COM 2030 Exercises 1:
Finite Automata and Regular Languages
Solutions

1. Consider the following regular grammar with start symbol A:

$$\begin{aligned} A &\rightarrow aB \\ A &\rightarrow a \\ B &\rightarrow bC \\ B &\rightarrow b \\ C &\rightarrow cB \\ C &\rightarrow c \end{aligned}$$

(a) Derive a transition diagram for a finite automaton that accepts the language generated by this grammar using the technique suggested in the proof of Proposition 3.1.

Proposition 3.1: For each alphabet Σ,

$$\{L(G) : G \text{ is a regular grammar over } \Sigma\} = \{L(M) : M \text{ is a finite automata over } \Sigma\}$$

Let the G be the grammar $(\Sigma_N, \Sigma_T, A, R)$ where $\Sigma_N = \{A, B, C\}$, $\Sigma_T = \{a, b, c\}$ and R is the set of rewrite rules above.

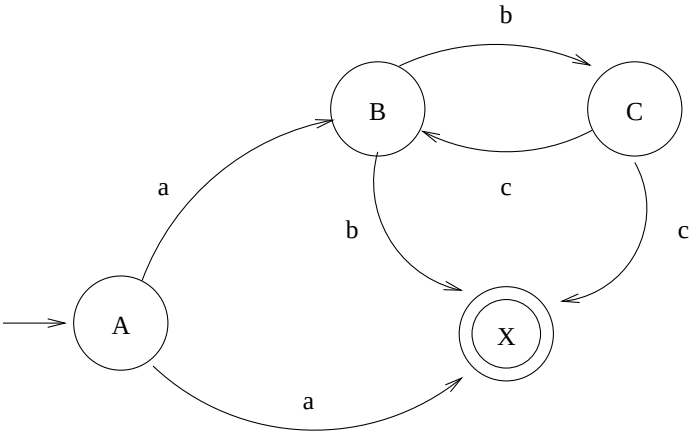
Define an equivalent grammar G' by replacing all rewrite rules in G of the form $X \rightarrow P$ where P is a terminal, by two rules $X \rightarrow PY$ and $Y \rightarrow \lambda$ where Y is a new nonterminal. So in this case we get:

$$\begin{aligned} A &\rightarrow aB \\ A &\rightarrow aX \\ X &\rightarrow \lambda \\ B &\rightarrow bC \\ B &\rightarrow bX \\ C &\rightarrow cB \\ C &\rightarrow cX \end{aligned}$$

Then, define M to be the nondeterministic finite automaton $(S, \Sigma, \rho, \iota, F)$ where

i. S is the set of nonterminals in G';
ii. ι is the start symbol in G';
iii. F is the set of nonterminals of G' that appear on the left of a λ-rule;
iv. ρ consists of the triples (P, x, Q) for which G' contains the a rewrite rule of the form $P \rightarrow xQ$.

Thus:



(b) Formally define a deterministic finite automaton that will accept the same strings as this nondeterministic finite automaton using the technique of Proposition 2.4 and draw a transition diagram for the derived deterministic finite automaton.

The solution relies on the construction in the proof of Proposition 2.4: for each nondeterministic finite automaton M there is a deterministic finite automaton M' that accepts exactly the same language.

Following the construction in the proof let $M' = (S', \Sigma, \delta, \iota', F')$ be a DFA constructed in the following way from M of part (a).

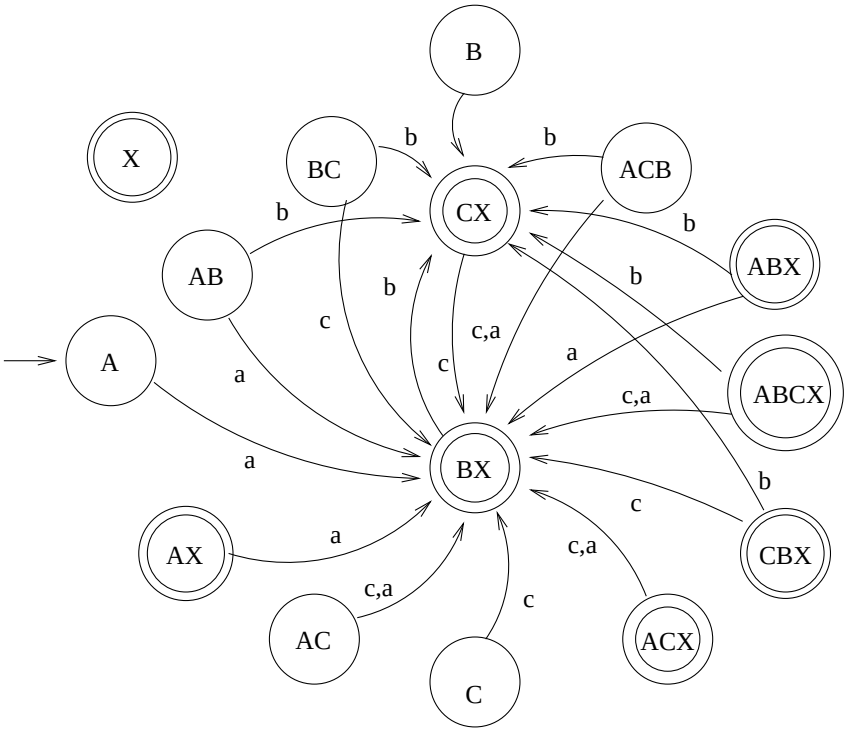
i. $S' = \mathcal{P}(S)$ (the powerset of S);

ii. $\iota' = \{\iota\}$;

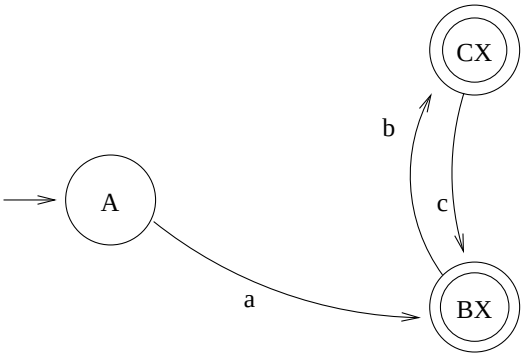
iii. F' is the set of subsets of S that contain at least state in F;

iv. δ is a function from $S' \times \Sigma$ into S' such that for each symbol $x \in \Sigma$ and each state $s' \in S'$, $\delta(s', x)$ is the set of all $s \in S$ such that $(u, x, s) \in \rho$ for some $u \in s'$ (so, $\delta(s', x)$ is the set of all states in S that can be reached from a state in s' over an arc labelled x).

The transition diagram for M' looks like this, where the empty state in S' is not shown and all transitions not shown are to the empty state.



By eliminating nonreachable states, we get the following deterministic automaton:



(c) Derive a regular expression representing the language generated by this grammar using the technique suggested in the proof of Proposition 4.1.

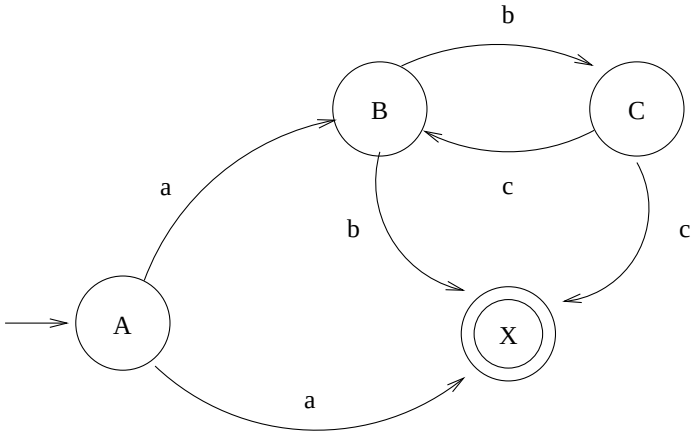
Proposition 4.1: Given an alphabet Σ, the regular languages over Σ are exactly the languages that are represented by regular expressions over Σ. The proof provides a construction for generating a regular expression from a transition diagram for a finite automaton M. The proof involves using generalised FA's in which transitions may be labelled with REs. It proceeds by reducing the number of states in M one at a time, increasing the complexity of the RE's in the transitions until there is a two-state FA when the final RE may be read off.

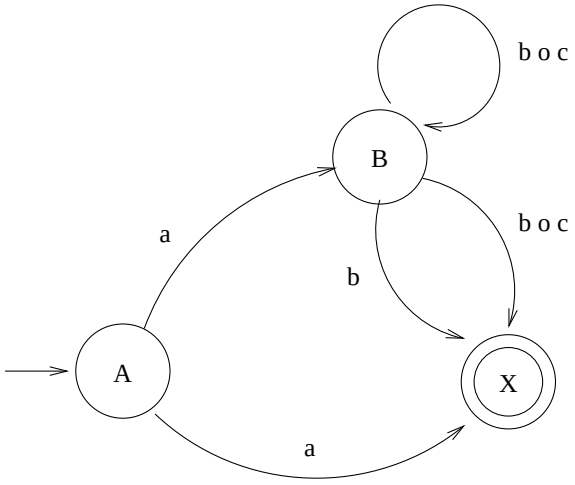
The construction requires we work with a transition diagram with only one accept state. If there is more than one we take copies of the original but with one accept state each then find the RE for each and then take the union of the RE's.

To eliminate a state X proceed as follows. First, replace all arcs which orginate in X and terminate in X with a single arc whose label is the union of the REs on the original arcs.

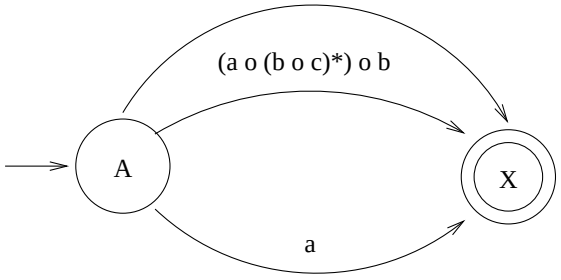
Call this RE r. Next, for each arc terminating in X whose label is the RE p and each arc originating in X whose label is the RE q replace this pair of arcs with a single arc from the originating state of the first arc to the terminating state of the second arc with the RE $((p \circ r^*) \circ q)$.

The following sequence shows the elimination of states C and B.

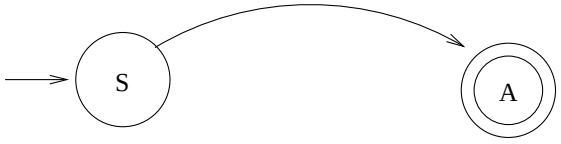




(a o (b o c)*) o (b o c)



a U (a o (b o c)*) o b U (a o (b o c)*) o (b o c)



When only the start and end states remain, then in this simple scenario we take the union of the REs on the arcs from the start state to the accept state.
Thus, the RE for M and hence for the language in 1. is

$$a \cup ((a \circ (b \circ c)^*) \circ b) \cup ((a \circ (b \circ c)^*) \circ (b \circ c))$$

2. (a) Write a regular expression that represents the language of zero or more a's followed by one or more b's followed by zero or more a's.

$$a^* \circ (b \circ b^*) \circ a^*$$

(b) Write a regular expression that represents the language consisting of one or more pairs ab where between any two instances of ab exactly one c may or may not occur.

$$(a \circ b) \circ ((a \circ b) \cup (c \circ (a \circ b)))^*$$

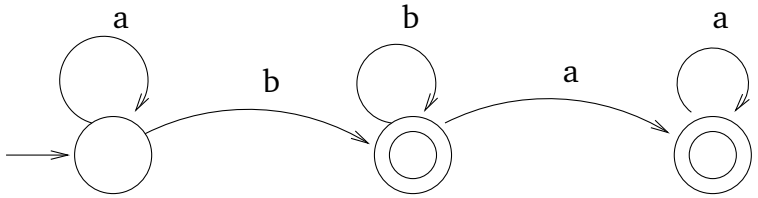
(c) Draw a transition diagram for a finite automaton that accepts

i. the language in (a);

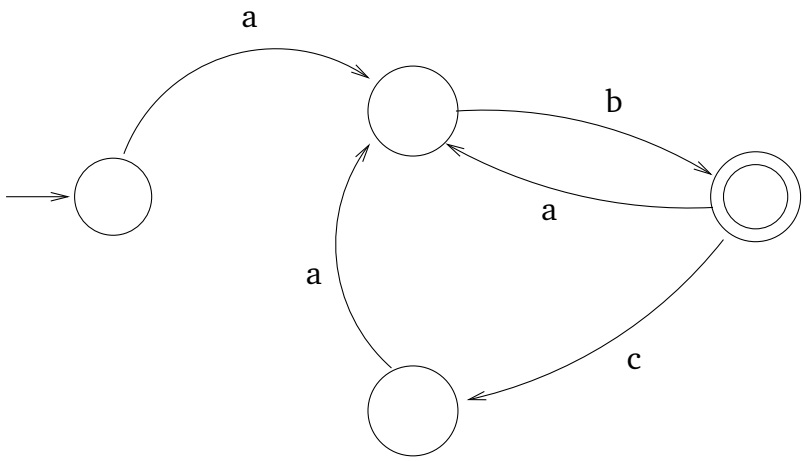
ii. the language in (b);

iii. the Kleene star of the union of the languages in (a) and (b) – this automaton should be constructed according to the principles for constructing automata corresponding to regular expressions demonstrated in Part 1 of the proof for Proposition 4.1.

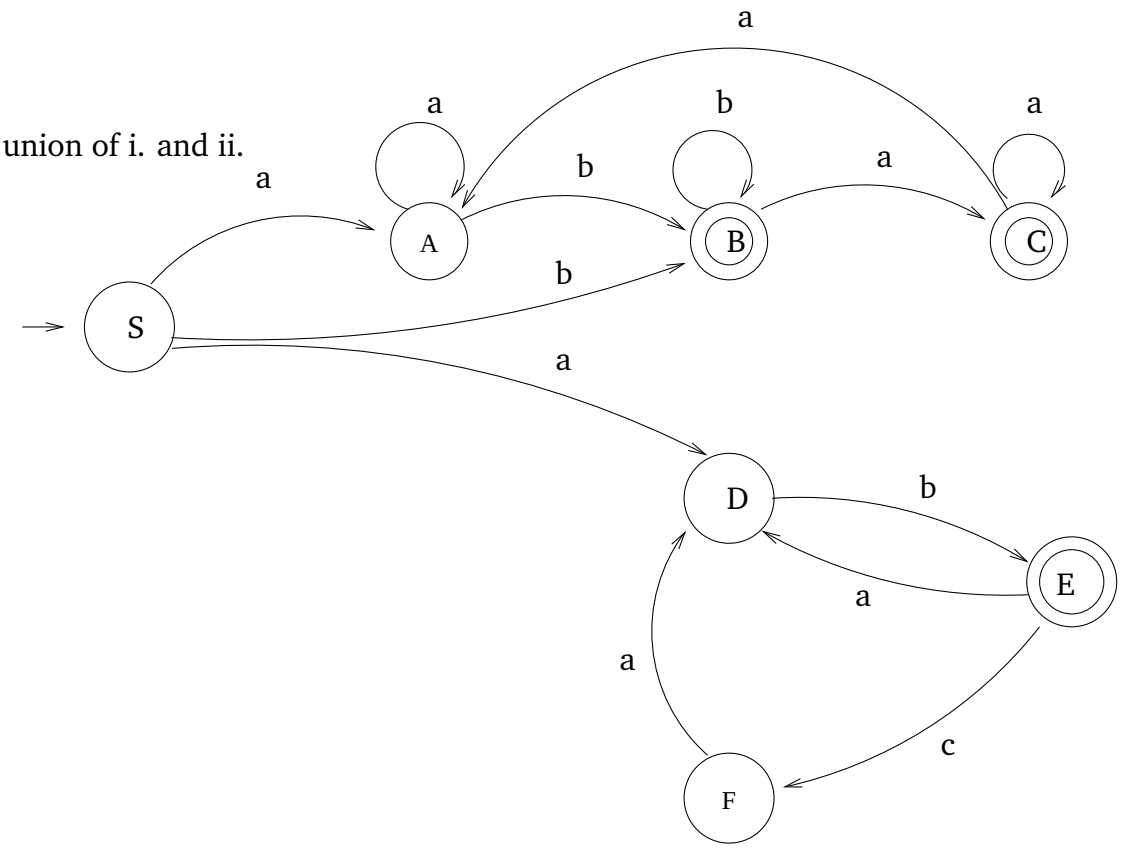
i



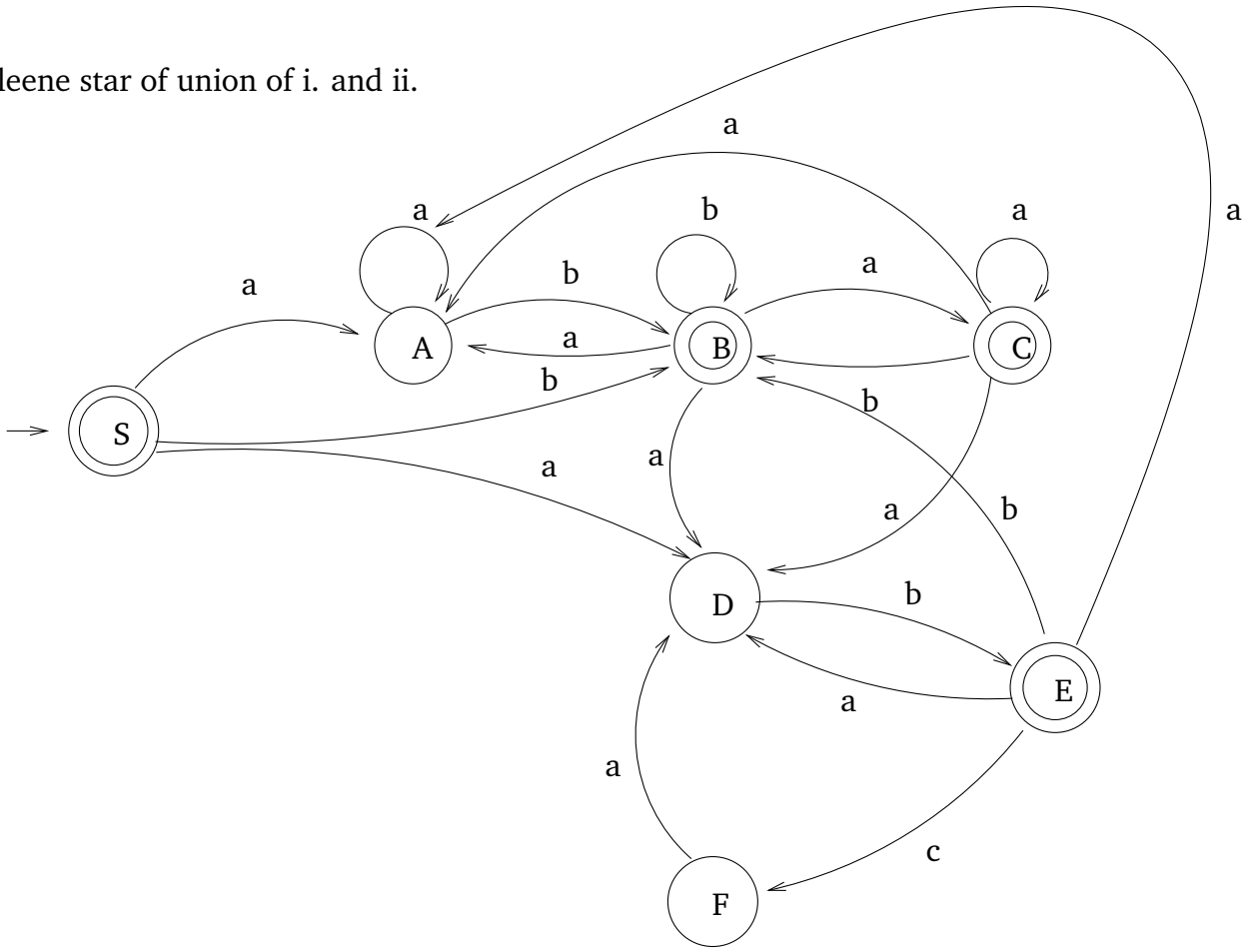
ii.



iii. union of i. and ii.



iii Kleene star of union of i. and ii.



(d) Using the technique of Proposition 3.1 derive a regular grammar that generates the same language as that accepted by the automaton in (c) iii.

$$
\begin{aligned}
S &\rightarrow \lambda \\
S &\rightarrow aA \\
S &\rightarrow bB \\
S &\rightarrow aD \\
A &\rightarrow aA \\
A &\rightarrow bB \\
B &\rightarrow aA \\
B &\rightarrow bB \\
B &\rightarrow bC \\
B &\rightarrow aD \\
B &\rightarrow \lambda \\
C &\rightarrow aA \\
C &\rightarrow bB \\
C &\rightarrow aC \\
C &\rightarrow aD \\
C &\rightarrow \lambda \\
D &\rightarrow bE \\
E &\rightarrow aD \\
E &\rightarrow cF \\
E &\rightarrow aA \\
E &\rightarrow bB \\
E &\rightarrow \lambda \\
F &\rightarrow aD
\end{aligned}
$$

For each transition in the transition diagram from state X to state Y with arc label z add a rule $X \rightarrow zY$. For each accept state Z add a rule $Z \rightarrow \lambda$.

This grammar generates the same strings as the automaton accepts since for each path through the automaton to the an accept state there is a corresponding derivation from this grammar of the string labelling the automaton's path.

3. (a) Show that if L_1 and L_2 are regular languages then $L_1 \cap L_2$ is regular.

Since L_1 and L_2 are regular there are deterministic finite automata M_1 and M_2 such that $L(M_1) = L_1$ and $L(M_2) = L_2$. We assume that δ_1 and δ_2 are total; i.e., for every state $s_1 \in S_1$ and every symbol $x_1 \in \Sigma_1$, $\delta_1(s_1, x_1)$ is defined and is in S_1 (similarly for δ_2). The transition diagram for any machine can be transformed into one which depicts a total function by adding, if necessary, a dummy state at which all arcs from each state which have not been specified in the original diagram terminate.

Let $M_1 = (S_1, \Sigma_1, \delta_1, \iota_1, F_1)$ and $M_2 = (S_2, \Sigma_2, \delta_2, \iota_2, F_2)$.

Define a new machine $M = (S, \Sigma, \delta, \iota, F)$ where

$$\begin{aligned} S &= S_1 \times S_2 \\ \Sigma &= \Sigma_1 \cap \Sigma_2 \\ \iota &= (\iota_1, \iota_2) \\ F &= \{(h_1, h_2) \in S \mid h_1 \in F_1 \text{ and } h_2 \in F_2\} \\ \delta &= \{(((p_1, p_2), x), (q_1, q_2)) \mid (p_1, p_2), (q_1, q_2) \in S, x \in \Sigma, \\ &\qquad \delta_1(p_1, x) = q_1 \text{ and } \delta_2(p_2, x) = q_2\} \end{aligned}$$

If a string is accepted by both M_1 and M_2 then we can write down two sequences of transitions corresponding to the paths through each machine such that if we pair the states at the same position in each transition sequence we get a path through M which terminates at a paired state each of whose component states is an accept state in the corresponding original machine and which is therefore an accept state in M. Hence each string accepted by M_1 and M_2 is accepted by M.

If a string is accepted by M then given the sequence of transitions through M we can write down two sequences of transitions one corresponding to a sequence of transitions through M_1 the other to one through M_2 ending in each case in an accept state of the relevant machine. Hence each string accepted by M is accepted by both M_1 and M_2.

It follows that $L(M) = L(M_1) \cap L(M_2) = L_1 \cap L_2$ Since M is a finite automaton $L(M)$ is regular; i.e. $L_1 \cap L_2$ is regular.

(b) Show that if L_1 and L_2 are regular languages then $L_1 - L_2$ is regular.

Since L_1 and L_2 are regular there are deterministic finite automata M_1 and M_2 such that $L(M_1) = L_1$ and $L(M_2) = L_2$. Let $M_1 = (S_1, \Sigma_1, \delta_1, \iota_1, F_1)$ and $M_2 = (S_2, \Sigma_2, \delta_2, \iota_2, F_2)$. Make the same assumption about δ_1 and δ_2 being total as in part (a).

Define a new machine $M = (S, \Sigma, \delta, \iota, F)$ where

$$\begin{aligned} S &= S_1 \times S_2 \\ \Sigma &= \Sigma_1 \\ \iota &= (\iota_1, \iota_2) \\ F &= \{(h_1, h_2) \in S \mid h_1 \in F_1 \text{ and } h_2 \notin F_2\} \\ \delta &= \{(((p_1, p_2), x), (q_1, q_2)) \mid (p_1, p_2), (q_1, q_2) \in S, x \in \Sigma, \\ &\qquad \delta_1(p_1, x) = q_1 \text{ and } \delta_2(p_2, x) = q_2\} \end{aligned}$$

If a string is accepted by M_1 and not by M_2 then we can write down a sequence of transitions corresponding to the path through M_1 and ending in an accept state in M_1 and a sequence of transitions from M_2 which does not end in an accept state. This pair of sequences corresponds to a path through M which terminates in an accept state of M since the accept states of M will be those states whose compound labels name an accept state of M_1 and states of M_2 that are not accept states. Hence if a string is accepted by M_1 and not by M_2 then it will be accepted by M.

If a string is accepted by M then given the sequence of transitions through M we can write down the two sequences of transitions through M_1 and M_2 which these machines would perform on being presented with the string. Note that each accept state in M contains an accept state in M_1 but not one in M_2. Hence each string accepted by M is accepted by M_1 but not by M_2.

It follows that $L(M) = L(M_1) - L(M_2) = L_1 - L_2$ Since M is a finite automaton $L(M)$ is regular; i.e. $L_1 - L_2$ is regular.

(c) Show that if L is a regular language then the language obtained from writing the strings of L backwards is also regular.

Since L is regular there is a finite automaton M such that $L(M) = L$. $M = (S, \Sigma, \delta, \iota, F)$. From M we can construct a new automaton M' as follows:

i. reverse the direction of all the arcs in M;

ii. create a new start state ι' for M', remove all of M's accept states and draw an arc from ι' to each state in M' from which an arc leading to an accept state in M had originated and label it with the same symbol it had in in M;

iii. change ι into the single accept state of M'.

M' accepts exactly the strings that M accepted written backwards. Since M' is a finite automaton it follows that the language it accepts is regular. Therefore the language obtained by writing the strings of L backwards is regular.