UNIVERSITY OF EDINBURGH

COLLEGE OF SCIENCE AND ENGINEERING

SCHOOL OF INFORMATICS

INFR08008 INFORMATICS 2A: PROCESSING FORMAL AND NATURAL LANGUAGES

Wednesday 12th August 2015

09:30 to 11:30

INSTRUCTIONS TO CANDIDATES

1. **Answer all five questions in Part A, and two out of three questions in Part B. Each question in Part A is worth 10% of the total exam mark; each question in Part B is worth 25%.**
2. **Use a single script book for all questions.**
3. **Calculators may be used in this exam.**

Convener: D. K. Arvind
External Examiner: C. Johnson

THIS EXAMINATION WILL BE MARKED ANONYMOUSLY

PART A

Answer ALL questions in Part A

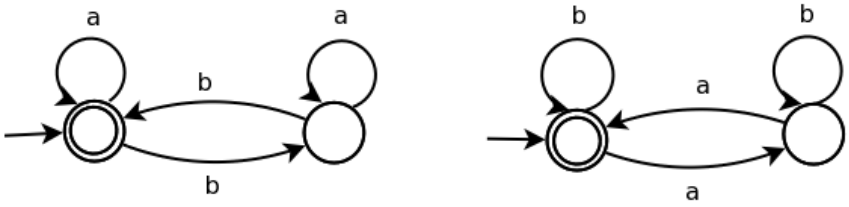
1. (a) Three stages in the language processing pipeline, for a statically-typed programming language such as Haskell, are *lexing*, *parsing*, and *type-checking*. Briefly describe what the input to and output from each stage is. Illustrate your answers with tiny but well-chosen examples. [*9 marks*]

   (b) Name one other stage in the language processing pipeline of a statically-typed programming language such as Haskell. [*1 mark*]

2. Suppose M_1 and M_2 are deterministic finite automata (DFAs), over the alphabet $\Sigma = \{a, b\}$, recognising the languages L_1 and L_2 respectively.

   (a) Using M_1 and M_2, describe how to construct a DFA that recognises the language $L_1 \cap (\Sigma^* - L_2)$ of all strings that are in L_1 but not in L_2. [*4 marks*]

   (b) Illustrate your construction from part (b) in the case that M_1 and M_2 are the two DFAs below.



[*4 marks*]

   (c) Suppose $L \subseteq \Sigma^*$ is a language generated by a context-free grammar (CFG) with set of terminals Σ. Is it possible, in general, to find a CFG that generates the language $\Sigma^* - L$? Briefly justify your answer. [*2 marks*]

3. Consider a pushdown automaton (PDA) with two control states $Q = \{q1, q2\}$, start state $q1$, input alphabet $\Sigma = \{a, b\}$, stack alphabet $\Gamma = \{\bot, a\}$ (where $\bot$ is the start symbol), and transition relation:

$$q1 \xrightarrow{a, \bot \,:\, a} q1 \qquad q1 \xrightarrow{a, a \,:\, aa} q1 \qquad q1 \xrightarrow{b, a \,:\, a} q2$$
$$q2 \xrightarrow{a, a \,:\, \epsilon} q2 \qquad q2 \xrightarrow{\epsilon, a \,:\, \epsilon} q2$$

   The automaton accepts on empty stack. (In the above description, we use the general notation

$$q \xrightarrow{c, x \,:\, \alpha} q'$$

   to mean that when the automaton is in control state $q \in Q$ and $x \in \Gamma$ is popped from the top of the stack, the input symbol or empty string $c \in \Sigma \cup \{\epsilon\}$ can be read to reach control state $q' \in Q$ with $\alpha \in \Gamma^*$ pushed onto the stack.)

QUESTION CONTINUES ON NEXT PAGE

(a) Describe in detail an execution of the above PDA that accepts the string

aaaba

[*8 marks*]

(b) Give a concise mathematical definition of the language L recognised by the PDA above. [*2 marks*]

4. (a) State *Zipf's law* for the frequency of word occurrences within a typical corpus of natural language text. [*2 marks*]

(b) Suppose that in a corpus that conforms closely to Zipf's law (even for the most common word types), the four most common word types account for 2500 of the tokens. Estimate how many tokens are covered, in total, by the 10 word types with frequency rank in the range 21–30 inclusive. An approximate calculation is acceptable. [*3 marks*]

(c) The Viterbi tagging algorithm requires a matrix of *transition probabilities* and another matrix of *emission probabilities*. Explain briefly how such probabilities may be extracted from a corpus of tagged text. [*3 marks*]

(d) Suppose that we have extracted probabilities from a given corpus as in part (c), and now run the Viterbi algorithm on the text of that same corpus. Would you expect a higher proportional accuracy rate (i.e. proportion of correct taggings) for occurrences of the 10 most frequent word types, or the 10 least frequent ones? Briefly justify your answer. [*2 marks*]

5. Consider the following context-free grammar with start symbol S:

$$
\begin{array}{rcl}
\mathsf{S} & \rightarrow & \mathsf{NP\ VP} \\
\mathsf{NP} & \rightarrow & \mathsf{NP\ NP} \\
\mathsf{VP} & \rightarrow & \mathsf{V\ NP} \\
\mathsf{NP} & \rightarrow & \text{boat} \mid \text{man} \mid \text{watches} \mid \text{fish} \\
\mathsf{V} & \rightarrow & \text{man} \mid \text{watches} \mid \text{fish}
\end{array}
$$

Construct a CYK parse chart for the sentence

boat man watches fish

Beside each entry in the chart, write the number of possible ways of parsing the corresponding word sequence that yield the non-terminal in question. For example, if the segment 'man watches fish' can be analysed as NP via three different parse trees, you should write NP 3 in the appropriate cell. If a cell contains entries for several different non-terminals, each of these should be annotated separately with the corresponding number of parse trees.

You should take account of all possible analyses of segments, whether or not they contribute to a parse of the complete sentence. [*10 marks*]

PART B

Answer TWO questions in Part B

6. The context-free grammar below has a single nonterminal S, and terminals:

$$\mathsf{a} \quad \mathsf{b} \quad (\quad)$$

The productions are:

$$\mathsf{S} \rightarrow \mathsf{a} \mid \mathsf{S}\,\mathsf{b} \mid (\,\mathsf{S}\,\mathsf{S}\,)$$

(a) This grammar is *not* LL(1). Justify this assertion. [*1 mark*]

(b) Write out an LL(1) grammar that generates the same language as the grammar above. [*5 marks*]

(c) Write out the *first set* of every nonterminal in your LL(1) grammar. [*3 marks*]

(d) Write out the *follow set* of every nonterminal in your LL(1) grammar. [*3 marks*]

(e) Write out the parse table for your LL(1) grammar. [*6 marks*]

(f) Describe the step-by-step execution of the LL(1) predictive parsing algorithm, when parsing the expression below using your parse table from (e) above.

(a a)

[*7 marks*]

7. In English morphology, a simple version of the *Y-replacement* rule says that the letter y is replaced by ie if it occurs at the end of a stem, is immediately preceded by any letter except a,e,i,o,u, and is immediately followed by the suffix s.

   (a) Design a finite state transducer that applies the Y-replacement rule to translate from an intermediate (i.e. segmented) word form to a surface form, e.g.

$$\begin{aligned} horse\widehat{\ }s\# &\rightarrow horses\# \\ pony\widehat{\ }s\# &\rightarrow ponies\# \\ donkey\widehat{\ }s\# &\rightarrow donkeys\# \\ ass\# &\rightarrow ass\# \end{aligned}$$

   Your transducer should have input alphabet $\{a,\ldots,z,\widehat{\ },\#\}$, where $\widehat{\ }$ denotes a morpheme boundary and # denotes a word boundary, and output alphabet $\{a,\ldots,z,\#\}$. The replacement rule should be applied only when the y in question is immediately followed by $\widehat{\ }s\#$. In contrast to the definition of finite state transducers given in lectures, it is permissible for your transducer to output a sequence of several characters on a single transition.

   You may use abbreviating conventions in your presentation of the transducer, but should include an explanation of these in your answer. [*12 marks*]

   (b) Now suppose that your transducer were applied *in reverse* in order to split surface forms into their constituent morphemes. For each of the three words in the sentence

$$he\#\ plies\#\ trades\#$$

   list all possible intermediate forms that the transducer may output. [*4 marks*]

QUESTION CONTINUES ON NEXT PAGE

(c) Suppose that our lexicon contains the following entries for the POS categories PRN, N, V:

PRN	*he*
N	*plie*, *trade*
V	*hear*, *ply*, *trade*

(The example is somewhat artificial: *plie* enters the lexicon as the noun *plié* (denoting a position in ballet) with the accent stripped off.) Suppose now that the outputs from part (b) are passed through a further (non-deterministic) transducer that replaces each word stem by its possible POS categories, and furthermore applies the transformations

$$\text{N}\,\widehat{}\,s \;\Rightarrow\; \text{NPL} \qquad \text{V}\,\widehat{}\,s \;\Rightarrow\; \text{V3S}$$

Thus, for example, this second transducer will convert the intermediate form *hear* to V, and *hear*$\widehat{}\,s$ to V3S; in these cases these are the unique possible outputs.

For each of the three words

he# plies# trades#

list all possible POS tags that may result once both the above phases have been applied. [*3 marks*]

(d) We may now use a version of the Viterbi algorithm to determine the most probable sequence of tags for the above phrase, given the set of possibilities established by part (c). We here use a simplified form of the algorithm that is concerned purely with tags, not with words, so that no emission probabilities are involved.

Determine the most probable tag sequence for *he plies trades* that is consistent with your solution to (c), using the following transition probability matrix. Show your working, and include backtrace pointers in your Viterbi matrix.

	to PRN	to N	to NPL	to V	to V3S
from start	0.4	0.2	0.3	0.1	0.0
from PRN	0.1	0.1	0.1	0.3	0.4
from N	0.1	0.2	0.1	0.1	0.5
from NPL	0.1	0.1	0.2	0.5	0.1
from V	0.2	0.2	0.5	0.1	0.0
from V3S	0.2	0.2	0.5	0.1	0.0

[*6 marks*]

8. This question concerns the semantics of a language for describing family relationships. Consider the following context-free grammar with start symbol S.

S	→	Name is NP
NP	→	Name
NP	→	NP 's Rel
Rel	→	Rel1
Rel1	→	father | daughter | brother | grandchild
Name	→	Alice | Brian | Calum | Diana

The reason for the two non-terminals Rel and Rel1 will emerge in part (d) below.

This grammar allows us to generate sentences such as 'Alice is Brian's daughter'. We shall here consider this to be saying only that Alice is a daughter of Brian's, with no implication that she is the *only* daughter. Likewise, the sentence 'Calum is Diana's brother's grandchild' leaves open the possibility that Diana has several brothers, each of whom has several grandchildren.

For the purpose of our semantics, we shall use two base types: the usual type t of *truth values*, and a type p of *people* — for the latter, we have the logical constant symbols *Alice*, *Brian*, *Calum*, *Diana*. Our logical language will also include unary relations *Male*, *Female* and binary relations *Father*, *Mother* on the type p: for example, we may write $Male(x)$ to mean that x is male, and $Mother(y, z)$ to mean that y is the mother of z. We also have the equality predicate $=$ on the type p.

We wish to give a semantics that associates a term of type t with a complete sentence (i.e., a phrase of category S). Phrases of category NP should receive interpretations of type $<p, t>$, since, for example, 'Brian's daughter' denotes not a particular person, but rather a property of people (several people might fit the description). Phrases of category Rel and Rel1 should receive interpretations of type $<p, <p, t>>$, since these denote relationships between two people.

(a) Provide semantic attachments for each of the rules of the above grammar so as to assign the intended meaning to each phrase in a compositional way. You may use the usual notations of first order logic and λ-calculus for this purpose. The following two examples are given to get you started (you need not copy these out):

NP →	Name	$\{\lambda x.\, x = \mathsf{Name.Sem}\}$
Rel1 →	daughter	$\{\lambda xy.\, Female(x)\ \&\ (Father(y, x) \vee Mother(y, x))\}$

[12 marks]

(b) Draw the parse tree for the sentence 'Brian is Alice's father', allowing plenty of room for annotations. Then annotate each node with the interpretation given by your semantics from part (b). You should not perform any β-reductions at this stage. *[7 marks]*

QUESTION CONTINUES ON NEXT PAGE

(c) Show explicitly how the λ-expression associated with the S node in part (c) reduces to a normal form via a sequence of β-reduction steps. [*3 marks*]

(d) Now suppose that we augment our grammar with the new rule

$$\mathsf{Rel} \quad \rightarrow \quad \text{only } \mathsf{Rel1}$$

Given any binary relation $R(x, y)$, write a logical formula that defines a new relation $\overline{R}(x, y)$, which holds exactly when x is the *only* element w such that $R(w, y)$ holds. Use this idea to provide an appropriate semantic attachment for the above clause. [*3 marks*]