SCHOOL OF MATHEMATICAL AND COMPUTER SCIENCES

Computer Science

F29LP2

Language Processors (Mock)

Semester 2 201314

Sometime before 5 May 2014

Duration: As long as you like

ANSWER BOTH QUESTIONS (ACTUAL EXAM WILL BE THREE)

Answer each question in a separate script book.

Some words on using this mock paper

There is no concept in this paper that you have not seen already in the lecture notes and exercises. However, I have tried to pitch the difficulty level of this paper slightly above what you will face in the exam. Exam conditions are always harder, because of the stress.

I believe that if you can understand and do these questions, then you are certain to get a decent grade in the exam.

You *must* attempt this entire paper *before* looking at the answers. Have you attempted the paper yet?

ANSWER:

If you see this text, you are looking at the version with model answers. Close this document and try to version without model answers, first.

Good luck.

1. (a) Explain in clear and precise English the precise meaning of the term *formal language*, in the context of this course. (2)

ANSWER:

A set of tokens called an alphabet*, and a set of strings over those tokens which are considered to be in the formal language.*

(b) Consider the following regular expressions:

1. .?
2. .+
3. .*
4. .!
5. .$
6. .
7. .$$
8. .^

In English or otherwise, explain what languages (over ASCII characters) these regular expressions specify. (8)

ANSWER:

1. *The set of ASCII strings of length at most one.*
2. *The set of ASCII strings of length at least one.*
3. *The set of all ASCII strings.*
4. *The set of all ASCII strings of length 2 whose second character is* !.
5. *The set of all ASCII strings of length precisely one.*
6. *The set of all ASCII strings of length precisely one. The lack of* $ *here is a red herring; the wording of the question makes clear that we are matching the* whole *string, not a substring of it, so that all these regexps are in effect terminated by an invisible* $ *anyway.*
7. *Yup. The set of all ASCII strings of length precisely one.*
8. *The empty set. This is because* ^ *matches the start of the string, and* . *matches an initial character before* ^*, so this particular regular expression is in effect a paradox which no string can satisfy.*

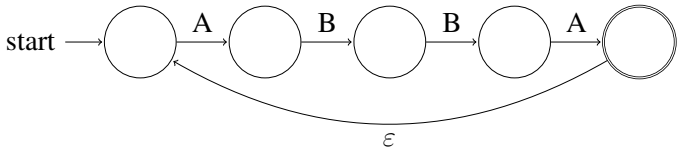
(c) 1. Explain in English what a non-deterministic finite automaton (NFA) with ε-moves is.
2. Explain intuitively how an NFA with ε moves can be considered to specify a language.
3. Explain the connection with regular expressions.

(3)

ANSWER:

1. *An NFA with ε-moves is a rooted graph (the root is the 'initial state') with edges either labelled by elements of a set of tokens Σ (called an 'alphabet'), or without a label (this is sometimes indicated by annotating them with ε. Some of the nodes are designated as 'final states'.*
2. *If we trace possible paths from the initial state to some final state (possibly passing through other final states without stopping) and record the tokens labelling edges, then we get a set of strings. These strings are a language over Σ.*
3. *NFA with ε moves determine precisely the same class of languages as do regular expressions.*

(d) Express as a regular expression the language accepted by the following automaton:



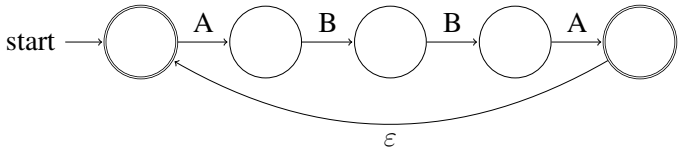
(2)

ANSWER:

(ABBA)+

(intuitively: "ABBA forever")

(e) Explain precisely, in English or otherwise, what the difference is between the previous regular expression and the one determined by this automaton:



(1)

ANSWER:

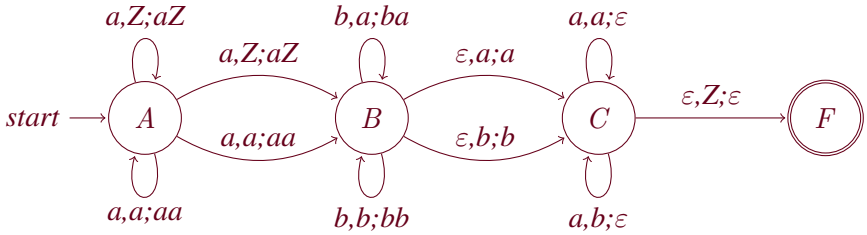
(ABBA)*

Though we could listen to ABBA all day should the opportunity arise, we do not necessarily *need to listen to ABBA.*

(f) Draw a PDA that recognises the language $\{a^i b^j a^{i+j} \mid i \geq 1, j \geq 0\}$. Your answer must clearly state the acceptance mode used. (4)

ANSWER:

Acceptance by empty stack or by final state; either works here:



2. (a) Give one example each of

- a left-recursive grammar, (1)
- a right-recursive grammar, (1)
- a grammar that is both left- and right-recursive. (1)

ANSWER:

Left-recursive grammar: $S ::= S0$ *(start symbol is* S*). Right-recursive grammar:* $S ::= 0S$*. Grammar that is both:* $S ::= S0S$ *or* $S ::= SS$ *or even just* $S ::= S$*.*

(b) Write a context-free grammar for the English language with nonterminals $\langle sentence \rangle$, $\langle noun \rangle$, $\langle verb \rangle$, $\langle definite\text{-}article \rangle$ (words like 'the' or 'that'), and $\langle adverb \rangle$ ('quickly', 'happily'). Your grammar should be sufficiently developed to produce the following sentences:

- The cat scratched the mat.
- Linux rocks.
- Jamie happily writes questions.

We do not care if your grammar also produces a incorrect sentences, such as "The the cat scratched the mat". You may ignore case. (6)

ANSWER:

- $\langle sentence \rangle ::= \langle noun \rangle \langle verb \rangle \langle noun \rangle \mid \langle noun \rangle \langle verb \rangle$
- $\langle noun \rangle ::= cat \mid mat \mid Linux \mid Jamie \mid questions \mid \langle definite\text{-}article \rangle \langle noun \rangle$
- $\langle definite\text{-}article \rangle ::= the$
- $\langle verb \rangle ::= scratched \mid is \mid writes \mid rocks \mid \langle adverb \rangle \langle verb \rangle$
- $\langle adverb \rangle ::= happily$

I would not set this question in an exam because it is too ambiguous; a student might ignore the spirit of the question and simply write the grammar

$\langle sentence \rangle ::= The\ cat\ scratched\ the\ mat \mid Linux\ rocks \mid \ldots.$

However, for a mock this is OK.

By the way, in practice we need to distinguish between nouns *and* noun phrases*, and* verbs *and* verb phrases*. These are used to exclude the pathalogical (i.e. grammatically incorrect) sentences which the simple-minded grammar above can produce.*

(c) Consider the following grammars:

$$T ::= T0 \mid T1 \mid \varepsilon$$
$$S ::= 0S \mid 1S \mid \varepsilon$$
$$U ::= UU \mid 0 \mid 1 \mid \varepsilon$$

- All three grammars generate the same language. What is it? (1)

  ANSWER:

  Possibly empty binary strings (strings of 0s or 1s).

- Rank the grammars in order from best to worst from an implementational point of view, and explain your ranking. (2)

  ANSWER:

  S, T, then U. S is right-recursive and well-suited to implementation since it 'eats' characters from the left, so a naive implementation that applies the first available rule to the input stream will just work recognise the relevant language. T is left-recursive, and though there is nothing wrong with this mathematically, if we are foolish enough to give a naive left-to-right implementation, then it will immediately loop. U is clearly deprecated, since not only is it likely to loop, but it may loop exponentially (due to the rewrite $U ::= UU$).

(d) Take a *natural number* to be an element of the language determined by the regex $0 \mid [1\text{-}9][0\text{-}9]^*$, and a *decimal number* to be an element of the language determined by the regex $(0 \mid [1\text{-}9][0\text{-}9]^*)\backslash.[0\text{-}9]^+$ (so 00 is not a number but 10 is, and $1.$ is not a decimal number but 0.00 and 0.01 are decimal numbers).

Write a grammar (which need not be context-free) that will generate sentences over tokens $\{0, \ldots, 9, ., \approx\}$ of the form "$D \approx N$", where D is a decimal number and N is a natural number and N is equal to D rounded down to the nearest whole number.

So for instance, your grammar should recognise $10.9 \approx 10$ and $0.49 \approx 0$.

You may use dots notation to indicate evident repetition of a succession of rules, as in "$S ::= 0 \mid \cdots \mid 9$". Answers that are not evidently correct may score zero marks; if in doubt, provide a clear English explanation of how your answer works. Clearly state the start symbol. (4)

ANSWER:

$$
\begin{aligned}
N &::= S \mid 1S' \mid \cdots \mid 9S' \\
S &::= 0 \mid 1 \mid \cdots \mid 9 \\
S' &::= \varepsilon \mid 0S' \mid \cdots \mid 9S' \\
E &::= N.SS' \approx N
\end{aligned}
$$

The start symbol is E.

(e) Write a grammar to recognise sentences over $\{1, 2\}$ such that the sum of the 1s is equal to the sum of the 2s (in other words: there are twice as many 1s as 2s). Clearly state the start symbol. (2)

ANSWER:

$$S ::= \varepsilon \mid 1S1S2S \mid 1S2S1S \mid 2S1S1S$$

Start symbol is S.

(f) Can your grammar be left-factored and so made deterministic to eliminate potential backtracking? Explain. (2)

ANSWER:

I don't think so. Certainly we can branch on whether the first input symbol (by convention the leftmost symbol) is 1 or 2, but there is no way by reading a fixed number of symbols from the start *of the stream, that we can decide whether the corresponding 1 occurs before or after the single matching 2—there could be a million 1s and only then five hundred thousand 2s.*

END OF PAPER