Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Declaration concerning plagiarism

I have read and I understand the plagiarism provisions in the General Regulations of the *University Calendar* for the current year, found at http://www.tcd.ie/calendar

I have completed the Online Tutorial in avoiding plagiarism 'Ready, Steady, Write', located at http://tcd-ie.libguides.com/plagiarism/ready-steady-write

STUDENT NUMBER: 23373999

SIGNED: Luke Bailey

DATE: 26-01-2025

MAU22C00 Assignment 2

1)(a) Alphabet = {a, b, c}
Start symbol = <S>
Production Rules

<S> -> a<S>a
<S> -> b<A>
<A> -> b<A>b
<A> -> c<A>c
<A> -> c

Since the Start Symbol is <S> either is valid

1. Starting with <S> -> a<S>a

Repeatedly applying this rule creates a core of <S> surrounded by 'a' n times (with n being the number of times the rule is applied).

Hence, strings produced in this manner have the structure $a^n <S> a^n$

2. <S> -> b<A>

This rule is the only one allowing a conversion from the non-terminal <S> to <A>, and a 'b' is prepended to the string, hence after considering this rule with the previous we have a string in the form $b\, a^n <S> a^n$

3. Rules $\langle A \rangle \to b \langle A \rangle b$

$\langle A \rangle \to c \langle A \rangle c$

We must consider these rules togher because hey may be interchanged at any moment. Hence $\langle A \rangle$ may be surrounded by m pairs of either b or c depending on the number of iterations. to describe the string ~~x~~ $x \in b, c$

After considering this, the string must be in the form

$$a^n b \, x^m \langle A \rangle x^m a^n$$

Where

$x \in b, c$

m is iteration count 2

n is iteration count 1

4. $\langle A \rangle \to c$

This final rule terminates the string. Hence strings must be of the form

$$a^n b \, x^m c \, x^m a^n$$

Or in other words, 'c' surrounded by m matching pairs of 'b' or 'c', prepended by 'b' surrounded by n pairs of 'a'

MAU22C00 Assignment 2

(b)

The pumping lemma states:

If L is a regular language, then there is a number p where if w is any word in L of length at least p, then $w = xuy$ for words x, y, u satisfying

$u \neq \epsilon$

$\|xu\| \leq p$

$xu^n y \in L \; \forall n \geq 0$

$x = a^n$

$u = b x^m c x^m$

$v = a^n$

For simplicity $m = 0$

$x = a^n$

$u = bc$

$v = a^n$

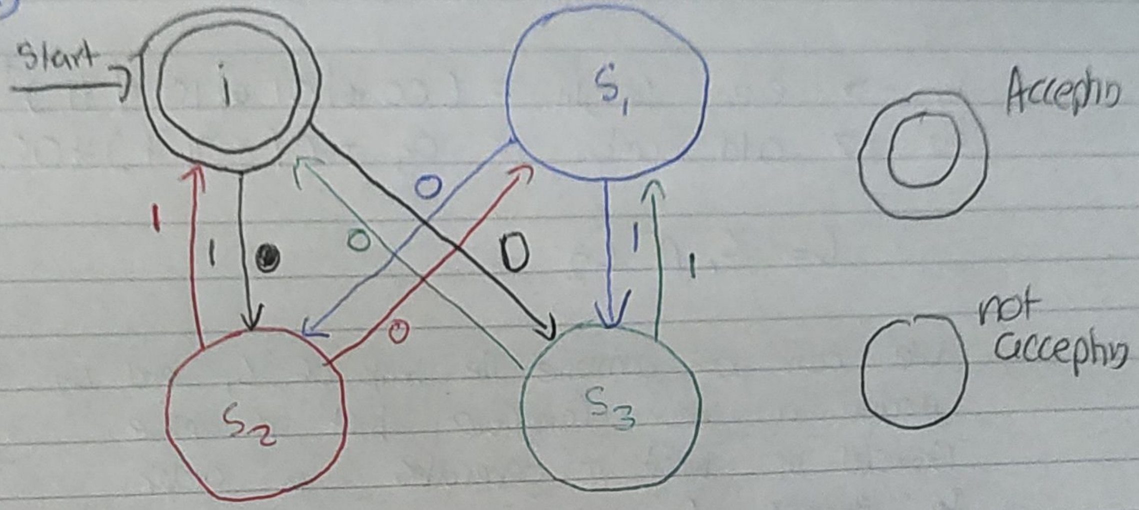
Pumping this would create strings of the form

$a^n \; bcbc \; a^n$

This is not valid as there is no central 'c' such that it is ~~...~~ prepended by an unmatched 'b' as per the language rules in (a). Hence this is not a regular language.

B

Q2(a)



	Length	# of 0	Accept
i	even	odd	✓
S_1	even	even	✗
S_2	odd	odd	✗
S_3	odd	even	✗

(b) $A = \{0, 1\}$

Start Symbol = i

Non terminals = i, S_1, S_2, S_3

Production rules:

$i \to 0S_3$

$i \to 1S_2$

$i \to \epsilon$

$S_1 \to 1S_3$

$S_1 \to 0S_2$

$S_2 \to 1i$

$S_2 \to 0S_1$

$S_3 \to 1S_1$

$S_3 \to 0i$

This cannot produce any other strings as it satisfies the above state acceptor diagram perfectly

(c) $L \to$ even length and odd number of 0s

$L_1 \to$ even length $= (00 | 01 | 10 | 11)^*$
$L_2 \to$ odd number of 0s $= (1^*01^*)^*0(1^*)$

$L = L_1 \cap L_2$

We can just combine the traits of L_1 and L_2 to produce a new language that is more restricted in what it generates in order to express L

(d) $w = 010101$

$x = ?$
$u = ?$
$y = ?$

$w = xuy$

$|xu| \leq 4$

$u \neq \epsilon$

$xu^ny \in L \; \forall n \geq 0$

$x = 0$
$u = 1010$
$y = 1$

$|0\,1010| \leq 4$
$5 \leq 4$

$1010 \neq \epsilon$

$xu^n y \in L \quad \forall n \geq 0$

$n = 0$	01	as u has 2 0s the oddness stays the same with pumpin
$n = 1$	0 1010 1	
$n = 2$	0 1010 1010 1	
$n = \ldots$	...	

as u has 4 letters the length stays even when pumped

Q3(a) A = finite alphabet
L = regular language

Since L is regular there must be a finite state automaton that expresses it

$$M = (S, A, t, i, f)$$

Therefore there must also be one for L^* if it is also regular.

We can do this by wrapping M in a new automaton M'

$$M' = (\{i, S\}, A, t \circ \{t(i, S)\}, f \circ \{t(\epsilon, i)\})$$

This allows us to pass words to M to be recognised, and if they are, loop again to produce the repetition of the Kleene star

Hence L^* is also regular

(b) L = regular language over A
A = alphabet

$A^* \setminus L$ = compliment of L in A^*

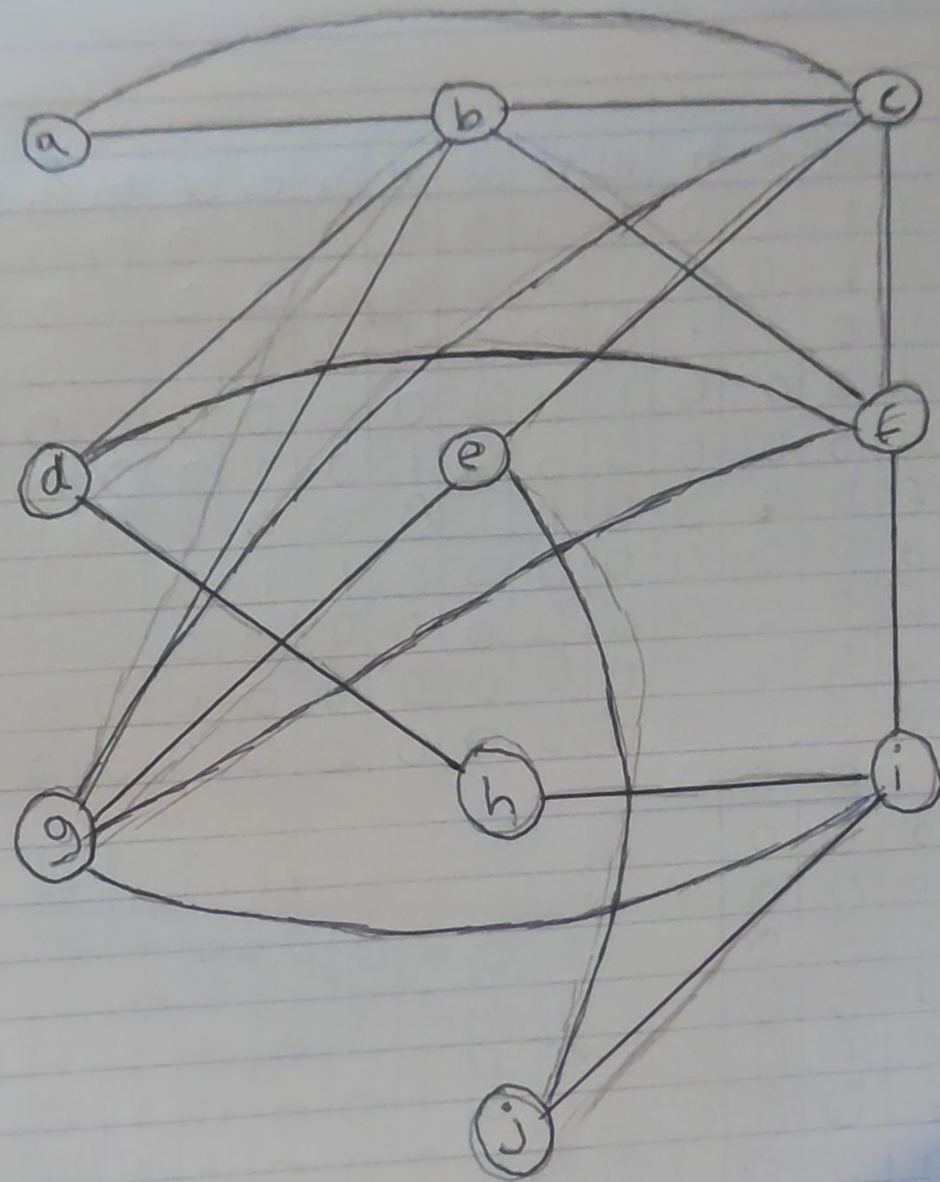
Same as (a), finite state automaton exists

We can keep this almost the same, but instead change the accepting states to be the compliment of what they were before to produce $A^* \setminus L$

$M' = (\{i, s\}, A, t \cup \{t(i, s)\},$
$S \setminus f \cup \{t'(\epsilon, i\})$

7

Q4 (a)



8

10

(b) Incidence table

	ab	ac	bc	bd	bf	df	ce	cg	eg	cf	bg	fg	fi	gi	dh	hi
a	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
b	1	0	1	1	1	0	0	0	0	0	1	0	0	0	0	0
c	0	1	1	0	0	0	1	1	0	1	0	0	0	0	0	0
d	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0
e	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
f	0	0	0	0	1	1	0	0	0	1	0	1	1	0	0	0
g	0	0	0	0	0	0	0	1	1	0	1	1	0	1	0	0
h	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
i	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1
j	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	ej	ij
a	0	0
b	0	0
c	0	0
d	0	0
e	1	0
f	0	0
g	0	0
h	0	0
i	0	1
j	1	1

Incidence Matrix

$$
\begin{pmatrix}
1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&6\\
1&0&1&1&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&1&1&0&0&0&1&1&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&1&0&1&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&1&1&0&0&0&1&0&1&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&1&0&1&1&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&1&0&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1
\end{pmatrix}
$$

(c) Adjacency Table

	a	b	c	d	e	f	g	h	i	j
a	0	1	1	0	0	0	0	0	0	0
b	1	0	1	1	0	1	1	0	0	0
c	1	1	0	0	1	1	1	0	0	0
d	0	1	0	0	0	1	0	1	0	0
e	0	0	1	0	0	0	1	0	0	1
f	0	1	1	1	0	0	1	0	1	0
g	0	1	1	0	1	1	0	0	1	0
h	0	0	0	1	0	0	0	0	1	0
i	0	0	0	0	0	1	1	1	0	1
j	0	0	0	0	1	0	0	0	1	0

Adjacency Matrix

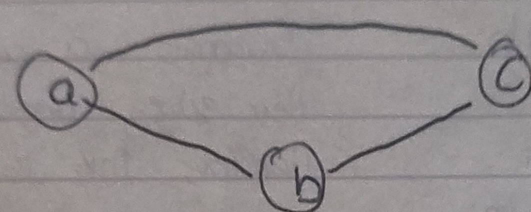
$$\begin{pmatrix}
0&1&1&0&0&0&0&0&0&0\\
1&0&1&1&0&1&1&0&0&0\\
1&1&0&0&1&1&1&0&0&0\\
0&1&0&0&0&1&0&1&0&0\\
0&0&1&0&0&0&1&0&0&1\\
0&1&1&1&0&0&1&0&1&0\\
0&1&1&0&1&1&0&0&1&0\\
0&0&0&1&0&0&0&0&1&0\\
0&0&0&0&0&1&1&1&0&1\\
0&0&0&0&1&0&0&0&1&0
\end{pmatrix}$$

(d) This is not a complete graph as every vertex does not have the maximum number of edges connecting to it. For example, 'a' has 2 incident edges, but in a complete graph it would have 9 (# nodes - 1) incident edges

(e) This is not a bipartite graph as it cannot be split into two separate graphs that are isolated from each other

(f) This is not a regular graph as for it to be regular each vertex must have the same number of incident edges, however, for example 'a' has 2 incident edges and 'c' has 4, making this graph not regular.

(g) This graph has a regular subgraph



This graph is 2 regular as each vertex has degree 2

(h) Degrees:

a: 2 ←

b: 5

c: 5

d: 3

e: 3

f: ~~4~~ 5

g: 5

h: 2 ←

i: 4

j: 2

The lowest degree present is 2, therefore if we remove a's edges and h's edges we can produce 3 components, since individual vertices count as components. Hence, the minimum number of edges we can remove is 4.

12

(i) {f, d, h, i} = vertices

~~{f, d, ... i~~ ~~Swapping d and i~~

(i)

$\varphi(a) = j$
$\varphi(ac) = jc$
$\varphi(ab) = jb$
$\varphi(j) = a$
$\varphi(ji) = ai$
$\varphi(je) = ae$

$\varphi(x) = x$

Swapping a and j then their edges preserves adjacency, therefore this is an isomorphism

(ii)

$\varphi(a) = h$
$\varphi(ac) = hc$
$\varphi(ab) = hb$
$\varphi(h) = a$
$\varphi(hd) = ad$
$\varphi(hi) = ai$

This also satisfies the conditions for a graph isomorphism, hence φ is not unique