COMP2270/6270 – Theory of Computation
Fifth week

School of Electrical Engineering & Computing
The University of Newcastle

Exercise 1) Write a regular grammar for each of the following languages:

a) $\{w \in \{\mathtt{a}, \mathtt{b}\}^* : w$ contains an odd number of a's and an odd number of b's$\}$.

$G = (\{EE, EO, OE, OO, \mathtt{a}, \mathtt{b}\}, \{\mathtt{a}, \mathtt{b}\}, EE, R)$, where $R =$

$EE \rightarrow \mathtt{a}\ OE$
$EE \rightarrow \mathtt{b}\ EO$
$OE \rightarrow \mathtt{b}\ OO$
$OE \rightarrow \mathtt{a}\ EE$
$OO \rightarrow \varepsilon$
$EO \rightarrow \mathtt{a}\ OO$
$EO \rightarrow \mathtt{b}\ EE$
$OO \rightarrow \mathtt{a}\ EO$
$OO \rightarrow \mathtt{b}\ OE$

b) $\{w \in \{\mathtt{a}, \mathtt{b}\}^* : w$ does not end in aa$\}$.

$S \rightarrow \mathtt{a}A \mid \mathtt{b}B \mid \varepsilon$
$A \rightarrow \mathtt{a}C \mid \mathtt{b}B \mid \varepsilon$
$B \rightarrow \mathtt{a}A \mid \mathtt{b}B \mid \varepsilon$
$C \rightarrow \mathtt{a}C \mid \mathtt{b}B$

c) $\{w \in \{\mathtt{a}, \mathtt{b}\}^* : w$ contains the substring abb$\}$.

$S \rightarrow \mathtt{a}S \mid \mathtt{b}S \mid \mathtt{a}T$
$T \rightarrow \mathtt{b}W$
$W \rightarrow \mathtt{b}X$
$X \rightarrow \mathtt{a}X \mid \mathtt{b}X \mid \varepsilon$

EXTRAS FROM THE BOOK

d) $\{w \in \{\mathtt{a}, \mathtt{b}\}^* :$ if w contains the substring aa then $|w|$ is odd$\}$.

It helps to begin by rewriting this as:

$\{w \in \{\mathtt{a}, \mathtt{b}\}^* : w$ does not contain the substring aa or $|w|$ is odd$\}$

The easiest way to design this grammar is to build an FSM first. The FSM has seven states:

- S, the start state, is never reentered after the first character is read.
- T_1: No aa yet; even length; last character was a.
- T_2: No aa yet; odd length; last character was a.
- T_3: No aa yet; even length; last character was b.
- T_4: No aa yet; odd length; last character was b.
- T_5: aa seen; even length.
- T_6: aa seen; odd length.

Now we build a regular grammar whose nonterminals correspond to those states. So we have:

$S \rightarrow \mathtt{a}T_2 \mid \mathtt{b}T_4$
$T_1 \rightarrow \mathtt{a}T_6 \mid \mathtt{b}T_4 \mid \varepsilon$
$T_2 \rightarrow \mathtt{a}T_5 \mid \mathtt{b}T_3 \mid \varepsilon$
$T_3 \rightarrow \mathtt{a}T_2 \mid \mathtt{b}T_4 \mid \varepsilon$
$T_4 \rightarrow \mathtt{a}T_1 \mid \mathtt{b}T_3 \mid \varepsilon$
$T_5 \rightarrow \mathtt{a}T_6 \mid \mathtt{b}T_6$
$T_6 \rightarrow \mathtt{a}T_5 \mid \mathtt{b}T_5 \mid \varepsilon$

e) $\{w \in \{\mathtt{a}, \mathtt{b}\}^* : w \text{ does not contain the substring } \mathtt{aabb}\}$.

$S \rightarrow \mathtt{a}A$	$A \rightarrow \mathtt{a}B$	$B \rightarrow \mathtt{a}B$	$C \rightarrow \mathtt{a}A$
$S \rightarrow \mathtt{b}S$	$A \rightarrow \mathtt{b}S$	$B \rightarrow \mathtt{b}C$	$C \rightarrow \varepsilon$
$S \rightarrow \varepsilon$	$A \rightarrow \varepsilon$	$B \rightarrow \varepsilon$	

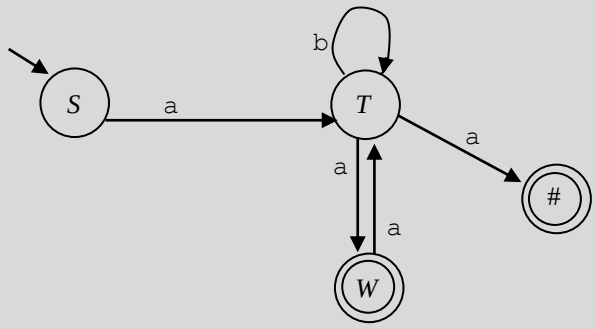
Exercise 2) Consider the following regular grammar *G*:

$S \rightarrow \mathtt{a}T$
$T \rightarrow \mathtt{b}T$
$T \rightarrow \mathtt{a}$
$T \rightarrow \mathtt{a}W$
$W \rightarrow \varepsilon$
$W \rightarrow \mathtt{a}T$

a) Write a regular expression that generates $L(G)$.

$\mathtt{a}\,(\mathtt{b} \cup \mathtt{aa})^*\,\mathtt{a}$

b) Use *grammartofsm* to generate an FSM *M* that accepts $L(G)$.



Exercise 3) Yes. FSM → Regular grammar algorithm is essentially opposite to Regular grammar → FSM algorithm.

Exercise 4) Let $L = \{w \in \{\mathtt{a}, \mathtt{b}\}^* :$ every a in w is immediately followed by at least one $\mathtt{b}\}$.

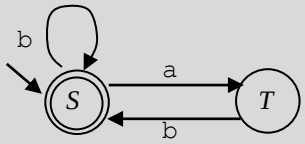
a) Write a regular expression that describes *L*.

$(\mathtt{ab} \cup \mathtt{b})^*$

b) Write a regular grammar that generates *L*.

$S \rightarrow \mathtt{b}S$
$S \rightarrow \mathtt{a}T$
$S \rightarrow \varepsilon$
$T \rightarrow \mathtt{b}S$

c) Construct an FSM that accepts *L*.



Exercise 5) (Exercise 1, of Chapter 8 of Ref. [1]) For each of the following languages L, state whether or not L is regular. Prove your answer.

a) $\{a^i b^j : i, j \geq 0 \text{ and } i + j = 5\}$.

Regular. There is only finite number of strings of a's and b's that totals to 10. So it is regular.

A simple FSM with 10 states accepts L. There is one set of five states that correspond to the case where only a's have been seen. The states count the a's. There is a second set of five states that correspond to the case where at least one b has been seen. These count the total number of characters.

b) $\{a^i b^j : i, j \geq 0 \text{ and } i - j = 5\}$.

Not regular. *L* consists of all strings of the form a*b* where the number of a's is five more than the number of b's. We can show that *L* is not regular by pumping. Let $w = a^{k+5}b^k$. Since $|xy| \leq k$, *y* must equal a^p for some $p > 0$. We can pump *y* out once, which will generate the string $a^{k+5-p}b^k$, which is not in *L* because the number of a's is is less than 5 more than the number of b's.

c) $\{a^i b^j : i, j \geq 0 \text{ and } |i - j| \equiv_5 0\}$.

Regular. Note that $i - j \equiv_5 0$ iff $i \equiv_5 j$. *L* can be accepted by a straightforward FSM that counts a's (mod 5). Then it counts b's (mod 5) and accepts iff the two counts are equal.

d) $\{w \in \{0, 1, \#\}^* : w = x\#y$, where $x, y \in \{0, 1\}^*$ and $|x| \cdot |y| \equiv_5 0\}$. (Let · mean integer multiplication).

Regular. For $|x| \cdot |y|$ to be divisible by 5, either $|x|$ or $|y|$ (or both) must be divisible by 5. So *L* is defined by the following regular expression:
$((0 \cup 1)^5)^*\#(0 \cup 1)^* \cup (0 \cup 1)^*\#((0 \cup 1)^5)^*$, where $(0 \cup 1)^5$ is simply a shorthand for writing $(0 \cup 1)$ five times in a row.

Exercise 6) Could the intersection of two infinite languages be a regular language? Justify your answer.

Yes. For example $a^* \cap a^* = a^*$ and is regular (and infinite). Could it be finite? Yes $a^* \cap b^* = \varepsilon$ which is finite. How about 2 infinite regular languages, could their intersection be non-regular (no!). What about the intersection of 2 non-regulars. Could it be regular? Yes (if it is for example, empty)

Exercise 7) When do we say that a binary relation *R* is *closed under a property* ?

We say a binary relation *R* on a set *A* is closed under a property *P* iff *R* possesses *P*. For example the relation < on the set of all positive integer is closed under *transitivity.*

Exercise 8) Give five examples of the previous definition you have given in Exercise 7 (just above) as applied to languages. For instance: "*The set of even length strings of a's and b's is closed under concatenation.*" Justify your answers.

Thousands of possible answers, some include:
1. L = {w ∈ {0,1}* } is closed under reverse
2. Natural Numbers (integers >= 0) are closed under addition

3. Set of integers (positive and negative) are closed under subtraction
4. The relation “married to” on the set of all married people is closed under symmetry
5. The relation "x is older than y" on the set of all people is closed under transitivity

Exercise 9) Are regular languages closed under intersection? Justify your answer.

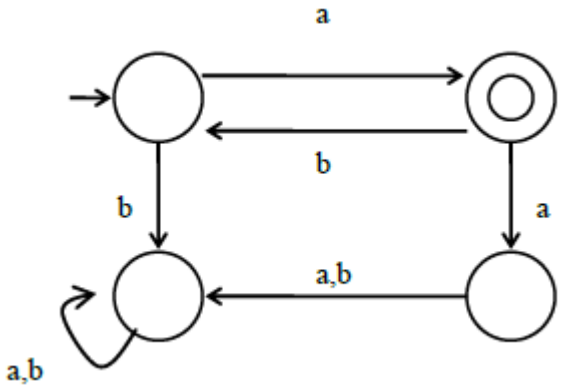
Yes. We know regular languages are closed under union and complement. And intersection can be written in terms of union and complement (De Morgan’s Law: $A \cap B = \neg (\neg A \cup \neg B)$). Therefore, the set of regular languages are closed under intersection.

Exercise 10) (Exercise 20, of Chapter 8 of Ref. [1]) Consider the language $L = \{x0^n y1^n z : n \geq 0, x \in P, y \in Q, z \in R\}$, where P, Q, and R are nonempty sets over the alphabet $\{0, 1\}$. Can you find regular languages P, Q, and R such that L is not regular? Can you find regular languages P, Q, and R such that L is regular?

Let P, Q, $R = \{\varepsilon\}$. Then $L = 0^n1^n$, and so is not regular. On the other hand, let $P = 0^*$, $Q = \{\varepsilon\}$ and let $R = 1^*$. Now $L = 0^*1^*$, which is regular.

Exercise 11) For the following examples describe informally the languages represented by the FSM and write down their regular expressions. You MUST use the algorithm *fsmtoregex* shown in class (page 142 of Ref[1]) and show your work.

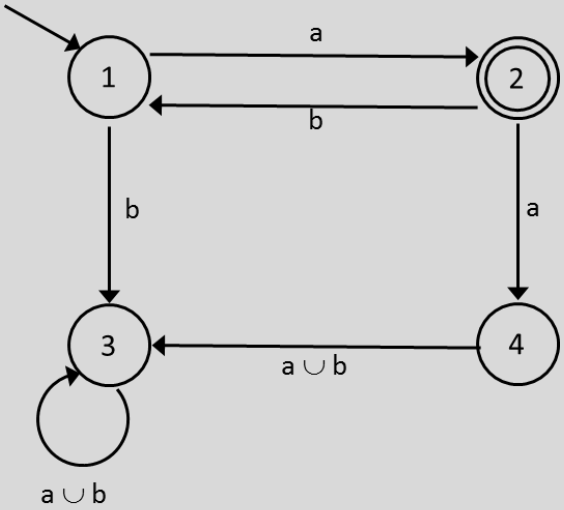
a)



Answer: a(ba)*

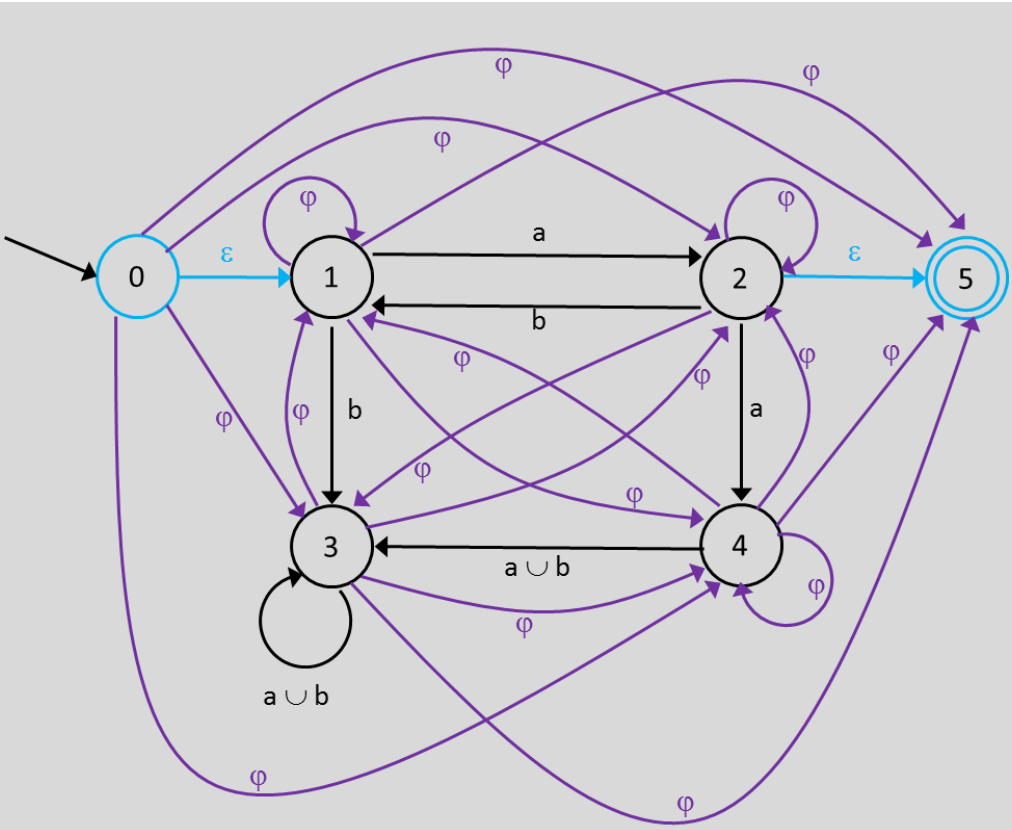
Detailed workout:

(a) Name the symbols and combine multiple transitions between two states using union. E.g. two transitions from State 4 to State 3 on a and b are combined using a ∪ b.



(b) Standradize the FSM by removing unreachable states, adding a new start state (if necessary), a new final state (if necessary) and add φ transitions if a transition is missing. [Check *standardize*(M:FSM) Algorithm 6.2 in page 139~143 of the text or slides of week 4 lecture] This give the following standardized FSM.

(i) State 0 was added as the new start state with an ε transtion to state 1 (original start state)
(ii) State 5 was added as the new final sate with an ε transtion from state 5 (original final state)
(iii) Missing transtions between states were added with φ transitions such that no incoming transtion is added to the start state (state 0) and no outgoing transition is added from the final state (state 5)



Now Ripping State 1:

In the new FSM update the transtion between any two states X, Y (or X to X) as follows

(X,Y) = (X,Y) ∪ (X, R) (R,R)* (R,Y)

Here R is the ripped state and (X,Y) means the label of the transtion from state X to Y.

E.g. In the new FSM the transition (0, 2) will be

(0, 2) = (0, 2) ∪ (0, 1) (1, 1)* (1, 2) [we are ripping state 1]
= φ ∪ ε φ* a
= φ ∪ ε ε a
= φ ∪ a
= a

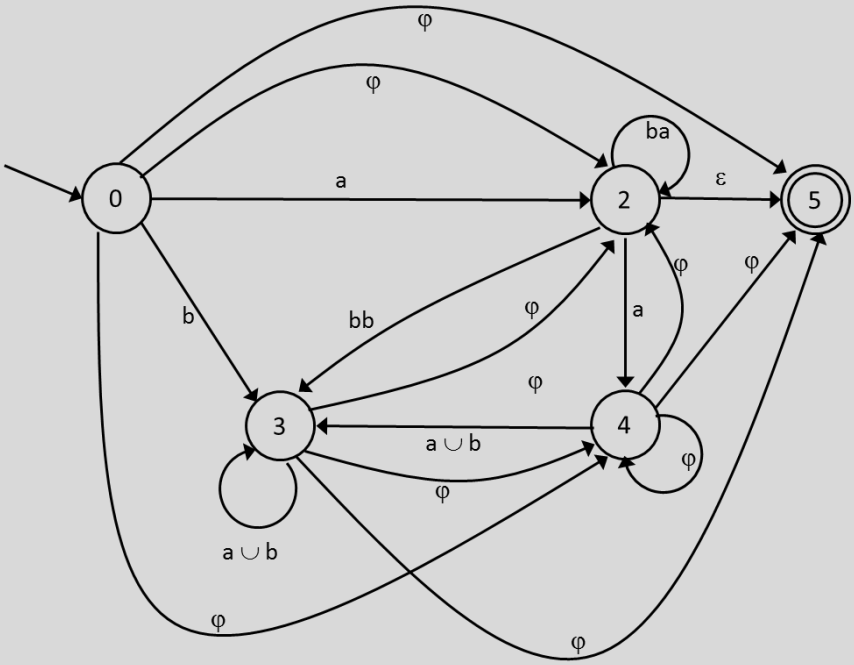
Similarly all other transitions are updated.

(0, 3) = φ ∪ ε φ* b = b
(0, 4) = φ ∪ ε φ* φ = φ
(0, 5) = φ ∪ ε φ* φ = φ

(2, 2) = φ ∪ b φ* a = ba
(2, 3) = φ ∪ b φ* b = bb
(2, 4) = a ∪ b φ* φ = a
(2, 5) = ε ∪ b φ* φ = ε

(3, 2) = φ ∪ φ φ* a = φ
(3, 3) = (a ∪ b) ∪ φ φ* b = (a ∪ b)
(3, 4) = φ ∪ φ φ* φ = φ
(3, 5) = φ ∪ φ φ* φ = φ

(4, 2) = φ ∪ φ φ* a = φ
(4, 3) = (a ∪ b) ∪ φ φ* b = (a ∪ b)
(4, 4) = φ ∪ φ φ* φ = φ
(4, 5) = φ ∪ φ φ* φ = φ

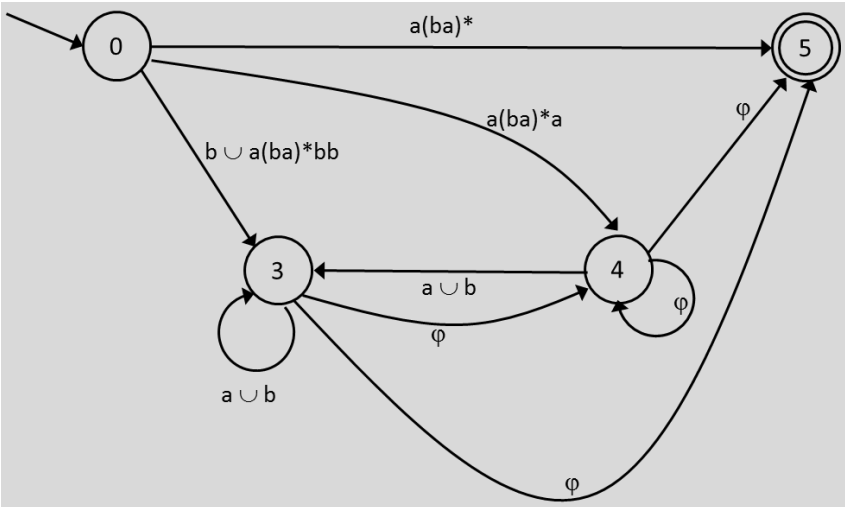


Now Rip state 2:

(0, 3) = b ∪ a (ba)* bb = b ∪ a (ba)* bb
(0, 4) = φ ∪ a (ba)* a = a (ba)*a
(0, 5) = φ ∪ a (ba)* ε = a (ba)*

(3, 3) = (a ∪ b) ∪ φ (ba)* bb = (a ∪ b)
(3, 4) = φ ∪ φ (ba)* a = φ
(3, 5) = φ ∪ φ φ* ε = φ

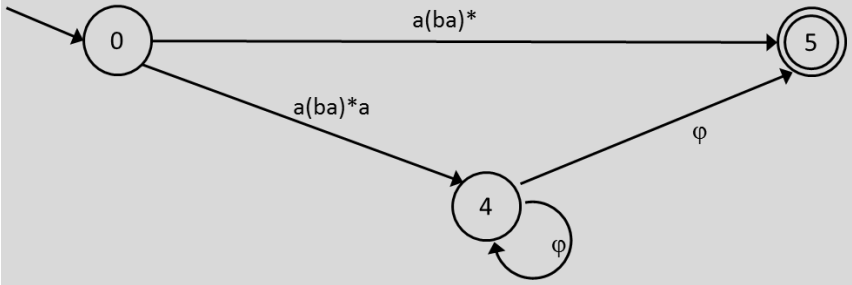
(4, 3) = (a ∪ b) ∪ φ (ba)* bb = (a ∪ b)
(4, 4) = φ ∪ φ (ba)* a = φ
(4, 5) = φ ∪ φ (ba)* ε = φ



Now Rip state 3:

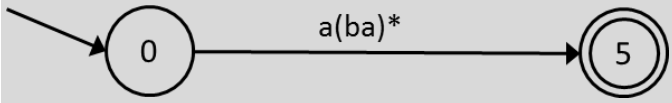
(0, 4) = (a(ba)*a) ∪ ((b ∪a(ba)*bb) (a∪b)* φ) = a (ba)*a
(0, 5) = (a(ba)*) ∪ (b ∪a(ba)*bb) (a∪b)* φ= a (ba)*

(4, 4) = φ ∪ ((a∪b) (a∪b)* φ) = φ
(4, 5) = φ ∪ ((a∪b) (a∪b)* φ)= φ



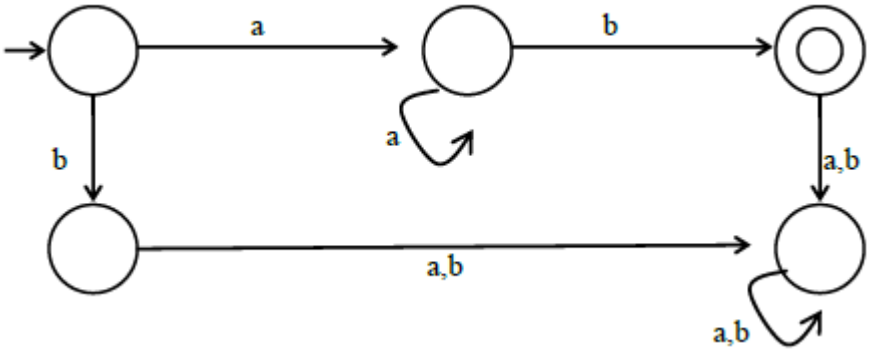
Now Rip state 4:

(0,5) = (a(ba)*) ∪ ((a(ba)*a) φ*φ) = a (ba)*



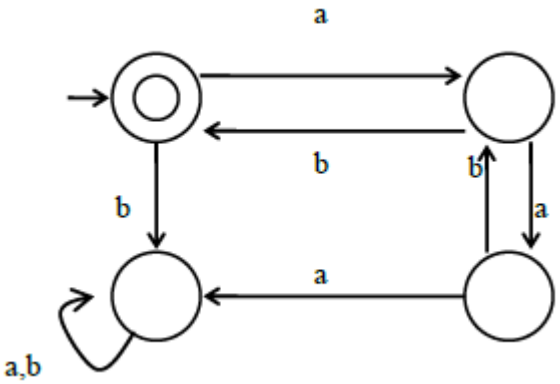
So the regular expression for the FSM is given by: a(ba)*

b)



$a^{+}b$

c)



(ab ∪ aa (ba)*bb)*