UNIVERSITY OF EDINBURGH

COLLEGE OF SCIENCE AND ENGINEERING

SCHOOL OF INFORMATICS

INFORMATICS 2A: PROCESSING FORMAL AND NATURAL LANGUAGES

August 19, 2010

14:30 to 16:30

Convener: J Bradfield
External Examiner: A Preece

INSTRUCTIONS TO CANDIDATES

1. **Candidates in the third or later year of study for the degrees of MA(General), BA(Relig Stud), BD, BCom, BSc(Social Science), BSc (Science) and BEng should put a tick (√) in the box on the front cover of the script book.**
2. **Answer Parts A and B. The short answer questions in Part A are worth 50% in total and are each worth the same amount. Part B contains THREE questions. Answer any TWO. Each is worth 25%. If you attempt all three questions, cross out one answer. If you do not, you will gain credit for two of the questions you have answered. The choice of questions gaining credit is at the discretion of the examination board.**
3. **Use a separate script book for part A and for each of the TWO questions from Part B. that you answer.**

Write as legibly as possible.
CALCULATORS ARE NOT PERMITTED.

Part A
ANSWER ALL QUESTIONS IN PART A. Use a separate script book for your answers to part A.

1. What is **tokenization** and what feature does a text need in order for it to be easily **tokenized**?

2. In what way does a text with a high **type/token ratio** look different from a text with a low **type/token ratio**?

3. Briefly describe two methods used by part-of-speech taggers to deal with **unknown words** (ie, words that didn't occur in the data used to train the tagger).

4. Given a context-free grammar, what is the relationship between a **derivation** of the string according to the grammar and a **tree diagram** for the string according to the same grammar?

5. Given a grammar rule

   A → B C

   briefly describe how it would be used in the **reduce** operation of a shift-reduce parser.

6. Name one type of ambiguity that increases the work load for a a chart parser. Briefly explain what extra work needs to be done.

7. Name one type of ambiguity that does **not** increase the work load for a chart parser. Briefly explain **why** the work load is not increased.

8. Simplify the following expression using both **alpha-conversion** and **beta-reduction**:

   λ y . λ x . love(x,y)(x)(john)

9. Briefly describe one property of **linear-indexed grammars** that is not shared by **context-free grammars**?

10. What does it mean for a grammar to be **weakly adequate** for a Natural Language?

11. A fellow student tells you he is proposing to construct a recursive descent parser for a grammar with the following rules:

    $$A \rightarrow BAa \mid c \quad B \rightarrow d \mid bB$$

    Do you think this is a good proposal? Justify your answer.

12. Given two *deterministic* FSMs $M_1 = (\Sigma_1, S_1, s_0^1, \delta_1)$ and $M_2 = (\Sigma_2, S_2, s_0^2, \delta_2)$. If we follow the standard construction for deriving the machine, M_3, that accepts the intersection of the languages accepted by $M_1$and M_2 is the machine M_3 deterministic?

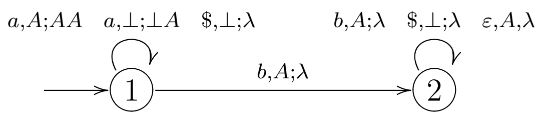
13. Suppose you are given a DFA M that has n states and the alphabet of M is Σ. You have been asked to find the machine $\bar{M}$ that accepts the *complement* of the language recognised by M. So, $\bar{M}$ accepts any member of Σ^* that is not accepted by M. How many states does $\bar{M}$ have? Justify your answer.

14. Consider a grammar with the following rules (the top symbol is S):

$$\begin{array}{rcl} S & \rightarrow & A \mid B \\ A & \rightarrow & a \mid aAa \\ B & \rightarrow & \varepsilon \mid aBa \end{array}$$

Is the grammar $LL(k)$ for any value of k? Justify your answer.

15. Consider the following Pushdown Automaton. The machine accepts with an empty stack and the initial symbol on the stack is $\perp$. In addition, we are using λ for the empty string of stack symbols and \$ is the end of input symbol. What is the language recognised by the machine? Justify your answer.



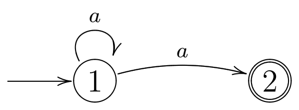
$a,A;AA$ $a,\perp;\perp A$ $\$,\perp;\lambda$
$b,A;\lambda$ $\$,\perp;\lambda$ ε,A,λ
$b,A;\lambda$
1
2

16. Consider the following two context-free grammars, G_1 and G_2 with the following sets of rules (S_1 is the top symbol for G_1 and S_2 is the top symbol for G_2. What is the language $L(G_1) \cap L(G_2)$? Is it context-free?

$$\begin{array}{rcl} S_1 & \rightarrow & AC \\ A & \rightarrow & \varepsilon \mid aAb \\ C & \rightarrow & cC \mid \varepsilon \end{array}$$

$$\begin{array}{rcl} S_2 & \rightarrow & A \\ A & \rightarrow & aAc \mid B \\ B & \rightarrow & bB \mid \varepsilon \end{array}$$

17. Consider following NFA:



Draw the finite automaton that accepts the complement of the language recognised by this machine, relative to the alphabet $\{a\}$.

18. Consider the CFG with the following rules. What is the set $\text{First}_1(A)$? Justify your answer.

$$\begin{array}{rcl} A & \rightarrow & aAb \mid CAd \mid \varepsilon \\ C & \rightarrow & c \mid \varepsilon \end{array}$$

19. Consider the following CFG. What is the set $\text{Follow}_1(B)$? Justify your answer.

$$\begin{array}{rcl} S & \rightarrow & A\$ \\ A & \rightarrow & Bc \mid BA \mid a \\ B & \rightarrow & Bb \mid \varepsilon \end{array}$$

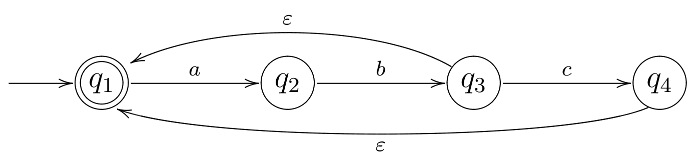
20. Is the grammar with the following rules *ambiguous*? A is the top symbol in the grammar. Justify your answer.

$$A \rightarrow aAa \mid aAb \mid c$$

Part B

ANSWER TWO QUESTIONS FROM PART B. Use a separate script book for each question you answer in part B.

1. Consider the following NFA M. M is a model of a simple system that repeatededly carries out the actions a followed by b, followed by c. Unfortunately, it also has a bug that means sometimes the c is omitted.



$$\begin{tikzcd} \arrow[r] & q_1 \arrow[r, "a"] & q_2 \arrow[r, "b"] & q_3 \arrow[r, "c"] \arrow[ll, bend right, "\varepsilon"'] & q_4 \arrow[lll, bend left, "\varepsilon"] \end{tikzcd}$$

   (a) Use the techniques of Kleene's theorem to convert this NFA to a Regular Expression. [*6 marks*]

   (b) The technician trying to fix the bug does not understand non-deterministic machines. Use the standard construction to convert M to a deterministic machine. [*6 marks*]

   (c) The technician's boss is fed up with how long it is taking to fix the bug. She argues that the bug doesn't happen very often and the system that uses this machine as a component works fine with an occasional omitted c. She proposes monitoring this by making a machine that accepts the language $L = \{x + c^{(\#_a(x) - \#_c(x))} \mid x \in \{abc, ab\}^*\}$. Strings in L comprise a string x in $\{abc, ab\}^*$ followed by a plus symbol followed by n c symbols where n is how many more as than cs appear in x. The technician thinks that this language could be recognised by a Finite State Automaton.

      i. Do you think the technician is right? [*1 marks*]

      ii. Write reasonably detailed notes to support your view. [*3 marks*]

      iii. Construct a machine that recognises the language L. [*3 marks*]

   (d) What kind of grammar would you use to specify the language L? Provide a brief outline of how you would expect the grammar to generate the language and provide a grammar for the language. [*6 marks*]

2. Given the grammar G1

G1	S → NP VP	NPR → John *(proper name)*
	NP → NPR	N → spider *(noun)*
	NP → N	N → plants *(noun)*
	NP → N N	N → seeds *(noun)*
	NP → PossNP NP	TV → plants *(transitive verb)*
	PossNP → NP Poss	Poss → 's *(possessive)*
	VP → TV NP	

and the sentence S1

S1: *John's spider plants seeds*

where *John's* is tokenized into the two tokens *John* and *'s*:

(a) Draw the parse trees for all analyses of **S1** according to grammar **G1**. [*5 marks*]

(b) Is the sentence **S1** unambiguous with respect to grammar **G1**, or locally ambiguous with respect to **G1**, or globally ambiguous with respect to **G1**? Briefly explain your answer. [*5 marks*]

(c) Assuming that a recursive descent (RD) parser applies the rules of **G1** in the given order, describe how an RD parser would proceed to parse sentence **S1**, up until the **first** point at which it would be forced to back up. [*5 marks*]

(d) Draw the chart that the CYK algorithm would construct for sentence **S1** based on grammar **G1**. [*5 marks*]

(e) For the initial step of the Earley (active chart parsing) algorithm, list the entries inserted into the chart, based on grammar **G1**, before anything in the input is examined. Then list what is entered into the chart by the algorithm when the first word of the input, *John*, is processed. [*5 marks*]

3. (a) Explain what it is for a context-free grammar to be *ambiguous*. What are the consequences of this for the efficiency of any parsing process for such languages? Ignore issues of lexical ambiguity: Assume each word belongs to only one class. *[3 marks]*

(b) Consider the following probabilistic context-free grammar. Where probabilities are not assigned explicitly, you can assume all rules for that nonterminal are equally probable. The rules of the grammar are (the top symbol is S):

$$\begin{array}{lcl} \mathrm{S} & \rightarrow & \mathrm{AS}\ [0.1] \mid \mathrm{SS}\ [0.8] \mid \mathrm{B}\ [0.1] \\ \mathrm{A} & \rightarrow & \mathrm{a} \mid \mathrm{b} \mid \mathrm{c} \\ \mathrm{B} & \rightarrow & \mathrm{b} \mid \mathrm{c} \end{array}$$

Using the above grammar, provide *two* derivation trees for the following sentence in the language, and provide a brief explanation of how the ambiguity arises:

abbc *[4 marks]*

(c) Calculate the probability for both your derivation trees and indicate the most likely structure for the sentence. Ignore probabilities for productions with A or B on the left hand side. These probabilities are not supplied. *[4 marks]*

(d) Redesign the above grammar by changing the probabilities on the productions so that the choice of more likely derivation is reversed. Recalculate the probabilities to demonstrate this is the case. *[6 marks]*

(e) What is the language generated by the grammar? Justify your answer. *[4 marks]*

(f) Devise a new, unambiguous, grammar that generates the same language as the original grammar. *[4 marks]*

Part A – Answers

1. What is **tokenization** and what feature does a text need, in order for it to be easily **tokenized**?

   Answer: Tokenization breaks a text consisting of a continuous sequence of characters into a sequence of discrete tokens, usually words. For a text to be easily tokenized, it must have obvious places – like white space or punctuation – where one can be easily recognized as separating the sequence of characters making up one token from those making up another.

   points: 1 point if tokenization is defined, but a necessary feature isn't specified.

2. In what way does a text with a high **type/token ratio** look different from a text with a low **type/token ratio**?

   Answer: The**Type/token ratio** is the ratio of the number of distinct word types to the total number of tokens in a text. (The text "buffalo buffalo buffalo" has a type/token ratio of 1/3, while "buffalo buffalo buffalo buffalo" has a type/token ratio of 1/4.) A text with a hight t/t ratio will contain alot of different words, while one with a low t/t ratio (as with one that only uses the 900 words of Basic English) will use the same words over and over again.

   points: 1 point if type/token ratio is defined, but nothing is mentioned about what texts with high and low t/t would look like.

3. Briefly describe two methods used by part-of-speech taggers to deal with **unknown words** (ie, words that didn't occur in the data used to train the tagger).

   Answer: Methods that might be mentioned include:

   - Skip over word without assigning a label;
   - Assign the word the most common part-of-speech (in English, "noun");
   - Use a regular expression to analyse the word and assign its type based on its internal structure.

   points: 1 point for each correct method.

4. Griven a context-free grammar, what is the relationship between a **derivation** of the string according to the grammar and a **tree diagram** for the string according to the same grammar?

   Answer: A derivation shows the sequence of grammar rules used in deriving the string from the start symbol. A tree diagram represents the set of equivalent derivations, independent of the order in which they apply.

   points: 1 point if the student just correctly explains a derivation. 0 points if the answer says that a tree diagram represents **all** derivations of the string.

5. Given a grammar rule

   A → B C

   briefly describe how it would be used in the **reduce** operation of a shift-reduce parser.

   Answer: A shift-reduce parser stores symbols in its stack. If popping the stack yields the symbols C and B, then A can be pushed onto the stack to replace them.

   Points: 1 point for knowing something about a shift-reduce parser. 2 points for full answer.

6. Name one type of ambiguity that increases the work load for a a chart parser. Briefly explain what extra work needs to be done.

   Answer: Types of ambiguity that a student might correctly mention here are:

   - Part-of-speech ambiguity, as each different PoS of a word instantiates must be added separately to the chart.
   - Attachment ambiguity (which allows the same word or phrase to attach at different parts of the growing tree), as each different attachment leads to a different entry in the chart.

   Points: 1 point for a type of ambiguity being named but not explained. 2 points for both.

7. Name one type of ambiguity that does **not** increase the work load for a chart parser. Briefly explain **why** the work load is not increased.

   Answer: All the types of ambiguity that a student can correctly mention here involve semantics:

   - lexical ambiguity, as a term with multiple meanings (all with the same part-of-speech) will only have a single entry in the chart;
   - referential ambiguity, as chart parsing ignores what a phrase refers to in the real world or the world created by the text;
   - scopal ambiguity, as chart parsing does not associate a text with its interpretation, and scope only affects the interpretation of a text.

   Points: 1 point for a type of ambiguity being named but not explained. 2 points for both.

8. Simplify the following expression using both **alpha-conversion** and **beta-reduction**:

   λ y . λ x . love(x,y)(x)(john)

Answer: λ y . λ x . love(x,y)(x)(john) $\Rightarrow$ λ y . λ z . love(z,y)(x)(john) $\Rightarrow$ λ z . love(z,x)(john) $\Rightarrow$ love(john,x)

Points: 0 points if student fails to rename the lambda variable x via **alpha-conversion** before carrying out the beta-reduction. 1 point, if they rename the variable but fail to get the correct answer. 2 points otherwise.

9. Briefly describe one property of **linear-indexed grammars** that is not shared by **context-free grammars**?

   Answer: A **linear-indexed grammar** is a weak form of context-sensitive grammar. One symbol in each rule can be indexed, and the index incremented or decremented via the rule application. This adds additional memory of what's been seen in the input. The symbols in CFGs don't have indices, so not as many input features can be remembered, so the grammar is less powerful.

   Points: 1 point for recognizing that LIGs are a form of CSGs.

10. What does it mean for a grammar to be **weakly adequate** for a Natural Language?

    Answer: It means that the grammar generates all and only the strings of the language, but doesn't necessarily assign them a correct structure from which their semantics can be computed.

11. This is a good proposal because there are no epsilon productions, the first sets of each production are obvious and disjoint so recursive descent will work efficiently with this grammar..

12. M_3 is deterministic because if a state (s_1, s_2) in the stateset of M_3 has an a transition then there is exactly one a transition for s_1 in M_1 and exactly one a transition from s_2 in M_2 because they are deterministic machines so there is is exactly one transition from (s_1, s_2) in M_3.

13. n because M is a DFA and to derive $\bar{M}$ all we need to do is change accepting states of M to non-accepting states in $\bar{M}$ and non-accepting states of M to accepting states of $\bar{M}$.

14. The grammar is not $LL(k)$ for any k. To make the choice of production from S we always need to consider the whole input to see if it is odd or even so the grammar cannot be $LL(k)$ for any k.

15. The machine accepts $\{a^n b^m \$ \mid n \geq m \geq 0\}$. In state 1 the stack is used to count the number of as input, on a b input the machine makes a transition to state 2 where it counts down for each b read until the initial stack symbol is popped when a \$ is read. In addition state 2 contains a transition that allows us to discard an A symbol fromthe stack so the number of bs can be smaller then the number of as.

16. The language is $a^n b^n c^n$, $n > 0$. The language is context-sensitive and not context-free.

17. This must be converted to a DFA before applying the algorithm. The complement of the language is $\{\varepsilon\}$ Th

18. $\{a, c, d, \varepsilon\}$

19. $\{c, b, a\}$

20. No it is not ambiguous because the choice of production is always determined by the last symbol in the production (so we can parse deterministically from right to left).

Part B – Answers

1. Question 1

   (a) Award three marks for constructing the system of equations and 3 marks for solving them. The equations are:

$$\begin{aligned} R_1 &= aR_2 + \varepsilon \\ R_2 &= bR_3 \\ R_3 &= cR_4 + \varepsilon R_1 \\ R_4 &= \varepsilon R_1 \end{aligned}$$

   Of the three points for solving the equations, award one point for finding that $R_1 = abcR_1 + abR_1 + \varepsilon$ and two points for solving the equation $R_1 = (abc + ab)R_1 + \varepsilon$ using the fact that $R = A^*B$ is the smallest solution to the equation $R = AR + B$. This gives us: $R_1 = (abc + ab)^*$.

   (b)

state	**a**	**b**	**c**
$\{q_1\}$	$\{q_2\}$		
$\{q_2\}$		$\{q_3, q_1\}$	$\{q_4, q_1\}$
$\{q_3, q_1\}$	$\{q_2\}$		$\{q_4, q_1\}$
$\{q_4, q_1\}$	$\{q_2\}$		

   Award 2 marks for an indication that the student knows the construction, three marks for the table, interpolated for inaccuracies, one mark for getting final states right.

   (c) i. The technician is wrong - one mark.

   ii. The language is not regular, this can be justified by looking at the dependencies and/or by outlining a pumping lemma proof. For example, if we choose $s = (ac)^k + b^k$, where k is parameter in the pumping lemma then this is a suitable choice of string. I'd award up to two marks for the observation that it is not regular and that the dependencies show this. An additional mark for some discussion of the use of the pumping lemma.

   iii. The easiest way to construct this is to use the original FSM as a starting point, extending the transitions to manipulate the stack so a symbol is pushed every time there is an ε transition from q_3 to q_1, then adding a transition labelled + from the final state of the original FSM to a machine that checks the number of bs read is eqal to the number of symbols pushed onto the stack in the initial phase. Award 1 mark for an awareness of PDA diagrams, one mark for pushing the symbols and one mark for checking the number of symbols on the stack.

(d) We would need to use a context-free grammar (one mark). The grammar generates matched sequences ab and c or the sequence abc with no matching c and does this iteratively (two marks). This is an example (award up to three marks):

$$S \rightarrow + \mid abcS \mid abSc$$

2. Question 2

   Given the grammar G1:

G1	S → NP VP	NPR → John *(proper name)*
	NP → NPR	N → spider *(noun)*
	NP → N	N → plants *(noun)*
	NP → N N	N → seeds *(noun)*
	NP → PossNP NP	TV → plants *(transitive verb)*
	PossNP → NP Poss	Poss → 's *(possessive)*
	VP → TV NP	

   and the sentence S1

   S1: *John's spider plants seeds*

   where *John's* is tokenized into the two tokens *John* and *'s*:

   (a) Draw the parse trees for all analyses of **S1** according to grammar **G1**.

   Answer: There is only ONE parse tree for the sentence, corresponding to the bracketting

   (S (NP (PossNP (NP (NPR John)) (Poss 's)) (NP (N spider))) (VP (TV plants)(NP (N seeds))))

   (b) Is sentence **S1** unambiguous with respect to grammar **G1**, or locally ambiguous with respect to **G1**, or globally ambiguous with respect to **G1**? Briefly explain your answer.

   Answer: The sentence is locally ambiguous because the initial substring *John's spider plants* has two analyses, both as an NP, and as an NP followed by a transitive verb.

   (c) Assuming that a recursive descent (RD) parser applies the rules of grammar G1 in the given order, describe how an RD parser would proceed to parse **S1**, up until the **first** point at which it would be forced to back up.

   Answer: The parser would apply the rules S → NP VP, NP → NPR, NPR →*John*, VP→TV NP, and TV → *plants*. Then it would discover that *plants* doesn't match the input token *'s*, and would have to back up, undoing the last four rules, and then choosing the next NP expansion rule.

   (d) Draw the chart that the CYK algorithm would construct for **S1** based on grammar G1.

   Answer:

	1	2	3	4	5
0	NP NPR	PossNP	NP		S
1		Poss			
2			NP N		
3				TV NP N	VP
4					NP N

(e) For the initial step of the Earley (active chart parsing) Algorithm, list the entries inserted into the chart, based on grammar G1, **before** any part of the input is examined. Then list what is entered into the chart by the algorithm when the first word of the input, *John*, is processed.

Answer: During the initial step, the following entries would be made:

i. $\gamma \rightarrow$ .S (*insert root symbol into the chart*)
ii. S $\rightarrow$. NP VP
iii. NP $\rightarrow$. NPR
iv. NP $\rightarrow$. N
v. NP $\rightarrow$. N N
vi. NP $\rightarrow$. PossNP NP
vii. PossNP $\rightarrow$. NP Poss

When the first word *John* is processed, the following entries would be added:

i. NPR $\rightarrow$ John .
ii. NP $\rightarrow$ NPR .
iii. S $\rightarrow$ NP . VP
iv. PossNP $\rightarrow$ NP . Poss

3. Question 3

   (a) One mark for the definitions of ambiguity. Two marks for a short discussion of diferent parsing approaches, their applicability to ambiguous grammars and the time efficiency of the different approaches.

   (b) The two parses differ on whether the S $\rightarrow$ S S is used or not. The use of this production gives rise to the less likely parse. The other is right linear and uses the first and last production only. Three marks for the parses and one marks for the explanation.

   (c) The probabilities for the parts of the parse that differ are: [[ab][bc]] – has a probability of $0.8 \times 0.8 \times 0.1 \times 0.1 \times 0.1 \times 0.1$ and [a [b [b c]]] has a probability of $0.1 \times 0.1 \times 0.1 \times 0.1$.

   (d) The approach here would be to reduce the probability of the first production for S very significantly, say to 0.01 and increase the probability of the last production to 0.19. Allocate 4 marks for the redesign and two marks for the recalculation.

   (e) The language generated is $((a+b+c)^*(b+c))^+$ because the second production just generates sequences of sentences in the language and the remaining productions describe the regular language $(a+b+c)^*(b+c)$.

   (f) Any grammar that generates the language. For example:

$$\begin{array}{lcl} \mathrm{S} & \rightarrow & \mathrm{AS} \mid \mathrm{BS} \mid \mathrm{B} \\ \mathrm{A} & \rightarrow & \mathrm{a} \mid \mathrm{b} \mid \mathrm{c} \\ \mathrm{B} & \rightarrow & \mathrm{b} \mid \mathrm{c} \end{array}$$

   Allocate 4 marks for a fully correct answer, deduct 1 mark for each distinct defect in the answer.