UNIVERSITY OF EDINBURGH

COLLEGE OF SCIENCE AND ENGINEERING

SCHOOL OF INFORMATICS

INFR08008 INFORMATICS 2A: PROCESSING FORMAL AND NATURAL LANGUAGES

Thursday 17th August 2017

09:30 to 11:30

INSTRUCTIONS TO CANDIDATES

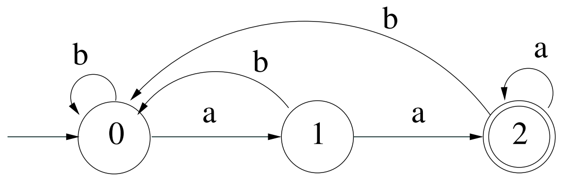
1. **Answer all five questions in Part A, and two out of three questions in Part B. Each question in Part A is worth 10% of the total exam mark; each question in Part B is worth 25%.**
2. **Use a single script book for all questions.**
3. **Calculators may be used in this exam.**

Convener: I. Simpson
External Examiner: I. Gent

THIS EXAMINATION WILL BE MARKED ANONYMOUSLY

Part A

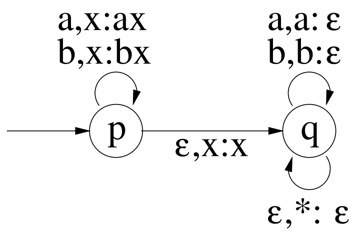
1. Consider the following DFA over the alphabet $\{a, b\}$:



   (a) Write down a set of simultaneous Kleene algebra equations in three variables X_0, X_1, X_2 corresponding respectively to the states $0, 1, 2$ above. [*3 marks*]

   (b) Using the rules of Kleene algebra and Arden's rule, solve the system of equations from part (a) to find a regular expression for the language accepted by the above DFA. Indicate explicitly any points at which you are invoking Arden's rule. [*7 marks*]

2. Consider the following non-deterministic pushdown automaton. The input alphabet is $\{a, b\}$, the stack alphabet is $\{*, a, b\}$, and the initial stack symbol is $*$. Acceptance is by empty stack. We use x as a variable that ranges over the stack alphabet, so that for instance $a, x : ax$ actually stands for the three transitions $a, * : a*$ and $a, a : aa$ and $a, b : ab$.



   (a) Construct a table that displays the course of an accepting computation of this NPDA for the input string *abba*. At each stage, show the transition applied, the input remaining, and the state of the stack. [*6 marks*]

   (b) Describe in words the language accepted by this NPDA. Design a context-free grammar that generates exactly this language. Is your grammar LL(1)? [*4 marks*]

3. Let G be the context-free grammar with non-terminals S, X, Y, start symbol S, terminals a, b, c, d, and productions

$$\begin{aligned} S &\rightarrow XY \\ X &\rightarrow \epsilon \mid abc \\ Y &\rightarrow \epsilon \mid dX \end{aligned}$$

   (a) What does it mean for a grammar to be in *Chomsky normal form* (CNF)? Give one practical reason for wishing to convert a grammar to CNF. [*3 marks*]

   (b) Use the standard procedure to convert the above grammar G into a grammar G' in CNF. [*6 marks*]

   (c) State the precise relationship between the language generated by G and that generated by G'. [*1 mark*]

4. Verbs in Arabic exhibit **root-and-pattern** morphology: the root consists of three consonants, as in the root words *ktb* ("to write") and *brq* ("to shine"). To inflect a verb, these consonants are inserted, in order, into a **pattern** which contains placeholders for them. We denote these placeholders using an uppercase C, as in *taCCuCu* (+Fem +Present) or *CaCaCa* (+Masc +Past). By combining roots and patterns in this way we obtain word forms like the following, which show the consonants of the root in **bold** for clarity:

Word	Analysis	English gloss
ta**kt**u**b**u	ktb +Fem +Present	"she is writing"
ka**t**a**b**a	ktb +Masc +Past	"he wrote"
ta**br**u**q**u	brq +Fem +Present	"she is shining"
ba**r**a**q**a	brq +Masc +Past	"he shone"

We would like you to write a morphological analyser for a fragment of Arabic. Your analyser will receive words as input and produce analyses as output, as in the first two columns of the above table.

   (a) Design a finite-state transducer that receives *taktubu* as input and outputs *ktb +Fem +Present*. Your transducer need not work for any other input/ output pair. [*3 marks*]

   (b) Explain how you would generalize your transducer so that it works for *all* possible sequences of three consonants, inflected by a *fixed* set of patterns paired with their morphological features (e.g. *CaCaCa* paired with *+M +Past*). If the fixed set of patterns is {*taCCuCu*, *CaCaCa*}, then your transducer must work for the examples above, as well as, e.g. *tajbulu* → *jbl +Fem +Present* ("she is creating"), or even the nonsensical *zazaza* → *zzz +Masc +Past*. [*4 marks*]

QUESTION CONTINUES ON NEXT PAGE

QUESTION CONTINUED FROM PREVIOUS PAGE

(c) Now assume that in addition to being given a fixed set of patterns, you are also given a fixed set of roots. Explain how you would further modify your transducer to work for all and only these roots and pattern/ feature pairs. [*3 marks*]

5. Choose a set of constants and predicates suitable for representing the following sentences in first order predicate logic, and translate the sentences into FOPL. For any sentence that is ambiguous, give a formula for all interpretations.

(a) All dogs bark. [*2 marks*]

(b) A dog does not like any cats. [*2 marks*]

(c) A dog does not like a cat. [*2 marks*]

(d) A dog chases a cat. [*2 marks*]

(e) Every dog that chases a cat catches it. [*2 marks*]

Part B

6. Spelling correction systems work by guessing which word a user meant to type, rather than the word they actually typed. An important requirement for spelling correction is to decide which words in the dictionary are most similar to the typed word. For example, if a user types *grammer*, it is more likely that they meant to type *grammar* than *humour*. To make this idea of similarity precise, we can compute the **edit distance** between two strings. Edit distance is the minimum number of character substitutions, insertions, and deletions required to convert an input string into an output string in a single left-to-right pass, without changing the order of the characters. For example, to convert *cats* into *curt*, we scan the input left-to-right and perform the following sequence of operations for an edit distance of three:

Input	Action	Output	Cost
c	keep	c	0
a	substitute *u*	u	1
	insert *r*	r	1
t	keep	t	0
s	delete		1
	Total cost =	3	

Many other sequences of edits can also convert *cats* to *curt*. For example, we could delete all the input characters and insert all of the output characters. However, this would have a cost of 8, so it is not the *minimum* cost sequence of edits.

(a) Design a single weighted finite-state transducer that represents all possible edit operations. Assume that your input and output alphabets consist only of lowercase letters *a* through *z*. You may use abbreviations for large numbers of similar transitions, so long as they are clear. How many states does your transducer require? How many edges? [*5 marks*]

(b) Illustrate the path through your transducer that models the edit from *cats* to *curt* described above. [*3 marks*]

(c) A sequence of edits can also be characterized by an intermediate string representing the sequence of changes from the input to the output. For example, in the cat of *cats* and *curt* we get the string *cSItD*, where S stands for substitute, I for insert, and D for delete. This can be represented graphically:

```
c  a  ϵ  t  s
|  |  |  |  |
c  S  I  t  D
|  |  |  |  |
c  u  r  t  ϵ
```

QUESTION CONTINUES ON NEXT PAGE

This representation suggests an alternate form for the finite-state representation of edit distance. Rather than use a single transducer, we can use two transducers: one to convert the input to the edit sequence, and a second to convert this sequence to the output. Draw these transducers. How many states do they have? How many edges? [*4 marks*]

(d) You may notice that using this representation, it is possible for a character to subsitute for itself! Is this a problem for computing the minimum edit distance? Why or why not? [*3 marks*]

(e) Many sequences of edits can convert *cats* to *curt*. For instance, we could delete all four characters of the input and insert all four characters of the output, giving us a distance of 8. However, we were careful to define the edit distance as the **minimum** number of edits to convert the input string into the output string. To compute this minumum, we can use a variant of the Viterbi algorithm that we learned in class. Using this algorithm, we will compute terms of the form $\textsc{Dist}(i,j)$, which represent the minimum edit distance between the first i characters of the input and the first j characters of the output. Let $\textsc{Cost}(i,j)$ be a function that returns 0 if the ith character of the input string equals the jth character of the output, and 1 if they differ. We can then define edit distance recursively:

$$\begin{aligned}\textsc{Dist}(0,0) &= 0\\ \textsc{Dist}(i,j) &= \min[\textsc{Dist}(i,j-1)+1, \textsc{Dist}(i-1,j)+1, \textsc{Dist}(i-1,j-1)+\textsc{Cost}(i,j)]\end{aligned}$$

The edit distance is then the value of $\textsc{Dist}(i,j)$ when i is the length of the input and j is the length of the output. Using the rule above, create a chart and use it to demonstrate that the minimum edit distance between *cats* to *curt* is indeed three. Since $\textsc{Dist}$ is indexed by two values, your chart should be a matrix, much like a Viterbi HMM chart. (Its interpretation will be slightly different, however.) [*6 marks*]

(f) Using the chart, identify a sequence of edits different from *cSItD* that also results in a distance of 3. [*4 marks*]

7. Consider the following grammar:

$$\begin{aligned} S &\rightarrow \text{NP VB JJ} \\ \text{NP} &\rightarrow \text{DT NN} \\ \text{DT} &\rightarrow \text{the} \\ \text{JJ} &\rightarrow \text{few} \mid \text{prime} \\ \text{NN} &\rightarrow \text{number} \mid \text{prime} \\ \text{VB} &\rightarrow \text{number} \end{aligned}$$

(a) Convert the grammar to Chomsky normal form. [*3 marks*]

(b) When you produced the normal form, you may have changed the first rule of the input grammar. Was the change you made the only possible choice? If it wasn't, what were other possible ways that you could have changed it? [*2 marks*]

(c) Using your CNF grammar, produce a CKY parse chart for the sentence *the prime number few.* Your chart should include all possible items discovered by CKY. [*5 marks*]

(d) Produce an Earley parse of the sentence using the *original* grammar (not the CNF grammar). [*5 marks*]

(e) Compare the set of consituents discovered by CKY (that is, any nonterminal appearing in the chart) with those discovered by Earley (that is, any item that triggers a COMPLETE operation by virtue of having a dot to the end of a production). How does the Earley parse differ from the CKY parse in terms of constituents constructed? Why? [*5 marks*]

(f) Now examine the *spans* associated with the pair of items that are combined when a COMPLETE operation is triggered by the Earley parser. Do these spans relate to your normal form grammar? Or to some normal form grammar? What do you conclude about the relationship of the Earley algorithm to Chomsky normal form? [*5 marks*]

8. In this question, we illustrate the fact that typical *scoping* rules for formal languages (for instance, the requirement that a variable cannot be used until it has been declared) cannot be captured by context-free grammars alone. Note that parts (d)–(f) of this question are quite independent of parts (b) and (c), which are themselves independent of part (a).

To study the phenomenon of scoping in an abstract form, we shall start with a language L over an alphabet $\{a, b, \cdots, z, \mathbf{decl}, \mathbf{use}\}$, where **decl** and **use** are special keywords each considered as a single symbol. The language L is defined by the following context-free grammar, whose start symbol is prog:

$$\begin{aligned} \mathsf{prog} &\rightarrow \epsilon \mid \mathsf{statement}\ \mathsf{prog} \\ \mathsf{statement} &\rightarrow \mathbf{decl}\ \mathsf{var} \mid \mathbf{use}\ \mathsf{var} \\ \mathsf{var} &\rightarrow \mathsf{char} \mid \mathsf{char}\ \mathsf{var} \\ \mathsf{char} &\rightarrow a \mid b \mid \cdots \mid z \end{aligned}$$

We say a program $p \in L$ is *well-scoped* if every statement $\mathbf{use}\ \alpha$ in p is preceded in p by a statement $\mathbf{decl}\ \alpha$ featuring the same variable α. For example,

$$\mathbf{decl}\ x\ \mathbf{decl}\ y\ \mathbf{use}\ x$$

is a well-scoped program, but

$$\mathbf{decl}\ xx\ \mathbf{decl}\ y\ \mathbf{use}\ x$$

is not. We write $K \subseteq L$ for the language consisting of well-scoped programs.

(a) What is the *lowest* level of the Chomsky hierarchy at which the larger language L resides? Justify your answer. [*3 marks*]

(b) Let V denote the set of all variables (i.e. all strings derivable from var by the above grammar). We have seen in lectures that the language

$$W = \{\alpha\alpha \mid \alpha \in V\}$$

is *not* context-free. Use this to show that the language

$$J = \{\mathbf{decl}\ \alpha\ \mathbf{use}\ \alpha \mid \alpha \in V\}$$

is not context-free either. (Hint: How could a context-free grammar for J be transformed into one for W?) [*4 marks*]

(c) Specify some *regular* language R such that $J = K \cap R$. Now apply a standard result from lectures to show that K is not context-free. [*4 marks*]

(d) Briefly explain what is meant by a *linear bounded automaton* (LBA). You need not give a full formal definition, but your explanation should at least make clear the significance of the words 'linear' and 'bounded'. [*7 marks*]

QUESTION CONTINUES ON NEXT PAGE

QUESTION CONTINUED FROM PREVIOUS PAGE

(e) Sketch briefly how our language K could be recognized by an LBA. Again you need not give the formal details, but you should informally describe an algorithm that could be implemented by an LBA. Include specific mention of the points at which your LBA will signal acceptance or rejection of the string. [*6 marks*]

(f) What does the existence of such an LBA imply about the place of K within the Chomsky hierarchy? [*1 mark*]