1 Finite State Automata and Regular Language

Consider the non-deterministic finite automaton $A$:

\[
\begin{array}{c}
S_0 \quad b \\
\downarrow \quad \uparrow \\
\{a, b\} \\
\downarrow \quad \uparrow \\
S_1 \quad b \\
\uparrow \quad \downarrow \\
S_2 \quad b
\end{array}
\]

$A$ is intended to recognise the language $L$ of strings over $\{a, b\}^*$ where every occurrence of $a$ is preceded by and followed by an occurrence of $b$.

(i) Use the ‘subset construction’ algorithm given in lectures to produce an deterministic finite automaton $B$ which recognises the same language as $A$.

Be clear on which states of $B$ represent which subsets of states of $A$. [2 marks]
(ii) Are any of the states of the DFA $B$ equivalent to each other? Answer this question using the algorithm given in lectures.

Be careful to explain how you know that the algorithm has terminated, so that you have discovered all possible groups of equivalent states.

[2 marks]

(iii) If you discovered any groups of states that are equivalent, use this and the procedure outlined in lectures to give a DFA $C$ that recognises the same language as $B$ but is minimal.

[1 mark]

(iv) Prove that the DFA $C$ recognises the intended language $L$. Ensure that you clearly state your two main proof obligations and prove them separately, and give your proofs in full rigorous detail.

[5 marks]

Hint 1: both proofs will be easier if you use the proof by contrapositive technique, where instead of proving the subgoal $P \Rightarrow Q$, you prove the equivalent $\neg Q \Rightarrow \neg P$.

Hint 2: Think about what all the different cases are in which a string would not be in $L$.

(v) Using the procedure given in lectures, convert the original NFA $A$ into a right-linear grammar.

[1 mark]

(vi) Let $K$ be the language of strings over $\{a, b\}^*$ where every occurrence of $a$ is preceded by or followed by (or both) an occurrence of $b$. Design a NFA that accepts $K$. You do not need to prove that it is correct.

[2 marks]

2 Pushdown Automata and Context Free Language

2.1 Grammars

(i) Find a right-linear grammar which accepts the same language as this NFA. Your grammar may (unlike the definition in lectures) contain productions of the form $A \rightarrow B$, thus it may contain the following sorts of productions:

$A \rightarrow a$ or $A \rightarrow aB$ or $A \rightarrow \epsilon$ or $A \rightarrow B$
It is understood that upper case letters are non-terminals, and lower-case letters are terminals.

NOTE: the lecture slides have been amended so that in the examples given the NFAs now contain $\epsilon$-edges.

(ii) Describe how to convert a grammar containing productions of the form $A \rightarrow B$ into a grammar which contains no productions of this form, but accepts the same language.

If your answer to part (a) contains productions of this form, convert your grammar to a right-linear grammar as defined in lectures (ie, not containing productions of the form $A \rightarrow B$), which accepts the same language.

(iii) Find a left-linear grammar which accepts the same language as this NFA. Your grammar may (unlike the definition in lectures) contain productions of the form $A \rightarrow B$, thus it may contain the following sorts of productions:

\[ A \rightarrow a \text{ or } A \rightarrow Ba \text{ or } A \rightarrow \epsilon \text{ or } A \rightarrow B \]

[2 marks]

2.2 A Content Comparison Problem

Design a deterministic PDA that accepts the language over \{a, b\} in which each string has more a’s than b’s. The input strings string will be terminated by a hash symbol (#). (Thus the input alphabet is \{a, b, #\}.)

For this PDA, strings will be deemed to be accepted when the PDA consumes the input and empties the stack, including Z (the symbol initially found on the stack).

[3 marks]

2.3 Palindrome Acceptance

Design an NPDA (Nondeterministic PDA) that recognises nonempty palindromes over the alphabet \{a, b\} and design it to guess whether or not the string is of odd or even length (ie, whether or not there is a middle symbol which is not repeated). (Palindromes are strings that read the same forwards or backwards). Included among the strings that it is required to accept are ababbaba and ababa.

Strings should be deemed to be accepted if the input is consumed and the stack is empty.

Confirm that ababbaba is accepted by giving a trace.

[4 marks]
3 Turing Machine and Computability

3.1 Palindrome Acceptance

Construct a Turing Machine that recognizes palindromes over the alphabet \{a, b\}.

In the initial state the tape of the TM will have a string made up of a’s and b’s with no embedded spaces and with the read head somewhere over the string.

Your machine should halt if (and only if) the string is a palindrome. In the case that the TM does halt, the tape should be left with just the symbol T on it, and the head positioned above it.

Your answer should consist of two parts:

- A description in plain English that simply describes how your machines works.
- A diagram for your Turing Machine, in the notation used in lectures.

[4 marks]

3.2 Computability

Assume that the Halting Problem has no solution. Is it undecidable whether an arbitrary TM, X, ever prints a given letter when provided input, w? Use, for contradiction, the fact that were this problem decidable it would give a solution to the Halting Problem.

[5 marks]