{true}, \mathsf{true}\rangle \\ &\rightsquigarrow^* \mathsf{xor\text{-}pair}\,(\mathsf{xor\text{-}pair}\,(\mathsf{xor\text{-}pair}\,\langle \mathsf{true}, \mathsf{true}\rangle)) \\ &\rightsquigarrow^* \mathsf{xor\text{-}pair}\,(\mathsf{xor\text{-}pair}\,\langle \mathsf{true}, \mathsf{false}\rangle) \\ &\rightsquigarrow^* \mathsf{xor\text{-}pair}\,\langle \mathsf{false}, \mathsf{true}\rangle \\ &\rightsquigarrow^* \langle \mathsf{true}, \mathsf{true}\rangle\end{aligned}$$

($\mathsf{xor\text{-}fun}\,\overline{n}$) *reduces to* $\langle \mathsf{true}, \mathsf{true}\rangle$ *if* n *is divisible by 3, to* $\langle \mathsf{true}, \mathsf{false}\rangle$ *if the remainder of division of* n *by 3 is 1, and to* $\langle \mathsf{false}, \mathsf{true}\rangle$ *if the remainder is 2.*

(c) In the context of complexity theory, explain what it means for a decision problem to belong to the classes: $\mathcal{P}$, $\mathcal{NP}$, $\mathcal{NP}$-complete. (7)

Answer:

- *A problem belongs to the class P if there is a Turing Machine that always terminates in polynomial time and decides each instance of the problem.*
- *It belongs to NP if either of the following equivalent conditions hold:*
  - *(i) There is a non-deterministic Turing Machine that runs in polynomial time and decides every instance of the problem.*
  - *(ii) There is a deterministic Turing Machine that runs in polynomial time and checks solutions to instances of the problem: when we give it as input an instance of the problem and a candidate solution, it terminates in an accepting states if and only if the solution is correct.*
- *A problem is NP-complete if it belongs to NP and every NP problem can be reduced in polynomial time to it.*