CITY UNIVERSITY LONDON

IN3017: Theory of Computation
Coursework

Introduction

This is the coursework for Theory of Computation. On completing this coursework, you should be able to:

- Read and write finite state automata (both deterministic and non-deterministic).
- Understand regular languages and regular expressions.
- Convert regular expressions to and from finite state automata.
- Recognise languages and apply arguments to show that a language is not regular, including use of the pumping lemma.
- Read and write context-free grammars;
- Read and write pushdown automata;
- Apply the pumping lemma for context-free languages;
- Read and write Turing Machines (and equivalent notions of computation).

The subject matter follows lectures 1-6 (and associated class exercises).

Information

Module marks and rubric

This coursework is worth 30% of the Theory of Computation module mark.
You should answer two questions, one from Part A and one from Part B.
Each question is worth 50 marks.

Grading

Solutions will be graded according to their correctness, including accuracy of the representation and syntactic correctness.

Submission deadline

This coursework should be handed in before 5pm on Sunday 1st December 2019 (Week 9). In line with School policy, late submissions will awarded no marks.

Submissions

You should submit your solutions via the coursework submission area in Moodle. Including images of handwritten work as part of your submission is acceptable, but it is your responsibility to ensure that work contained in such images in legible.

Feedback & Return

Marks and general feedback will be available as soon as possible, certainly on or before Friday 20th December 2019. Your individually marked submissions will be available as soon after this date as possible.

Plagiarism

This is an individual assignment. If you copy the work of others (either that of fellow students or of a third party), with or without their permission, you will score no marks and further disciplinary action will be taken against you.

PART A

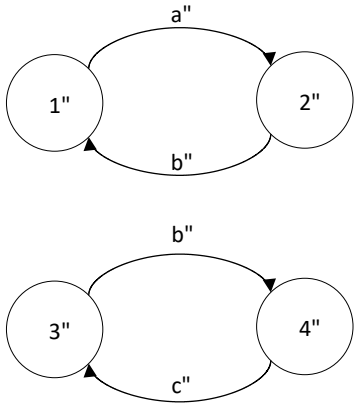
Question A1

Generate six integers *x*, *y*, *z*, *u*, *v* and *w* from your login. Call the first digit of your login *i*, the second *j* and the third *k*. The numbers are defined as follows:

1. $x = \lfloor i/2 \rfloor$. That is, divide *i* by 2 and take the floor (round down)
2. $y = (x+3) \bmod 5$
3. $z = \lfloor i/5 \rfloor + 1$
4. $u = \lfloor j/5 \rfloor + 3$
5. $v = \lfloor k/5 \rfloor + 1$
6. $w = 5 - v$

For example, if your login was abcd036, then $i = 0$, $j = 3$, $k = 6$ and you would find that $x=0$, $y=3$, $z=1$, $u=3$, $v=2$ and $w=3$.

a) Consider the incomplete (indeed disconnected) NFSA, with $\Sigma = \{a,b,c\}$, represented as a transition diagram below:



Complete this NFSA (to give NFSA V) by adding a start state, indicating an accept state and adding five transitions, as follows (where the values of *x*, *y*, *z*, *u*, *v*, *w* are as calculated above.

Recall that ((s,α),t) is a transition from state s to state t, labelled with α):

1. 0 is the start state to be added
2. *x* is the accept state
3. ((*y*,c),*x*)
4. ((3,a),*z*)
5. ((2,ε),*u*)
6. ((0,ε),*v*)
7. ((0,ε),*w*)

Represent V as a transition diagram.

[5 marks]

b) Transform the NFSA V into a DFSA that accepts the same language. It is not necessary to draw its transition diagram; a transition table including an indication of start and accept states is enough.

[20 marks]

c) Apply the algorithm contained in the proof of Kleene's Theorem to construct a regular expression that generates the language recognised by the NFSA V. The solution should clearly show each step of the process.

[25 marks]

Question A2

a) Suppose Σ is the last three letters in your login (if two or more letters are the same, please pick the next letter(s) in the Latin alphabet as your second/third symbol) and call those letters α, β, γ in the order in which they occur. Let *i, j, k* be the first, second and third digits of your login. That is, if your login is abcd123, Σ = {b,c,d}, α=b, β=c, γ=d, $i = 1$, $j = 2$, $k = 3$. Are the following languages regular? If so, give a regular expression for the language. If not, demonstrate this using the Pumping Lemma for regular languages.

i) The language of all strings over Σ defined as follows:

$$\{\beta^i\alpha^n\gamma\alpha^{n+j} \mid n \geq 0\}$$

For example, if Σ = {b,c,d} and $i = 1$, $j = 2$, cbbbdbbbbb is in the language, but cbbbdbbb, ccbbdbbbb, cbbddbbbb are not.

ii) The language of all strings over Σ whose first three characters form a palindrome over {β, γ} (where β, γ are defined as above), followed by *k* occurrences of α.

[25 marks]

b) Suppose, $L_1 = \{a^n b^n \mid n \geq 0\}$. Give another language L_2 such that $L_1 \cup L_2$ is regular.

[5 marks]

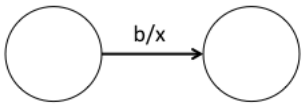
c) Let L be a regular language over Σ={a, b}. Demonstrate that the following is also a regular language [Hint: closure]:

M ={w | w$\in$L and w ends with ba}

[5 marks]

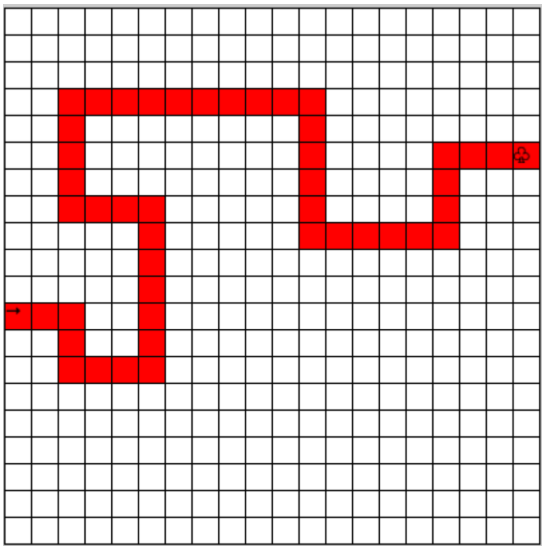
(Question A2 continues overleaf)

d) Recall that a *finite state transducer* (see Sipser, page 87) is like a deterministic FSA, except that in addition to consuming input, each transition can produce an output. Notationally, a transition looks like this:



where b is the consumed input and x is the output.

- An ant lives in a grid world, an *n* x *m* two-dimensional grid of cells. Within this grid is a pheromone trail leading to the tree whose leaves are the ant's food, each point of the trail consists of one cell. The trail is continuous, with one spot being either to the north, south, east of west of the next spot; no diagonals are allowed.
- The ant can detect what is straight ahead of it: either nothing or the grid boundary (E), the trail (P) or the tree (T).
- If the ant detects the tree it moves forward one cell (F), then will remain in the same place (S) forever.
- If the ant detects a trail cell, it moves forward one cell (F).
- If the ant detects empty it moves 90 degrees to the left (L) and again detects what is ahead of it. If it again detects empty it will move 90 degrees to the right (R) *twice* and again detect what is ahead of it.
- If the ant has detected empty ahead, to the left and to the right it remains in the same place forever.
- You may assume that at any point the ant will be able to detect at most one cell in the pheromone trail (there are no choices available).
- Using these rules, the ant, initially at the west side of the grid facing east, can start at the beginning of a pheromone trail and follow it to the tree. For example (where the coloured cells are the trail):



Give a finite state transducer to control the ant as above. The transducer should use what is detected in the cell ahead as input {E, P, T} and produce motion commands as output {L, R, F, S}. You may omit any transitions where the corresponding input is not possible.

[15 marks]

PART B

Question B1

Consider the following context-free grammar for a language S with alphabet {a,b}:

$S \rightarrow BDE \mid aE \mid b$
$B \rightarrow BC \mid \varepsilon$
$C \rightarrow b$
$D \rightarrow SC \mid \varepsilon$
$E \rightarrow BD \mid b$

a) Convert this grammar to Chomsky Normal Form (CNF).

[15marks]

b) Using your CNF grammar, construct a non-deterministic push-down automaton (NPDA) that recognises S.

[15 marks]

c) Show that the string 'bbabb' is in S by parsing your CNF grammar.

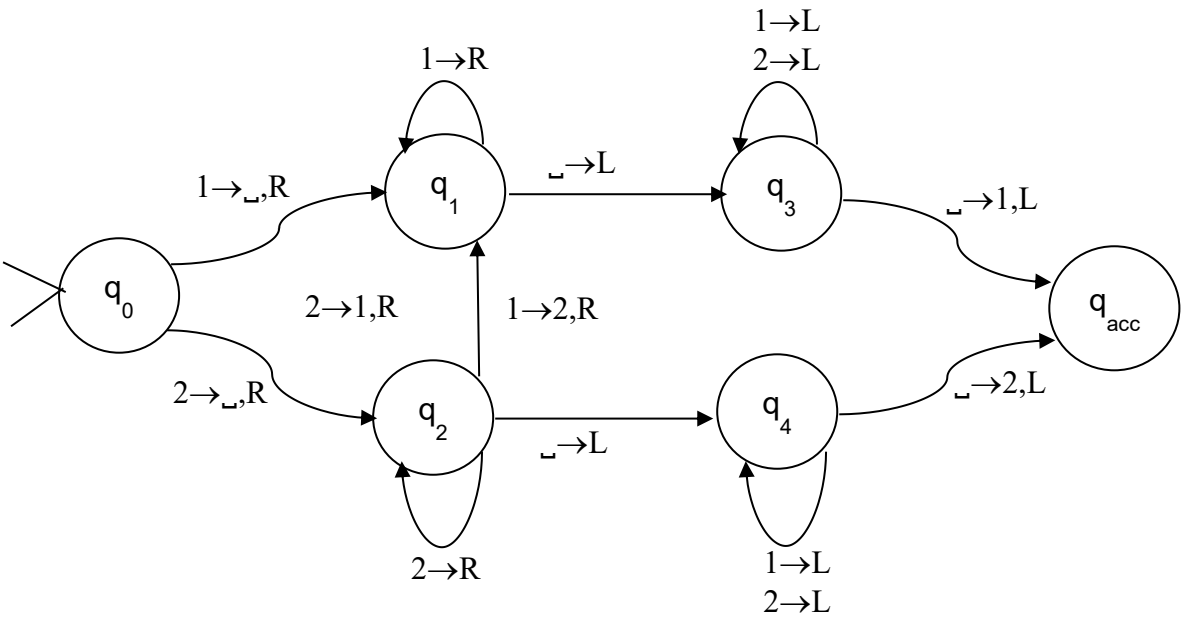
[10 marks]

d) Using your parse tree from part c), trace the operation of your NPDA on the string 'bbabb'.

[10 marks]

Question B2

Consider the Turing Machine represented diagrammatically below (where q_{reject} is not reachable and a non-null input is assumed).



a) Trace the computation with this machine on the input 12221

[5 marks]

b) Let Σ be the set of all lower case Latin letters, augmented with '*'. That is, Σ = {a, b, c, … , y, z, *}. Input will be made of these characters, although other characters might be used during computation. Let string *s* be the string consisting of the last two letters of your login.

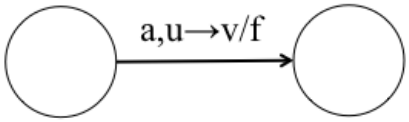
Where the input to the machine is a sequence of letters ending with a '*', write a Turing Machine that will write the number of occurrences of *s* as a substring of the input as a little-endian binary number in the cells immediately beyond the *.

So, for example, if your login was abcd123, your Turing Machine will count the number of occurrences of the string "cd" as a substring of the input and write that number as a little-endian binary number beyond the *. That is, for input acdzcdjcdcd* the machine will end its computation with the last four non-blank symbols on the tape being *001, whilst for input hycdcdiqcdg* the machine will end its computation with the last three non-blank symbols on the tape being *11. The symbols before the * may or may not be unchanged on halting (depending on your solution).
[A little-endian binary number has its least significant bit first, and its most significant last, so 4 is 001, 11 = 8 + 2 + 1 is 1101 and 14 = 8 + 4 + 2 is 0111.]

[20 marks]

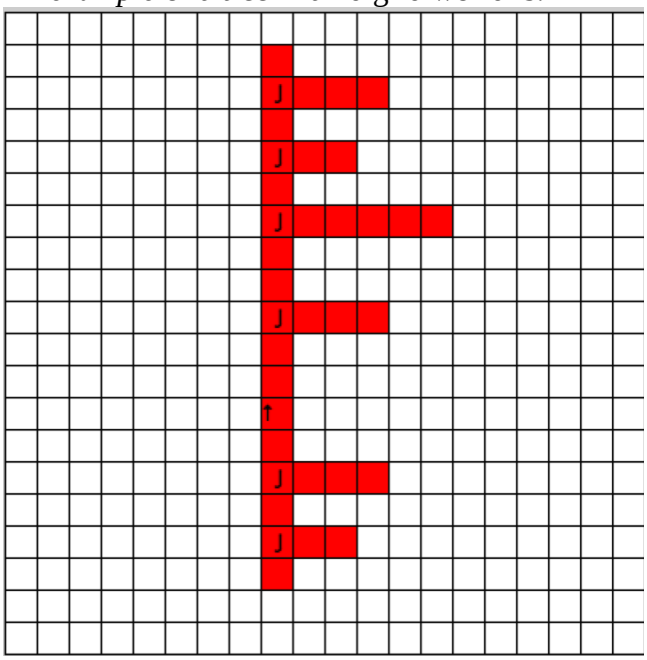
(Question B2 continues overleaf)

c) A *pushdown transducer* is the same as deterministic PDA, except that in addition to consuming input and pushing/popping a stack, each transition can produce an output. Notationally, a transition looks like this:



where a is the consumed input, u is popped, v is pushed and f is the output.

An ant is somewhere on the trunk of a tree in a 2d grid world. The tree consists of a vertical trunk and horizontal branches to the right. There are special junction cells where the trunk and a branch meet.
The ant can detect as input what is in front of it: a junction cell (J), part of the tree (T) or empty (E); if looking beyond the grid boundary the ant detects E.
Two junctions are not allowed to be next to each other, and the top of the tree is a T cell.
The ant can move one cell forward (F), 90 degrees to the right (R), 90 degrees to the left (L) or stay still (S).
An *example* of a tree in a 2d grid world is:



There are many possible trees and the number of branches, their length and how high the tree is might vary from example to example.

The ant starts somewhere on the trunk, facing up the trunk. This start point is somewhere is the middle of the tree, there may well be other branches below.

Write a ***deterministic*** *pushdown transducer* that will control the ant in such a way that it will explore the rest of the tree *above* it, before returning to the junction immediately above it and staying still. By exploring it is meant that the ant will visit every cell (including along the branches) above it in the tree. You may omit any transitions where the corresponding input is not possible.

[10 marks]

(Question B2 continues overleaf)

d) An *Unlimited Register Machine (URM)* is defined as follows (Cutland 1980):
A URM consists of a countable number of **registers**, R_1, R_2, R_3, ... and a **program**.
Each register contains an integer.
A program is made up of a numbered sequence of instructions. There are four kinds of instruction.
Z(*i*), which sets register R_i to 0.
S(*i*), which increases the value held in register R_i by 1.
J(*i*,*j*,*k*), which compare the values held in registers R_i and R_j. If they are the same, then jump to instruction *k* otherwise move to the next instruction.
T(*i*,*j*), which makes the assignment $R_j = R_i$.

A **computation** initialises a finite number of registers to non-zero values and executes the first instruction. If that instruction is a J (jump) instruction, the next instruction to be executed is the one given by the jump, for other instructions, the following instruction is the next executed. Computation continues until an undefined instruction is reached (that is, the end of the program is reached).
For example: consider the program
1. J(1,2,5)
2. S(2)
3. S(3)
4. J(1,1,1)

With initial configuration

5	2	0	0

Computation proceeds as follows:
1. No jump
2. $R_2 = 3$
3. $R_3 = 1$
4. Jump to 1
1. No jump
2. $R_2 = 4$
3. $R_3 = 2$
4. Jump to 1
1. No jump
2. $R_2 = 5$
3. $R_3 = 3$
4. Jump to 1
5. End.

Hence the final configuration is:

5	5	3	0

(The program calculated an encoding of 5-2=3)

Define a URM whose initial configuration is:

i	*j*	*k*	0

where *i*,*j*,*k* are three integers, which calculates $2*i+j+k+1$ and place this value in R_4.
[6 marks]

Illustrate the execution of your URM when *i*,*j*,*k* are the three digits from your login.
[4 marks]

Reference: N. J. Cutland (1980). *Computability*: *An introduction to recursive function theory*. Cambridge University Press.