Properties of Regular Languages

Closure properties for Regular Languages (RL)

This is different from Kleene closure

- ***Closure property:***
  - If a set of regular languages are combined using an operator, then the resulting language is also regular
- Regular languages are ***closed*** under:
  - Union, intersection, complement, difference
  - Reversal
  - Kleene closure
  - Concatenation
  - Homomorphism
  - Inverse homomorphism

RLs are closed under union

- IF L and M are two RLs THEN:
  - they both have two corresponding regular expressions, R and S respectively
  - (L U M) can be represented using the regular expression R+S
  - Therefore, (L U M) is also regular

How can this be proved using FAs?

L1 : DFA start with a and end with b

L2 : DFA start with b and end with a

L1 U L2: DFA start with a or b and end with a or b // all strings begin and end with different symbols.

Closure Under Concatenation

- R and S is a regular expression whose language is L and M respectively

L.M : R.S   Which is regular

<u>Concatenation of two DFA</u>

L1: DFA with all strings start with a

L2: DFA with all strings end with b

L1.L2: DFA with all strings that start with a then end with b

Closure Under Kleene Closure

- R is a regular expression whose language is L.

Then R* is also a regular expression

RLs are closed under intersection

- A quick, indirect way to prove:

  By DeMorgan's law:

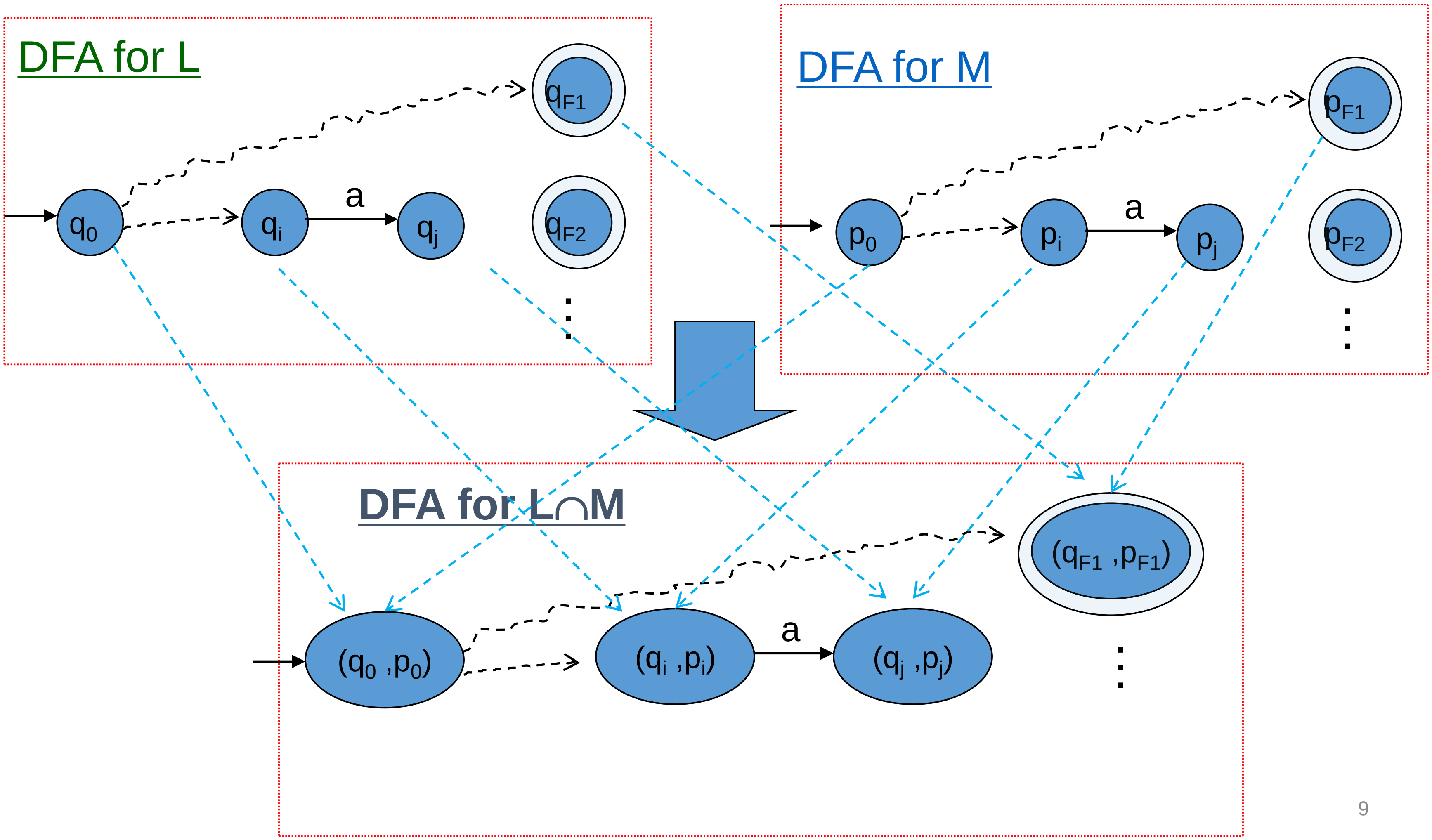
  $\overline{L} \cup \overline{M} = (L \cap M)$

  - Since we know RLs are closed under union and complementation, they are also closed under intersection
- A more direct way would be construct a finite automaton for $L \cap M$

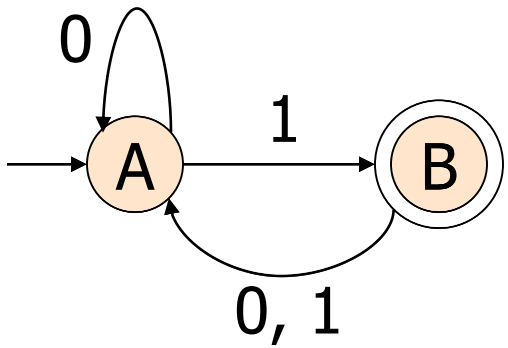
DFA construction for $L \cap M$

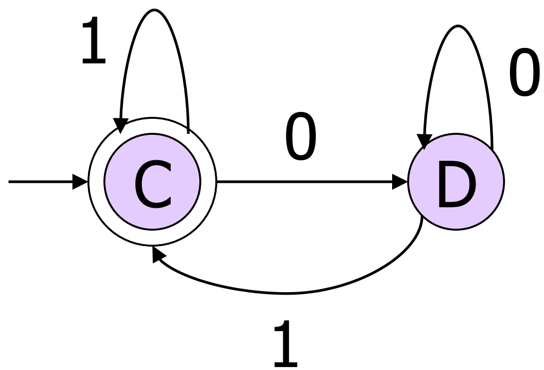
- A_L = DFA for L = $\{Q_L, \Sigma, q_L, F_L, \delta_L\}$
- A_M = DFA for M = $\{Q_M, \Sigma, q_M, F_M, \delta_M\}$
- Build $A_{L \cap M} = \{Q_L x\ Q_M, \Sigma, (q_L, q_M), F_L x\ F_M, \delta\}$ such that:
  - $\delta((p,q),a) = (\delta_L(p,a), \delta_M(q,a))$, where p in Q_L, and q in Q_M
- This construction ensures that a string w will be accepted if and only if w reaches an accepting state in both input DFAs.

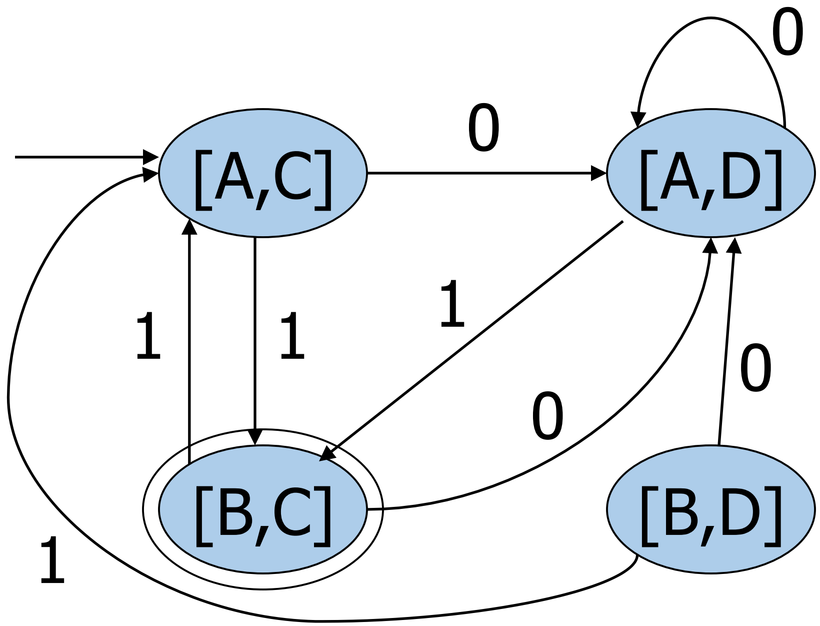
DFA construction for L ∩ M



Example: Product DFA for Intersection







DFA of L1 = {w in {a,b}*| where w has even number of a's}

DFA of L2 = {w in {a,b}*| where w has odd number of b's}

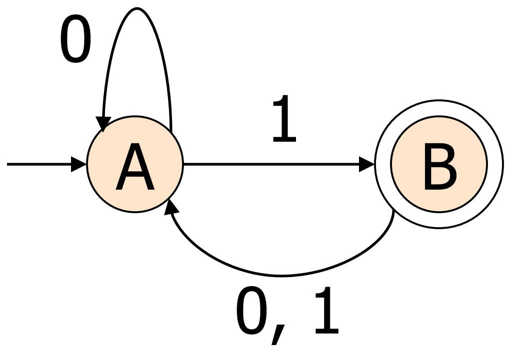
L1 ∩ L2: {w in {a,b}*| where w has even number of a's and odd number of b's}

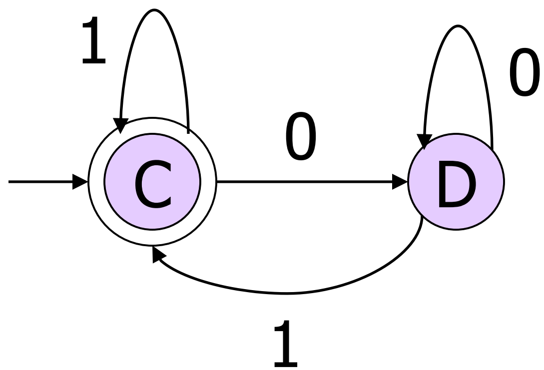
Closed under set difference

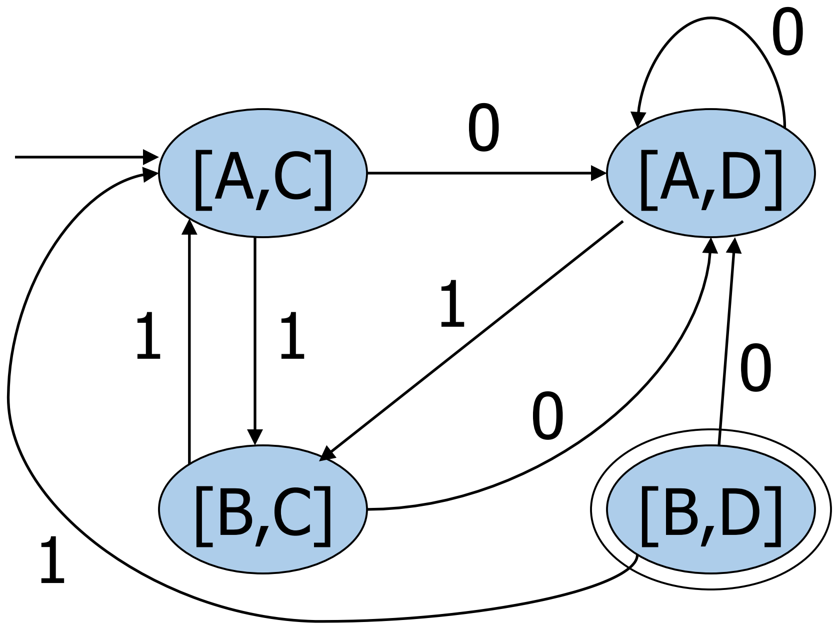
- Observation: $L - M = L \cap \overline{M}$

Therefore, L - M is also regular

Example: DFA for Difference (A-B)







Make the final states, the pairs where A-state is final but B-state is not

Notice: difference is the empty language

L1: DFA accept set of strings a’s and b’s where number of a’s is divisible by 3

L2: DFA accept all strings with even number of b’s over input {a,b}

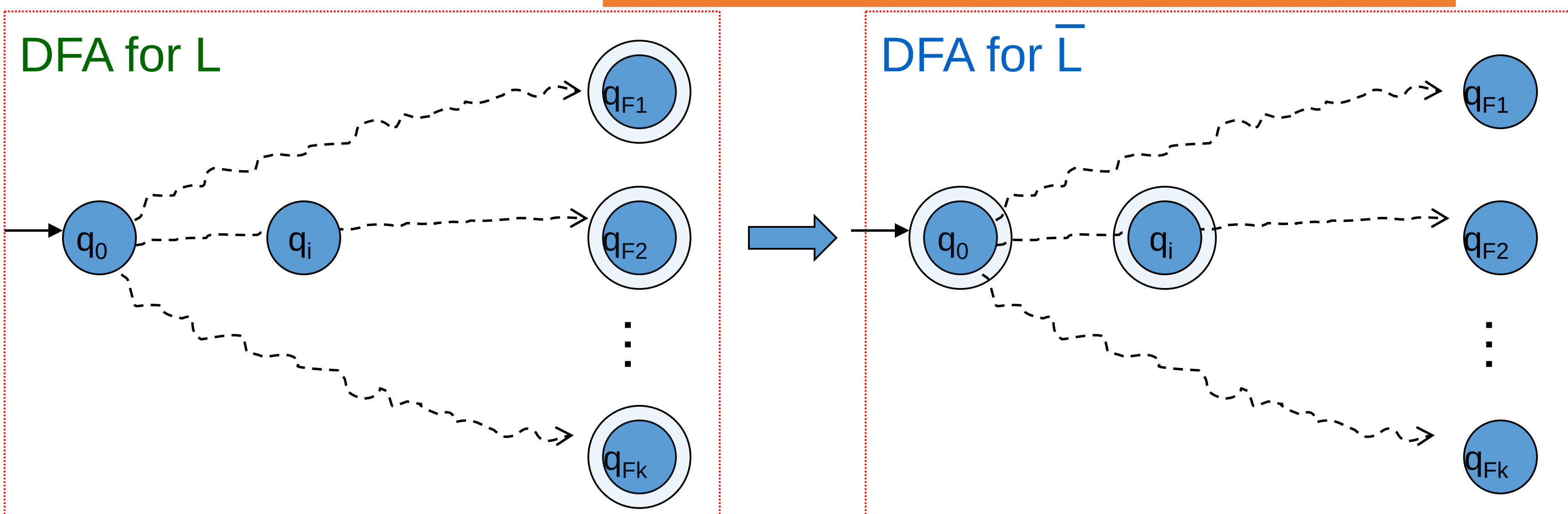
L1 - L2: DFA accept set of strings a’s and b’s where number of a’s is divisible by 3 and the number of b’s is not even (number of b’s is odd)

RLs are closed under complementation

- If L is an RL over ∑, then $\overline{L}$=∑*-L

➢ To show $\overline{L}$ is also regular, make the following construction

Convert every final state into non-final, and every non-final state into a final state



Assumes q0 is a non-final state. If not, do the opposite.

RLs are closed under reversal

Reversal of a string w is denoted by w^R

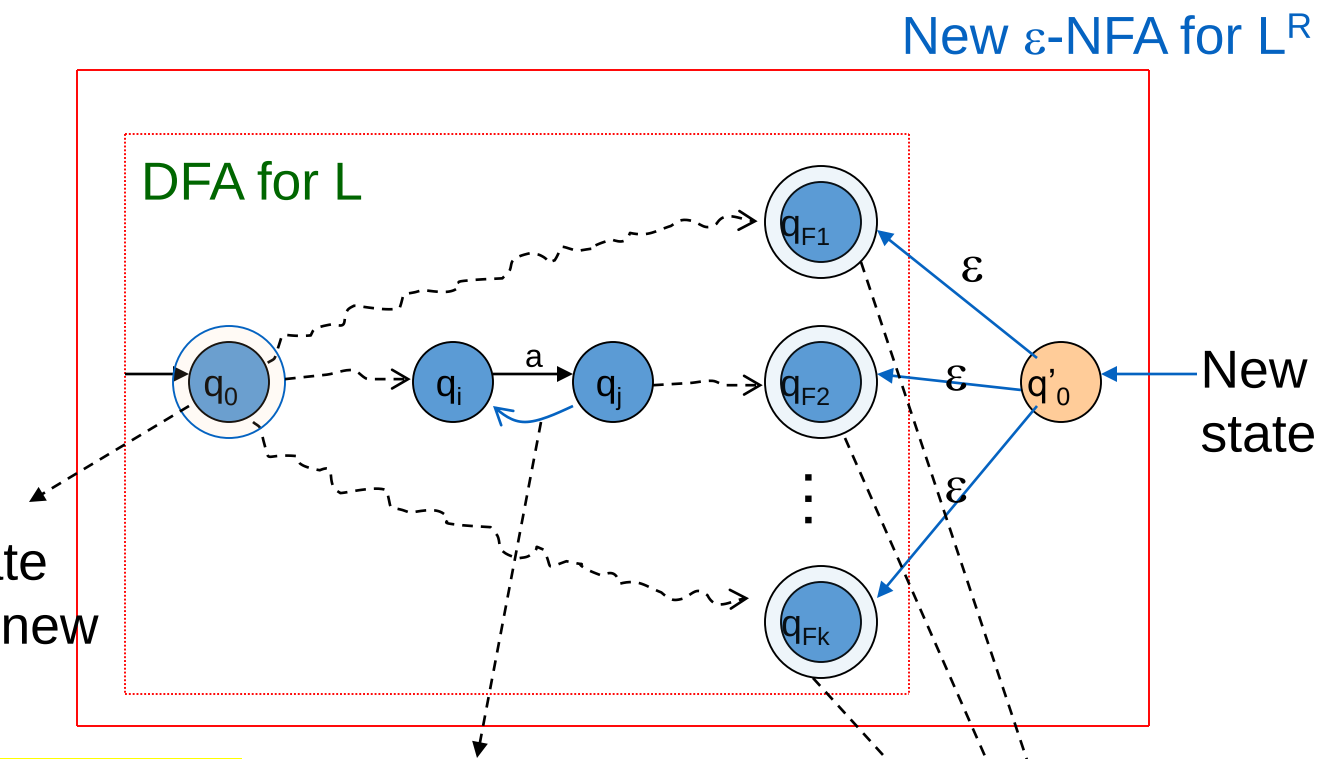
- E.g., w=00111, w^R=11100

Reversal of a language:

- L^R = The language generated by reversing all strings in L

Theorem: If L is regular then L^R is also regular

ε -NFA Construction for L^R



New start state

Make the old start state as the only new final state

What to do if q_0 was one of the final states in the input DFA?

Reverse all transitions

Convert the old set of final states into <u>non-final</u> states

If L is regular, L^R is regular (proof using regular expressions)

- Let E be a regular expression for L
- Given E, how to build E^R?
- <u>Basis:</u> If E= ε, Ø, or a, then E^R=E
- <u>Induction:</u> Every part of E (refer to the part as "F") can be in only *one* of the three following forms:

1. $F = F_1+F_2$
   - $F^R = F_1^R+F_2^R$
2. $F = F_1F_2$
   - $F^R = F_2^R F_1^R$
3. $F = (F_1)^*$
   - $(F^R)^* = (F_1^R)^*$

Example: Reversal of a RE

- Let E = $\mathbf{01^* + 10^*}$.
- $E^R = (\mathbf{01^* + 10^*})^R = (\mathbf{01^*})^R + (\mathbf{10^*})^R$
- $= (\mathbf{1^*})^R\mathbf{0}^R + (\mathbf{0^*})^R\mathbf{1}^R$
- $= (\mathbf{1}^R)^*\mathbf{0} + (\mathbf{0}^R)^*\mathbf{1}$
- $= \mathbf{1^*0 + 0^*1}$.

Homomorphisms

- A *homomorphism* is a function on strings that works by substituting a particular string for each symbol.

- Example: h(0) = ab; h(1) = b.