UNIVERSITY OF EDINBURGH

COLLEGE OF SCIENCE AND ENGINEERING

SCHOOL OF INFORMATICS

INFORMATICS 2A: PROCESSING FORMAL AND NATURAL LANGUAGES

August 18, 2009

14:30 to 16:30

Convener: M O'Boyle
External Examiner: R Irving

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 ($\surd$) 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. Using the alphabet **a**, **b**, **c**, create a regular expression with a binary dependency that can be realised in a regular grammar. Which are the elements that depend on one another?

2. Using the alphabet **a**, **b**, **c**, create a small context-free grammar (no more than 3 rewrite rules) that has a binary dependency that cannot be captured in a regular grammar. Which are the elements that depend on one another?

3. What does it mean to say that the words of a natural language occur in a *Zipfian distribution*?

4. Name a parsing algorithm that works by *dynamic programming*, and briefly explain how it belongs to this class of algorithms.

5. What does it mean for a language to have a compositional semantics?

6. What does it mean to *reduce* a lambda expression?

7. Name two forms of *context* that can be used in interpreting a natural language expression, and for each, give an example of how that form of context contributes to interpreting an expression of your choice.

8. What does it mean for a grammar to be *strongly adequate* for a Natural Language?

9. What does it mean for a probabilistic context-free grammar (PCFG) to be *consistent*?

10. What is currently assumed to be the *level of complexity* of a human language?

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

$$A \rightarrow BAa \mid a \qquad B \rightarrow \varepsilon \mid bB$$

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

12. Consider two FSMs $M_1 = (\Sigma_1, S_1, s_0^1, \delta_1)$ with n states and $M_2 = (\Sigma_2, S_2, s_0^2, \delta_2)$ with m states. 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, what is the state set of M_3 and what is its size in terms of m and n?

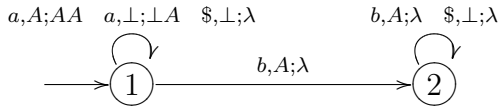
13. Suppose you are given an NFA M that has n states whose alphabet is Σ. You have been asked to find the machine $\bar{M}$ that accepts the *complement* of the language recognised by M. So, $\bar{M}$ accepts any member of Σ^* that is not accepted by M. What is the largest number of states in the resulting DFA? Justify your answer.

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

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

Is the grammar unambiguous? Justify your answer.

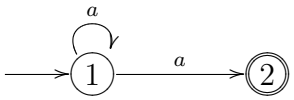
15. Consider the following Pushdown Automaton. The machine accepts with an empty stack and the initial symbol on the stack is $\perp$. In addition, we are using λ for the empty string of stack symbols and \$ is the end of input symbol. What is the language recognised by the machine? Justify your answer.



16. Consider the following Context-Sensitive Grammar. What is the language generated by this grammar? Justify your answer.

$$\begin{aligned} S &\rightarrow SABC \mid ABC \\ AB &\rightarrow BA \\ AC &\rightarrow CA \\ CA &\rightarrow AC \\ BA &\rightarrow AB \\ BC &\rightarrow CB \\ CB &\rightarrow CB \\ A &\rightarrow a \\ B &\rightarrow b \\ C &\rightarrow c \end{aligned}$$

17. Consider the following NFA:



Draw the finite automaton that accepts the complement of the language recognised by this machine.

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

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

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

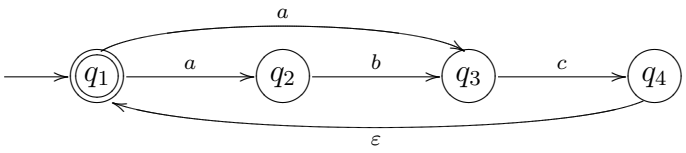
$$\begin{aligned} S &\rightarrow A\$ \\ A &\rightarrow Bc \mid BA \mid \varepsilon \\ B &\rightarrow Bb \mid \varepsilon \end{aligned}$$

20. Is the grammar with the following rules $LL(1)$? 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 repeatedly carries out the actions a followed by b, followed by c. Unfortunately, it also has a bug that means sometimes the b is omitted.



   (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. He 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 b. He proposes monitoring this by making a machine that accepts the language $L = \{x + b^{(\#_a(x)-\#_b(x))} \mid x \in \{abc, ac\}^*\}$. Strings in L comprise a string x in $\{abc, ac\}^*$ followed by a plus symbol followed by n b symbols where n is how many more as than bs appear in x. The technician thinks that this language could be recognised by a Finite State Automaton.

      i. Is the technician right? Write short notes to justify your view. [*4 marks*]

      ii. 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. Consider Grammar G1:

S → NP VP	Poss → her *(possessive)*
NP → Poss N	N → chicken *(noun)*
NP → N	DV → cooked *(ditransitive verb)*
NP → Pro	TV → cooked *(transitive verb)*
VP → TV NP	TV → saw *(transitive verb)*
VP → DV NP NP	Pro → you | her *(pronoun)*

(a) Draw the parse trees for all possible analyses of the sentence *"You cooked her chicken"* according to grammar G1. [*5 marks*]

(b) Draw the chart that the CYK algorithm would construct for the sentence *"You cooked her chicken"* based on grammar G1. [*5 marks*]

(c) Specify in one or two sentences what needs to be added to a CYK chart in order to extract parse trees from it. Draw a new version of the chart you did for part (b) with this added to it, such that you get the same parse trees as you drew for part (a). [*8 marks*]

(d) We can also use a chart to record the semantic interpretation of a string as it is being simultaneously parsed and interpreted. Consider the following semantic attachments to some rules from G1:

S[sem=<app(?np,?vp)>] → NP[sem=?np] VP[sem=?vp]
VP[sem =<app(?v, ?obj)>] → TV[sem=?v] NP[sem=?obj]
NP[sem=<app(?det,?n)>] → Poss[sem=?det] N[sem=?n]
NP[sem=?pro] → Pro[sem=?pro]

Poss[sem=< \Q P . exists x . (Q(x) & belongs_to(x, ref(she)) & P(x))>] → 'her'
N[sem=<chicken>] → 'chicken'
TV[sem=< \Q y . Q(\x . saw(y,x))>] → 'saw'
Pro[sem=< \P . P(you)>] → 'you'
Pro[sem=< \P . P(ref(she))>] → 'her'

Using these rules and semantic attachments, draw a new version of the chart that records the semantic interpretation of each phrase that is recognized when parsing the sentence *"You saw her chicken"*. For example, you would start by recording the interpretation of "you" (\P . P(you)) in element (0,1) of the chart. [*7 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 for a fragment of English, which is intended to parse a subset of descriptions of fashion items. Where probabilities are not assigned explicitly, you can assume all rules for that nonterminal are equally probable. The rules of the grammar are:

$$
\begin{array}{rcl}
\text{N} & \rightarrow & \text{Det Nom [0.5]} \mid \text{Nom [0.5]} \\
\text{Nom} & \rightarrow & \text{Nom Conj Nom [0.1]} \mid \text{Nom Nom [0.4]} \\
 & \rightarrow & \text{Adj Nom [0.1]} \mid \text{Noun [0.4]} \\
\text{Conj} & \rightarrow & \textit{and} \mid \textit{with} \mid \textit{or} \\
\text{Det} & \rightarrow & \textit{that} \mid \textit{this} \mid \textit{a} \mid \textit{the} \\
\text{Noun} & \rightarrow & \textit{fur} \mid \textit{leather} \mid \textit{bag} \mid \textit{skirt} \mid \textit{waders} \\
\text{Adj} & \rightarrow & \textit{fur} \mid \textit{leather} \mid \textit{clutch} \mid \textit{pink} \mid \textit{yellow}
\end{array}
$$

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

pink leather clutch bag

[*8 marks*]

(c) Calculate the probability for both your derivation trees and indicate the most likely structure for the fashion item description given above. Ignore probabilities for word choices from the nonterminals Conj, Det, Noun and Adj. These probabilities are not supplied. [*4 marks*]

(d) Provide brief comments on whether you think the parse with the highest probability matches the informal understanding of the term and if appropriate suggest how you might change the probabilities on the rules. [*3 marks*]

(e) Design a new, *unambiguous*, context-free grammar that generates the same language as the grammar above. Do not worry about ambiguity between Adj and Noun, you should not change the rules for the nonterminals Adj and Noun. [You do not need to allocate probabilities to the rules because the grammar is unambiguous.] [*7 marks*]

Part A – Answers

1. (ab*c)* Here a and c depend on one another.

2. S $\longrightarrow$ aAb
   A $\longrightarrow$ aAb | c

   Here a and b depend on one another.

3. It means that text observes a power law, and the frequency of a word in a corpus of texts is inversely proportional to its rank. There is a small set of frequently occurring words and a "long tail" of words that occur only very infrequently.

4. Both the CYK and the Earley algorithm belong to the class of dynamic programming algorithms by virtue of the fact that they break the problem of parsing the sentence as a whole into smaller problems of parsing the string of phrases that make it up, and the problem of parsing the phrases, into that of parsing the string of their subphrases.

5. In a language with a compositional semantics, the meaning of each sub-tree that makes up a sentence is formed from the meaning of its constituents.

6. Given the application of a lambda expression to a term, reduction involves binding its formal parameter to the term and simplifying the result.

7. The speaker's spatio-temporal environment (or that shared by speaker and listener) can serve as context for interpreting deictic expressions such as "here", "you", "now". The hearer and the speaker's shared knowledge can serve as context for interpreting definite referring expresions such as "the president", "the queen", "the university", etc. The current discourse can serve as context for interpreting reduced expressions such as pronouns and definite NPs.

8. A grammar that is strongly adequate for a language can generate all the strings of the language and assign them each a structure that supports its appropriate semantic interpretation.

9. With a consistent PCFG, the probability of all the sentences it yields sums to 1.

10. Human language is assumed to be "mildly context sensitive", but "context sensitive" and the complexity of linear indexed grammars would also be acceptable here.

11. This is a poor proposal because the grammar is left recursive and so a recursive descent parser would loop indefinitely because it could not determine how many times to apply the first production in analysing a string.

12. The stateset of M_3 is $S_1 \times S_2$ so the size of the stateset is nm.

13. 2^n because in order to find the complement we must convert M to a DFA and the stateset of the DFA is the powerset of the original NFA (but some of those states my be inaccessible from the initial state).

14. The grammar is unambiguous. A generates strings of odd length while B generates strings of even length.

15. The machine accepts $\{a^n b^n \$ \mid n \geq 0\}$. 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.

16. The language generated is any string s of as, bs and cs where the numbers of as, b and cs are equal and greater than zero. The first rule generates a non empty sequence of the form $ABCABC\ldots$, the next six rules allow us to permute As, Bs and Cs and the final three rules convert from nonterminals to terminals.

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

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

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

20. No it is not $LL(1)$ because the First_1 sets for the first two productions of A have non-empty intersection.

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:

$$\begin{aligned} R_1 &= aR_2 + aR_3 + \varepsilon \\ R_2 &= bR_3 \\ R_3 &= cR_4 \\ R_4 &= \varepsilon R_1 \end{aligned}$$

   Of the three points for solving the equations, award one point for finding that $R_1 = abcR_1 + acR_1 + \varepsilon$ and two points for solving the equation $R_1 = (abc + ac)R_1 + \varepsilon$ using the fact that $R = A^*B$ is the smallest solution to the equation $R = AR + B$. This gives us: $R_1 = (abc + ac)^*$.

   (b)

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

   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 $s = a^k c^k + b^k$, 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 a transistion from q_1 to q_3, 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 ac and b or the sequence abc with no matching b and does this iteratively (two marks). This is an example (award up to three marks):

$$S \rightarrow + \mid abcS \mid acSb$$

2. The sentence *"You cooked her chicken"* is ambiguous with respect to Grammar G1:

Grammatical rules	**Lexical rules**
S → NP VP	Poss → her *(possessive)*
NP → Poss N	N → chicken *(noun)*
NP → N	DV → cooked *(ditransitive verb)*
NP → Pro	TV → cooked *(transitive verb)*
VP → TV NP	TV → saw *(transitive verb)*
VP → DV NP NP	Pro → you | her *(pronoun)*

(a) Draw the parse trees for all possible analyses of *"You cooked her chicken"* according to grammar G1.

```
(S (NP (Pro you))
   (VP (TV cooked)
       (NP (Poss her)
           (N chicken))))

(S (NP (Pro you))
   (VP (DV cooked)
       (NP (Pro her))
       (NP (N chicken))))
```

(b) Draw the chart that the CYK algorithm would construct for the sentence *You cooked her chicken* based on grammar G1.

	1	2	3	4
0	Pro NP		S	S
1		TV DV	VP	VP
2			Pro Poss NP	NP
3				N NP

(c) Briefly indicate (1-2 sentences) what needs to be added to a CYK chart in order to extract parse trees from it. Draw a new version of the chart you did for part (b) with this added to it, such that you get the same parse trees as you drew for part (a).

To extract parse trees, every phrasal entry in the chart needs to keep track of the entries that enabled it to be entered into the chart. This can be done through pointers or explicit indexing. For convenience in latexing, the latter

is used below. The students will probably use pointers, since they're easier to draw.

	1	2	3	4
0	Pro (you) NP (0:1 (elt=1))		S (0:1 (elt=2), 1:3 (elt=1))	S (0:1 (elt=2), 1:4 (elt=1)) S (0:1 (elt=2), 1:4 (elt=2))
1		TV (cooked) DV (cooked)	VP (1:2 (elt=1), 2:3 (elt=1))	VP (1:2 (elt=1), 2:4 (elt=1)) VP (1:2 (elt=1), 2:3 (3rd elt), 3:4 (elt=2)))
2			Pro (her) Poss (her) NP (2:3 (elt=1))	NP (2:3 (elt=2), 3:4 (elt=1)
3				N (chicken) NP (3:4 (elt=1))

(d) We can also use a chart to record the semantic interpretation of a string as it is being simultaneously parsed and interpreted. Using the same notation as in Assignment 3, consider the following semantic attachments to a subset of G1.

S[sem=<app(?np,?vp)>] → NP[sem=?np] VP[sem=?vp]
VP[sem =<app(?v, ?obj)>] → TV[sem=?v] NP[sem=?obj]
NP[sem=<app(?det,?n)>] → Poss[sem=?det] N[sem=?n]
NP[sem=?pro] → Pro[sem=?pro]

Poss[sem=< \Q P . exists x . (Q(x) & belongs_to(x, ref(she)) & P(x))>] → 'her'
N[sem=<chicken>] → 'chicken'
TV[sem=< \Q y . Q(\x . saw(y,x))>] → 'saw'
Pro[sem=< \P . P(you)>] → 'you'
Pro[sem=< \P . P(ref(she))>] → 'her'

Using these rules and semantic attachments, draw a new version of the chart that records the instantiated semantic interpretations (ie, rather than phrase types and parts of speech as in part(b) above) that would be constructed during the parsing of *"You saw her chicken"*.

	1	2	3	4
0	\P . P(you)		saw(you,ref(she))	exists x . (chick(x) & belongs_to(x, ref(she)) & saw(you,x))
1		\Q y . Q(\x . saw(y,x))	\y . saw(y,ref(she))	\y . exists x . (chick(x) & belongs_to(x, ref(she)) & saw(y,x))
2			\P . P(ref(she)) \Q P . exists x . (Q(x) & belongs_to(x, ref(she)) & P(x))	\P . exists x . (chick(x) & belongs_to(x, ref(she)) & P(x))
3				chick

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 Nom → Nom Nom is used or not. This is the less natural parse. The other is right linear and parses the first three words as adjectives – this is appropriate in this case. Three marks for each parse and two marks for the explanation.

(c) The probabilities for the parts of the parse that differ are: [[pink leather][clutch bag]] – has a probability of $0.4 \times 0.4 \times 0.4 \times 0.1 \times 0.1$ and [pink [leather [clutch bag]]] has a probability of $0.1 \times 0.1 \times 0.1 \times 0.4$.

(d) The higher probability for the first parse is not particularly natural and arises because of the relatively low probability assigned to the Nom → Adj Nom rule. One mark for the "not natural" answer and two marks for a reasonable suggestion on how to change the probabilities.

(e) One approach is to make the probability of the Nom → Adj Nom higher, the probability of the Nom → Adj Nom rule is low and also increase the probability of theNom → Noun rule. Switching the probabilities of the two rules changes the priority.

(f) The following grammar is unambiguous and generates the same sentences:

$$
\begin{array}{rcl}
\text{N} & \rightarrow & \text{Det Nom} \mid \text{Nom} \\
\text{Nom} & \rightarrow & \text{N' COpt} \\
\text{COpt} & \rightarrow & \text{Conj Nom} \mid \varepsilon \\
\text{N'} & \rightarrow & \text{Adj N'} \\
 & \rightarrow & \text{Noun N'Opt} \\
\text{N'Opt} & \rightarrow & \text{N'} \mid \varepsilon
\end{array}
$$

Full marks for a correct answer, four marks for one with minor errors, two marks for an unambiguous attempt that is close to an answer.