UNIVERSITY OF EDINBURGH

COLLEGE OF SCIENCE AND ENGINEERING

SCHOOL OF INFORMATICS

INFR08008 INFORMATICS 2A: PROCESSING FORMAL AND NATURAL LANGUAGES

Monday 15 th August 2016

14:30 to 16:30

INSTRUCTIONS TO CANDIDATES

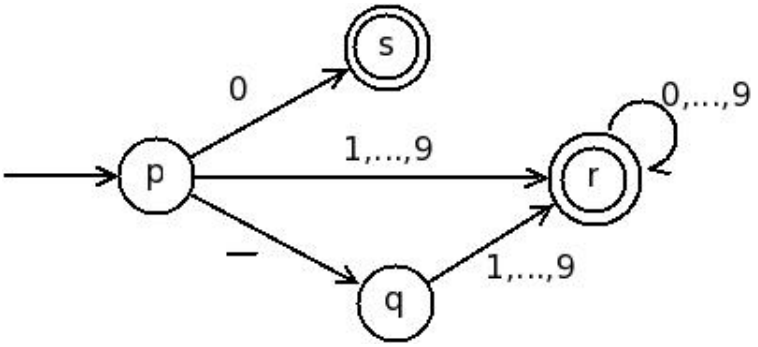
1. **Answer all five questions in Part A, and two out of three questions in Part B. Each question in Part A is worth 10% of the total exam mark; each question in Part B is worth 25%.**
2. **Use a single script book for all questions.**
3. **Calculators may be used in this exam.**

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

THIS EXAMINATION WILL BE MARKED ANONYMOUSLY

Part A

1. Consider the following NFA for defining a class of *integer literals*. The input alphabet is $\{-, 0, 1, \ldots, 9\}$.



   (a) Write down a set of equations expressing the relationships between the languages X_p, X_q, X_r, X_s associated with the states shown above. (To start you off, we may note immediately that $X_s = \{\epsilon\}$.) [*4 marks*]

   (b) Use Arden's rule to solve these equations and thus obtain regular expressions for each of X_p, X_q, X_r. [*3 marks*]

   (c) Suppose now that we use our NFA to lex the input string

$$0010-3450-089$$

   as a sequence of integer literals using the principle of longest match (i.e. maximal munch). List the sequence of lexical tokens that this yields, and indicate the point, if any, at which a lexing error will occur. [*3 marks*]

2. (a) List three common reasons why a given context-free grammar might fail to be LL(1). In each case, give an example of either a single production or a pair of productions that manifest the problem (you need not set up a whole grammar of which these productions are a part). [*6 marks*]

   (b) Convert the following grammar (with start symbol S) to an equivalent LL(1) grammar.

$$\begin{array}{lcl} \mathsf{S} & \rightarrow & \mathsf{T} \mid \mathsf{S} + \mathsf{S} \\ \mathsf{T} & \rightarrow & id \mid id\,(\,\mathsf{S}\,) \end{array}$$

[*4 marks*]

3. (a) Describe the three criteria for deciding whether two words belong to the same part-of-speech class. [*3 marks*]

   (b) Describe the difference between open word classes and closed word classes. Give an example of two open part-of-speech classes, and two closed ones. [*4 marks*]

   (c) Consider the following table of words that appeared in a corpus:

Rank	Word	Frequency
1	the	360000
2	that	180000
3	for	120000
4	is	90000
5	said	72000
6	on	60000
8	%	45000
9	it	40000
10	by	36000
12	from	30000
15	million	24000
16	at	22500
18	as	20000
20	with	18000
24	the	15000
25	was	14400

   Do the frequencies satisfy Zipf's law in this case? If not, explain why not. If they do, write down the formula that describes the relationship between the frequency and the rank as given by Zipf's law. [*3 marks*]

4. (a) Write as many logical forms as you can find for different interpretations of the sentence "Every boy has access to a door". Do the same for the sentence "Every boy has access to Door 3" (where you may treat the phrase "Door 3" as atomic). [*3 marks*]

   (b) Consider the sentence

   ```
   Boys access doors
   ```

   Apply the Viterbi algorithm on it and find the best POS sequence assuming the following hidden Markov model tables:

   Emission matrix:

	boys	access	doors
N	0.4	0.2	0.4
V	0.05	0.9	0.05

   Transition matrix (rows denote the current part of speech, and columns denote the part of speech we transition to):

	N	V
$\langle s \rangle$	0.7	0.1
N	0.1	0.4
V	0.5	0.1

   [*7 marks*]

5. A *palindrome* is a string that reads the same both forward and backwards.

   (a) Write Python code for a method called `reverse` which takes a string as an argument and reverses it. [*4 marks*]

   (b) Add a method called `isPalindrome` that uses the reverse function to tell whether a string is a palindrome or not. [*3 marks*]

   (c) Is the language of palindromes over $\{a, b\}$ regular? Give a brief explanation for why or why not (you need not give a full mathematical proof). [*3 marks*]

Part B

6. Consider the alphabet $\Sigma = \{e, s, w\}$, where the three symbols stand for *eat*, *sleep*, *work* respectively. We shall consider the language defined by the following regular expression (given here in mathematical notation):

$$((w(we)^*s) \ + \ (w(ws)^*e))^*$$

   (a) Draw the state diagram for an NFA (without ϵ-transitions) that accepts the above language. Your NFA should involve exactly 5 states, which you should label as 0,1,2,3,4 in any order. You are advised to construct your NFA directly by inspection of the above regular expression, rather than by following a general algorithm precisely. [*6 marks*]

   (b) Apply the subset construction to your NFA from part (a) to convert it into a DFA that accepts the same language. You need not include unreachable states in your diagram. You should indicate which state in your DFA serves as the 'garbage state' G. To reduce clutter, you need not display all transitions from non-garbage states to G, though you should include at least one such transition. [*9 marks*]

   (c) Given any state q in a DFA M, we may write L_q for the set of all strings that take us from q to an accepting state in M. With reference to the sets L_q, give a mathematical criterion for a DFA M to be *minimal*. [*3 marks*]

   (d) Is your DFA from part (b) minimal? If so, write down a small set of strings which collectively suffice to distinguish all states of the DFA, and justify that they do so. If not, construct the state diagram for the minimization of this DFA (you are not required to show the execution of the standard algorithm in detail). [*7 marks*]

7. In this question we consider an LL(1) grammar for a language of expressions. The terminals are the tokens +, ==, `if`, `then`, `else`, `endif`, along with the symbols *Int*, *Real*, *String* which stand for the lexical classes of integers, real numbers and strings respectively. The start symbol is Exp. The productions are as follows:

$$\begin{array}{rcl} \mathsf{Exp} & \rightarrow & \mathsf{Exp1}\ \mathsf{Ops} \\ \mathsf{Ops} & \rightarrow & \varepsilon \mid +\ \mathsf{Exp} \\ \mathsf{Exp1} & \rightarrow & \mathit{Int} \mid \mathit{Real} \mid \mathit{String} \mid \texttt{if}\ \mathsf{Cond}\ \texttt{then}\ \mathsf{Exp}\ \texttt{else}\ \mathsf{Exp}\ \texttt{endif} \\ \mathsf{Cond} & \rightarrow & \mathsf{Exp} == \mathsf{Exp} \end{array}$$

(a) Construct the LL(1) parse table for this grammar. You may find it helpful to compute the relevant First and Follow sets, but you are not required to do so. [*10 marks*]

(b) What problem would arise if the token `endif` were omitted from the above grammar? Give an example of a string that illustrates the problem, and indicate the point at which the problem would manifest itself in the attempt to construct a parse table. [*4 marks*]

(c) We next consider a *type system* which assigns one of the types `Int`, `Real`, `String` to certain expressions. We shall here consider the operator + to be *overloaded* so that it can bear any of the following five type signatures:

```
Int, Int    -> Int
Int, Real   -> Real
Real, Int   -> Real
Real, Real  -> Real
String, String -> String
```

For instance, the second of these allows us to form `3 + 4.5` as a well-typed expression of type `Real`. By contrast, the expression `"a" + 4.5` is not admitted by any of the above signatures. A condition $e{==}e'$ will be well-typed if and only if both e, e' are well-typed and have the same type.

Give a compositional set of typing rules that define which phrases of the language are well-typed, and what their possible types are. Your rules should have the following properties:

- An Exp or Exp1 phrase, if well-typed, should have exactly one of the types `Int`, `Real`, `String`.
- A Cond phrase, if well-typed, should by convention have type `Bool`.

QUESTION CONTINUES ON NEXT PAGE

- An Ops phrase may be assigned a type of the form t`->`t', where $t, t' \in \{$`Int`, `Real`, `String`$\}$. For instance, the operation +5 may be assigned the type `Int->Int`, since for example it may be applied to the integer expression 3 to yield the integer expression 3+5. Note, however, that an Ops phrase may have more than one possible type: for example, the operation +5 may also have the type `Real->Int`. Your typing rules should define the set of *all possible* typings in the case of Ops phrases.

You should give a typing rule for each production $X \rightarrow \alpha$ of the grammar, specifying the possible types for an X-phrase of the form α to be well-typed in terms of the types of the non-terminals (if any) appearing within α. (This is reminiscent of the compositional approach to natural language semantics.) Your rules may be expressed in a mix of English and mathematical notation. The most complex typing rule will be the one for Ops $\rightarrow$ + Exp. [*11 marks*]

8. The following is a parameterized CFG for a small subset of English:

S → NP[sbj] VP | NP[sbj] VP PP
NP[sbj] → D N | N
N → boy | boys | dog | dogs
VP → V | V NP[obj] | Aux V NP[obj]
Aux → doesn't | don't
NP[obj] → D N | N
V → stand | sit | stands | sits | chase | chases
PP → Prep NP[prp]
NP[prp] → D N | N
Prep → in | on
N → table | tables
N → boy | boys
D → the | a | these | this

(a) What is the lowest level in the Chomsky hierarchy at which the language defined by this grammar resides? If the language is regular, explain why and write a regular expression for it. If not, explain why not. [*5 marks*]

(b) The above grammar *over-generates* in the sense that it generates sentences that are not grammatical in English. Correct the grammar by adding agreement markers on the nonterminals (and perhaps copying or splitting rules) so that it only generates the *grammatical* sentences obtained from original grammar. In addition, give two examples of sentences generated by the original grammar that illustrate different ways in which such sentences may be ungrammatical. [*7 marks*]

(c) Give an example of two sentences generated by the original grammar that will have identical probability according to any probability assignment to the rules in the grammar as a probabilistic context-free grammar. Explain your answer. [*6 marks*]

(d) The following corpus of four parsed sentences is given to you:
[S [NP[sbj] [D these] [N boys]] [VP [V stand]]]

[S [NP[sbj] [N dogs]] [VP [V stand]] [PP [Prep on] [NP[prp] [N tables]]]]

[S [NP[sbj] [D the] [N dogs]] [VP [V chase] [NP[obj] [D the] [N boys]]]]

[S [NP[sbj] [D the] [N boy]] [VP [Aux doesn't] [V chase] [NP[obj] [D the] [N dog]]]]
Use this corpus to derive probabilities for the productions of the original grammar using frequency counts. [*7 marks*]