Properties of Regular Languages

Shakir Ullah Shah

Lecture Objective

- **Closure Properties**
- **Complementation**
- **Intersection**

Closure Properties

- A language that can be defined by a regular expression is called a **regular language**.
- Not all languages are regular
- In this lecture we will focus on the class of all regular languages and discuss some of their properties.

Theorem 10

If L_1 **and** L_2 **are regular languages, then** $L_1 + L_2$**,** L_1L_2**, and** L^*_1 **are also regular languages.**

Notes:

- $L_1 + L_2$ is the language of all words in either L_1 or L_2.
- L_1L_2 is the product language of all words formed by concatenating a word form L_1 with a word from L_2.
- L_1* is the language of all words that are the concatenation of arbitrarily many factors from L_1*.
- The result stated in Theorem 10 is often expressed as **“The set of regular languages is *closed* under union, concatenation, and Kleene closure”.**

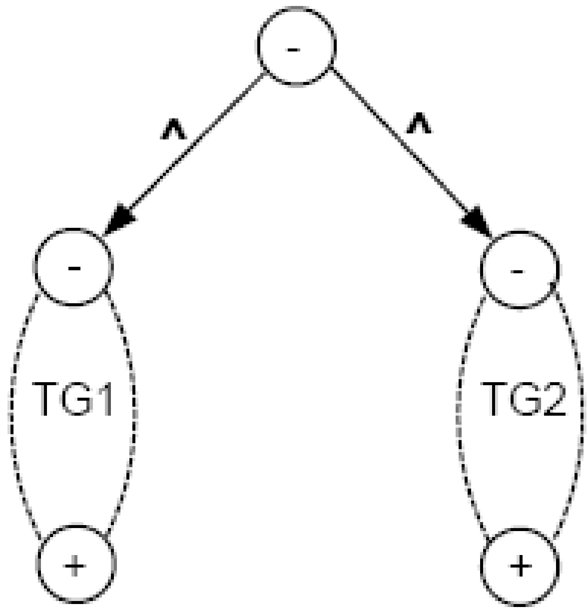
Proof by Machines

- Because L_1 and L_2 are regular languages, there must be TGs that accept them (by Kleene's theorem).
- Let TG_1 accepts L_1 and TG_2 accepts L_2.

- Assume that TG_1 and TG_2 each have a **unique start state** and a **unique separate final state**. If this is not the case originally, then we can modify the TGs so that this becomes true as in Kleene's theorem, Part 2 of the proof

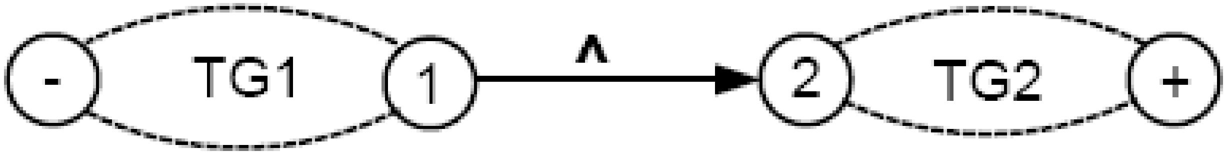
Proof contd.

- Then the TG described below accepts the language $L_1 + L_2$.



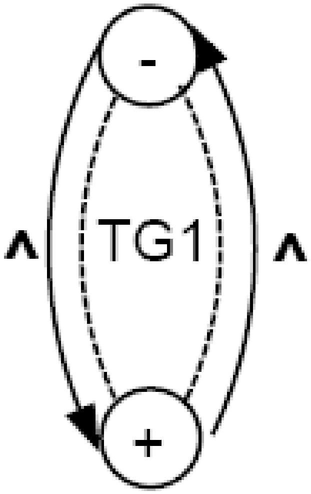
- By Kleene's theorem, since $L_1 + L_2$ is defined by this TG, it is also defined by a regular expression and hence is a regular language.

- The TG described below accepts the language L_1L_2 where state 1 is the former + of TG_1 and state 2 is the former - of TG_2.



- Since L_1L_2 is defined by this TG, it is also defined by a regular expression by Kleene's theorem, and therefore it is a regular language.

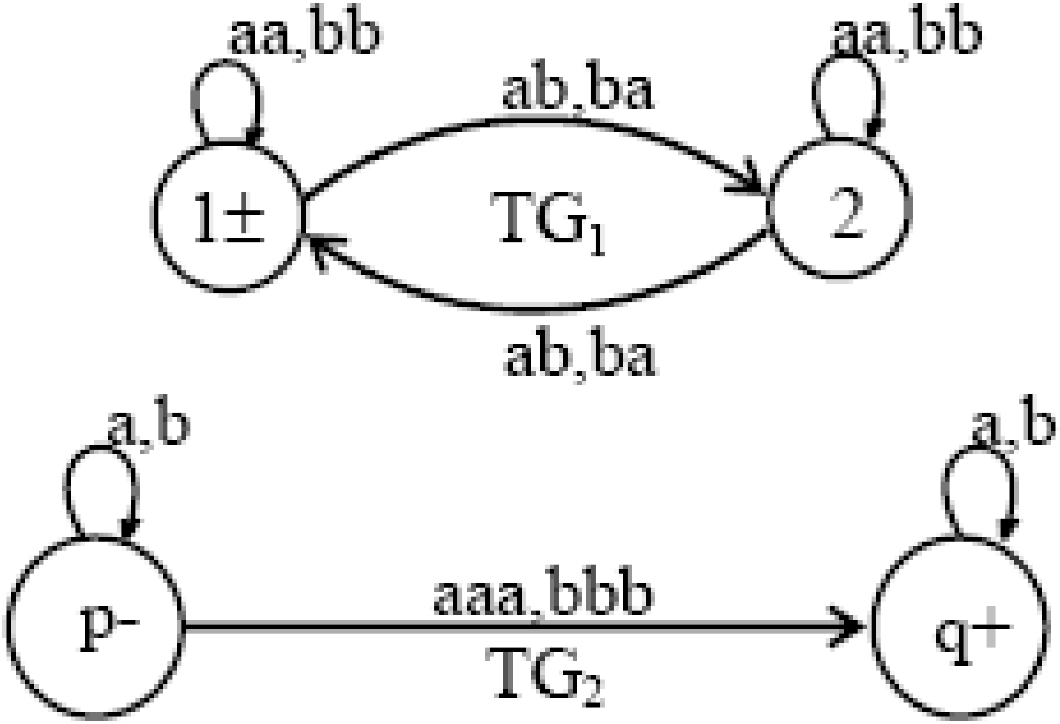
- The TG described below accepts the language L_1*.



- We begin at the - state and trace a path to the + state of TG1. At this point, we cold stop and accept the string or jump back, at no cost, to the - state and run another segment of the input string.

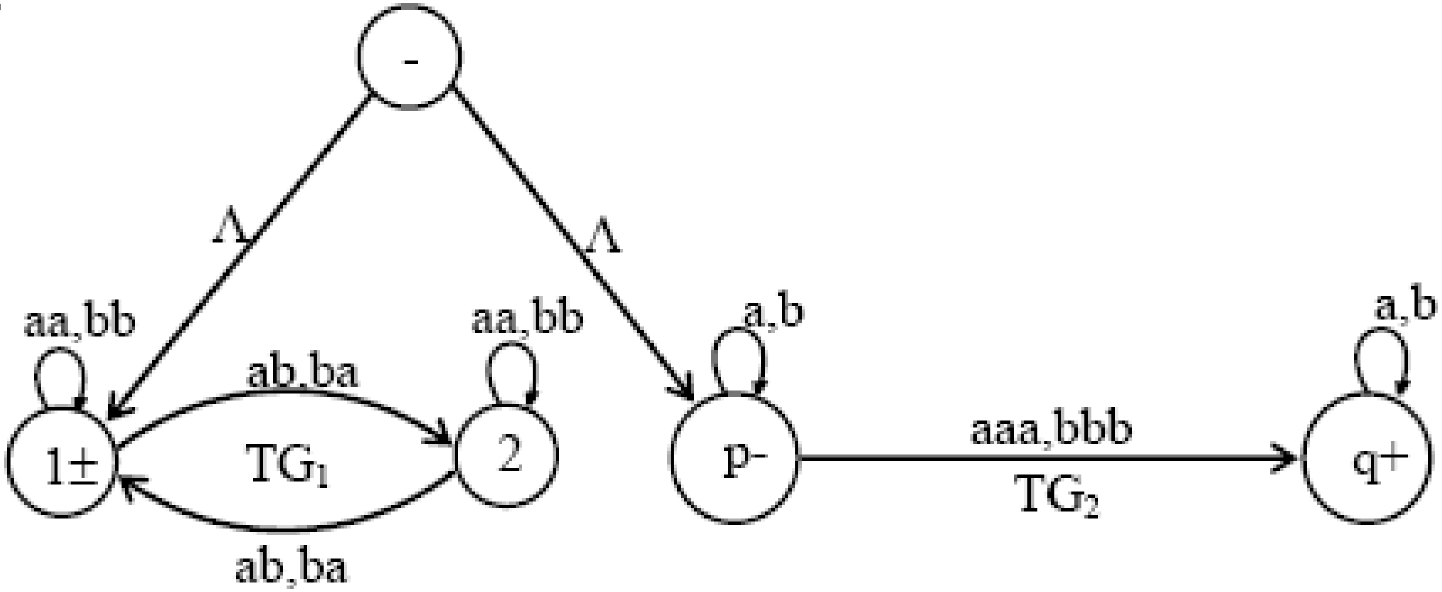
Example

- Consider the following TGs

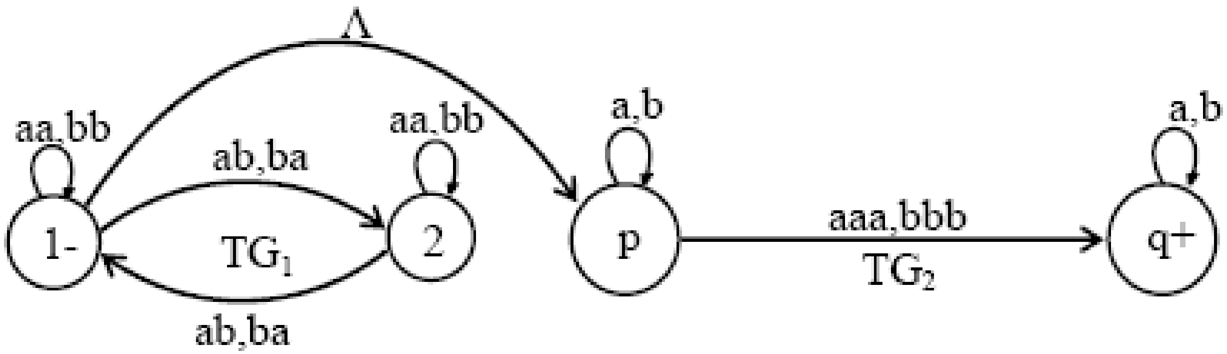


Example cont…

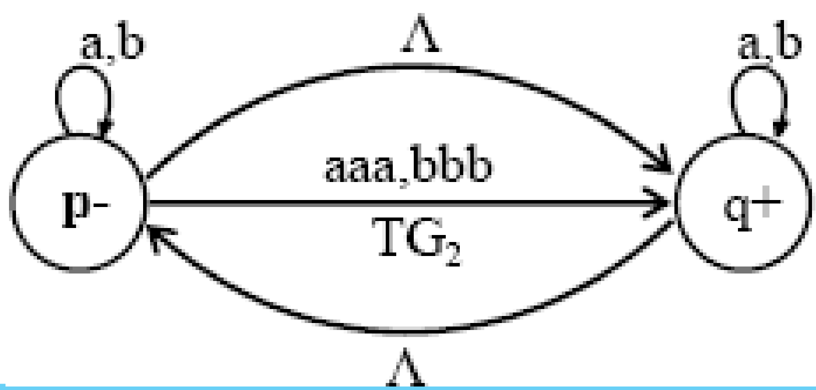
- Following may be a TG accepting $L_1 + L_2$



also a TG accepting L_1L_2 may be



and a TG accepting L_2*



Complements & Intersections

Complements

Definition:

- If L is a language over the alphabet , we define its

complement L´ to be the language of all strings of letters from **that are not words in L**.

Example:

– Let L be the language over the alphabet ∑ = {a, b} of all words that have a **double** a in them.

– Then, L^c or L´ is the language of all words that do **not** have a double a in them.

- Note that the complement of L´ is L. That is (L´) ´ = L or $(L^c)^c = L$

Theorem 11

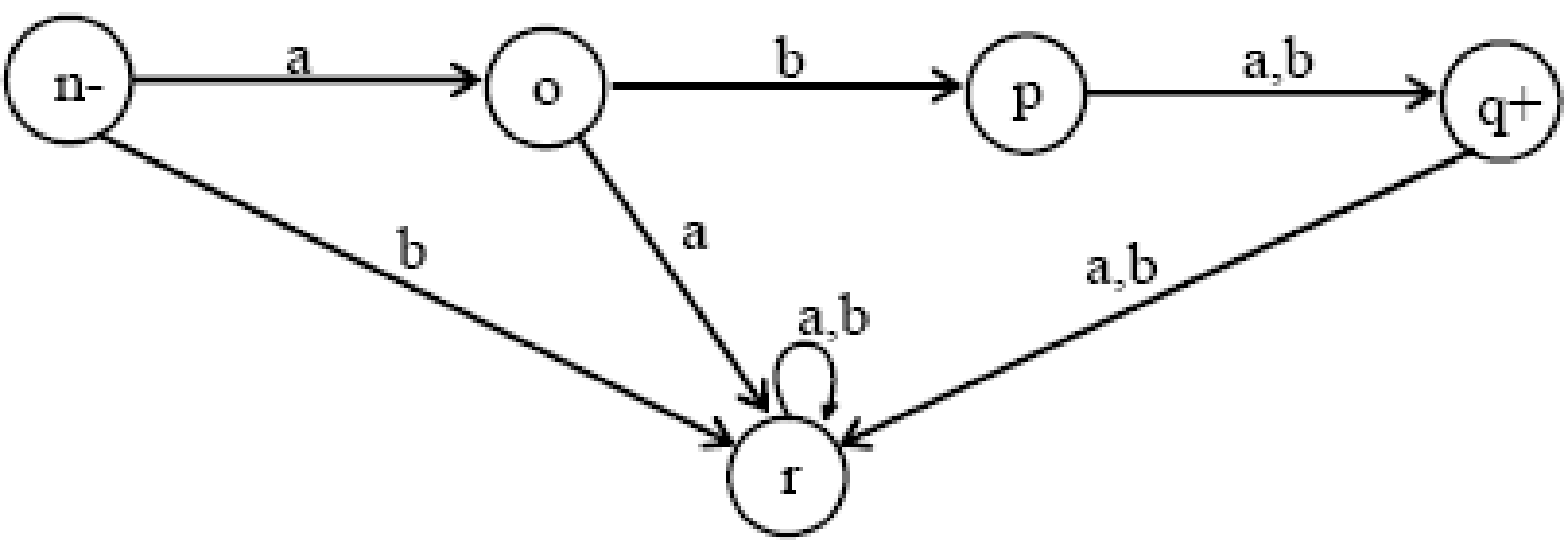
If L **is a regular language, then** L^c **is also a regular language. In other words, the set of regular languages is closed under complementation.**

Proof of theorem 11

- If L is a regular language, then by Kleene's theorem, there is some FA that accepts the language L.
- Some states of this FA are final states and some are not. Let us reverse the status of each state:
  - If it was a final state, make it a non-final state.
  - If it was a non-final state, make it a final state.
  - The start state gets reversed as follows: - ↔ ±
- If an input string formerly ended in a non-final state, it now ends in a final state, and vice versa.
- The new machine we have just built accepts all input strings that were not accepted by the original FA, and it rejects all the input strings that used to be accepted by FA.
- Hence using Kleene's theorem L^c can be expressed by some RE. Thus L^c is regular.

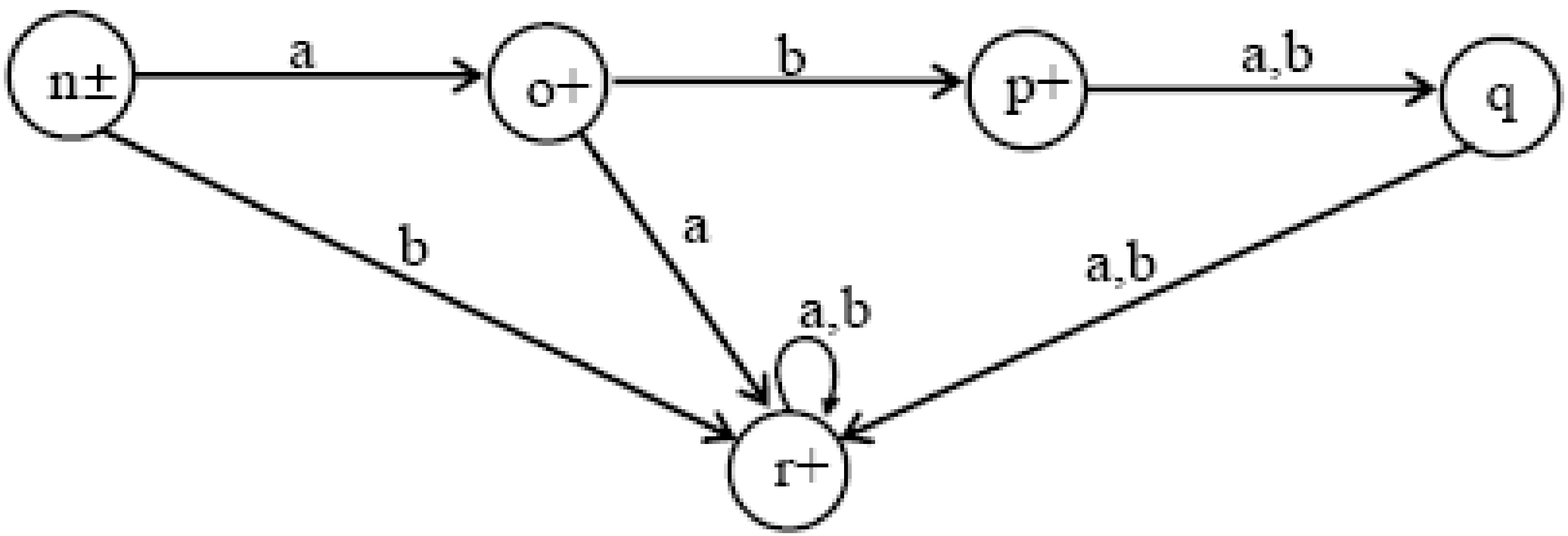
Example

- Let L be the language over the alphabet $\Sigma = \{a, b\}$, consisting of only two words aba and abb, then the FA accepting L may be



Example Cont…

- Converting final states to non-final states and old non-final states to final states, then FA accepting L^c may be



Intersection: Theorem 12

If L_1 **and** L_2 **are regular languages, then** $L_1 \cap L_2$ **is also a regular language. In other words, the set of regular languages is closed under intersection.**

Proof of Theorem 12

- By DeMorgan's law (for sets of any kind):
  $L_1 \cap L_2 = (L'_1 + L'_2)'$
- This means that the language $L_1 \cap L_2$ consists **of all words that are not in either L'_1 or L'_2.**
- Because L_1 and L_2 are regular, then so are L'_1 and L'_2 by Theorem 11.
- Since L'_1 and L'_2 are regular, so is $L'_1 + L'_2$ by Theorem 10.
- Now, since $L'_1 + L'_2$ is regular, so is $(L'_1 + L'_2)'$ by Theorem 11.
- This means L1 ∩ L2 is regular, because
  L1 ∩ L2 = $(L'_1 + L'_2)'$ by DeMorgan's law.

Example

- Consider two regular languages L_1 and L_2 , defined over the alphabet $\Sigma = \{a, b\}$, where

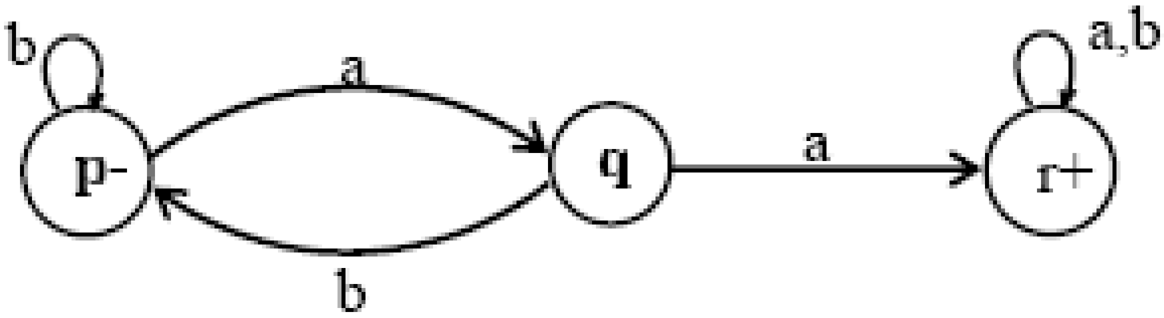
  L1 = language of words with **double a’s.**

  L2 = language of words containing **even number of a’s.**

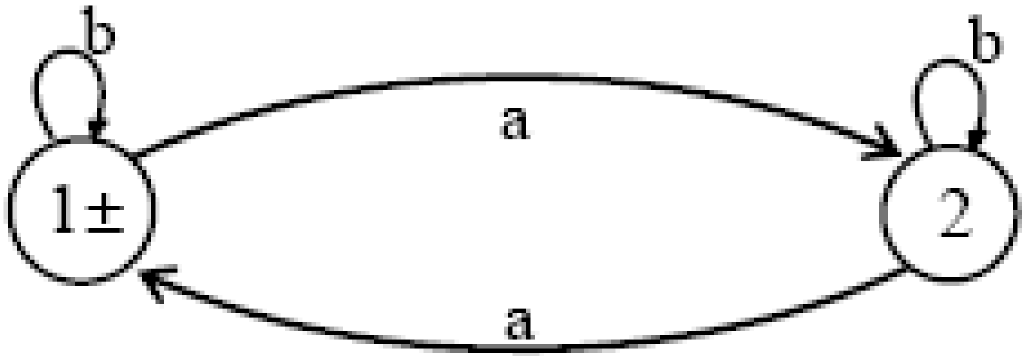
- FAs accepting languages L_1 and L_2 may be as follows

Example Cont….

- FA$_1$

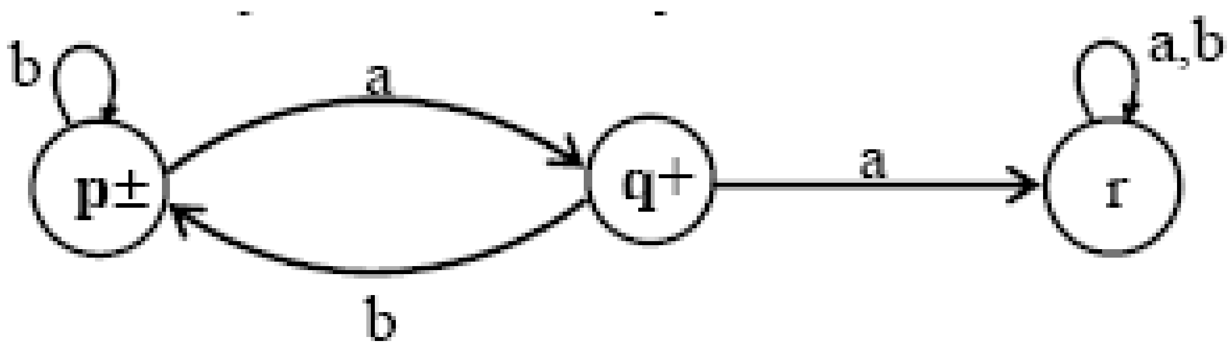


- FA$_2$

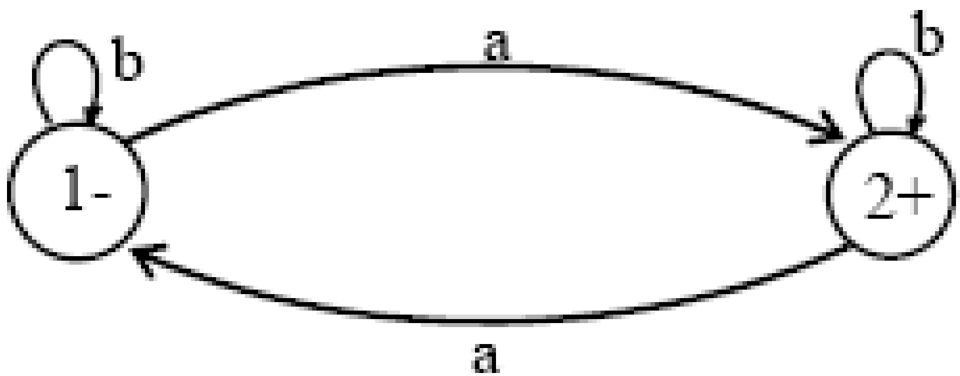


Example Cont….

- $FA_1{}^c$

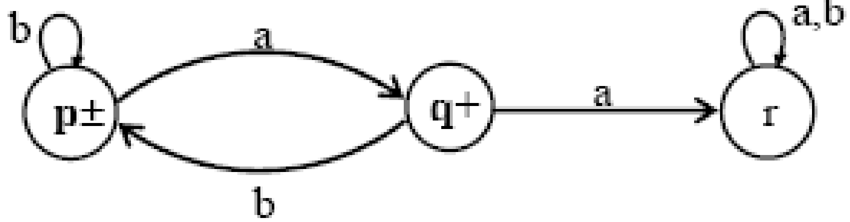


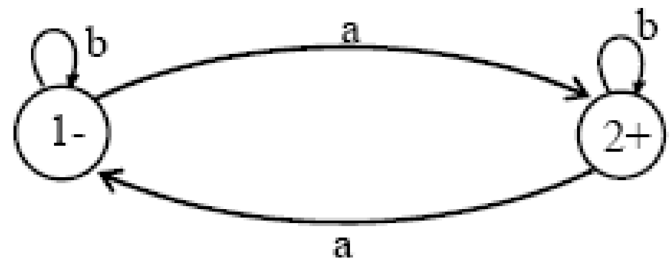
- $FA_2{}^c$



- Now FA accepting $L_1{}^c$ U $L_2{}^c$, using the method described earlier, may be as follows

Example Cont….



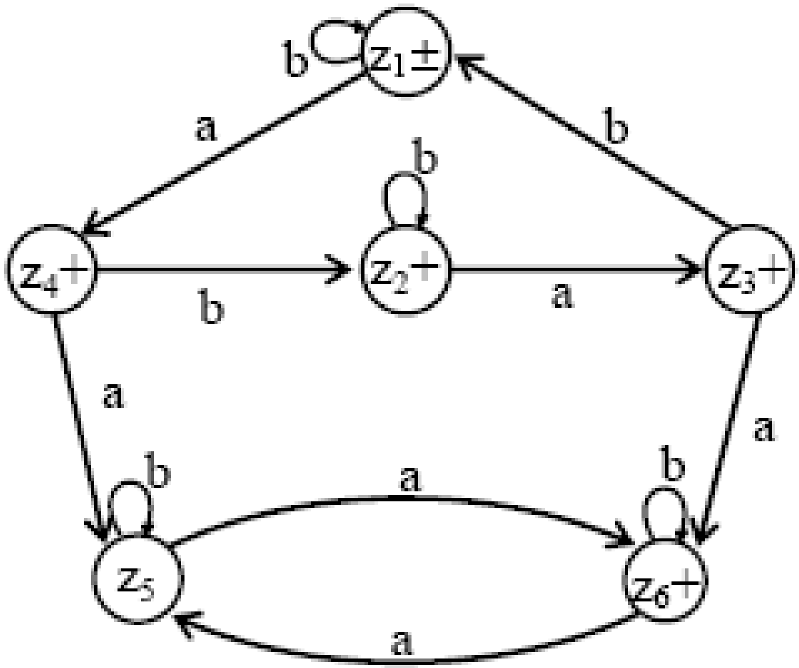


Old states	New states after reading	
	a	b
$z_1\pm \equiv (p, 1)$	$(q, 2) \equiv z_4$	$(p, 1) \equiv z_1$
$z_2 + \equiv (p, 2)$	$(q, 1) \equiv z_3$	$(p, 2) \equiv z_2$
$z_3 + \equiv (q, 1)$	$(r, 2) \equiv z_6$	$(p, 1) \equiv z_1$
$z_4 + \equiv (q, 2)$	$(r, 1) \equiv z_5$	$(p, 2) \equiv z_2$
$z_5 \equiv (r, 1)$	$(r, 2) \equiv z_6$	$(r, 1) \equiv z_5$
$z_6 + \equiv (r, 2)$	$(r, 1) \equiv z_5$	$(r, 2) \equiv z_6$

Example Cont….

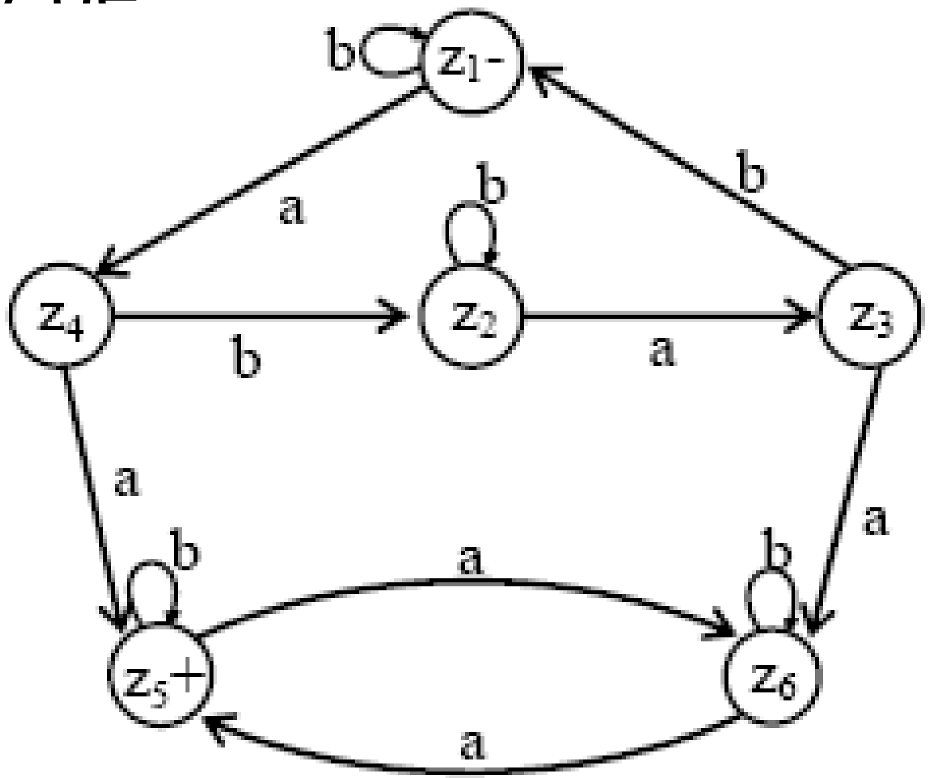
- Here all the possible combinations of states of $FA_1{}^c$ and $FA_2{}^c$ are considered

Old states	New states after reading	
	a	**b**
$z_1\pm \equiv (p, 1)$	$(q, 2) \equiv z_4$	$(p, 1) \equiv z_1$
$z_2 + \equiv (p, 2)$	$(q, 1) \equiv z_3$	$(p, 2) \equiv z_2$
$z_3 + \equiv (q, 1)$	$(r, 2) \equiv z_6$	$(p, 1) \equiv z_1$
$z_4 + \equiv (q, 2)$	$(r, 1) \equiv z_5$	$(p, 2) \equiv z_2$
$z_5 \equiv (r, 1)$	$(r, 2) \equiv z_6$	$(r, 1) \equiv z_5$
$z_6 + \equiv (r, 2)$	$(r, 1) \equiv z_5$	$(r, 2) \equiv z_6$



Example Cont….

- An FA that accepts the language $(L_1^c \cup L_2^c)^c = L_1 \cap L_2$ may be

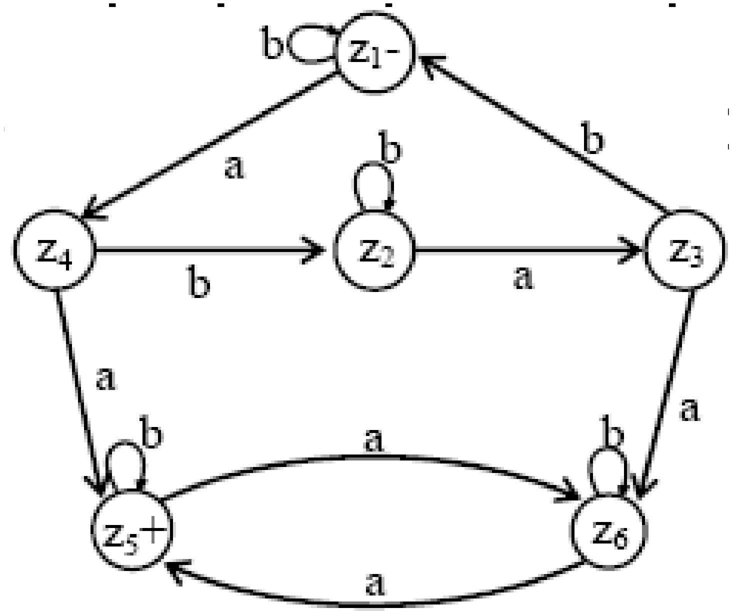


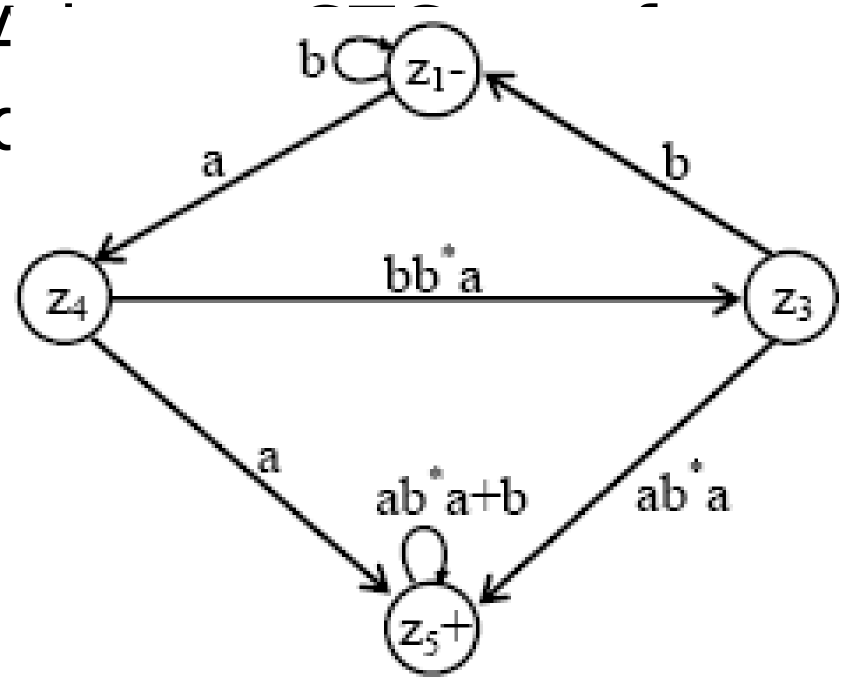
Example Cont….

- Corresponding **RE** can be determined as follows
- The regular expression defining the language $L_1 \cap L_2$ can be obtained, converting and

us F

$_2$ and



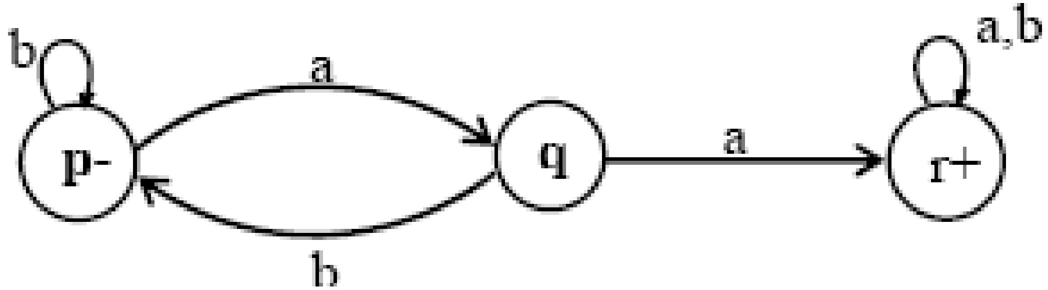


FA corresponding to intersection of two regular languages (short method)

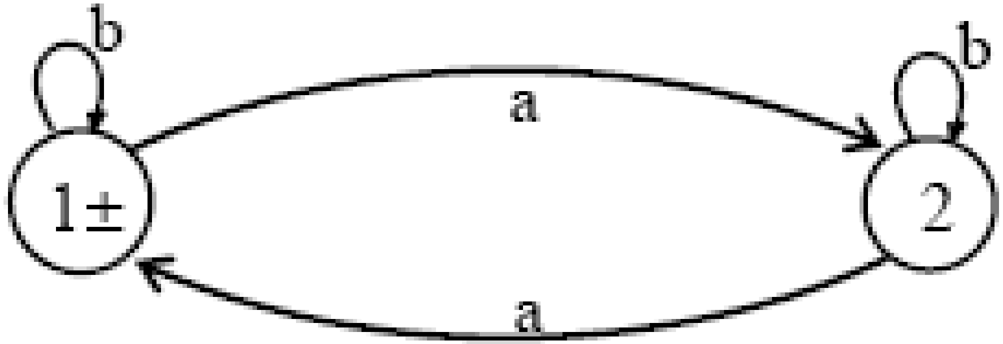
- Let FA_3 be an FA accepting $L_1 \cap L_2$, then **the initial state of FA_3 must correspond to the initial state of FA_1 and the initial state of FA_2.**
- Since the language corresponding to $L_1 \cap L_2$ is the intersection of corresponding languages L_1 and L_2, consists of the strings belonging to both L_1 and L_2, therefore **a final state of FA_3 must correspond to a final state of FA_1 and FA_2.**
- Following is an example regarding short method of finding an FA corresponding to the intersection of two regular languages.

Example

- Let r1 = (a+b)*aa(a+b)* and FA1 be

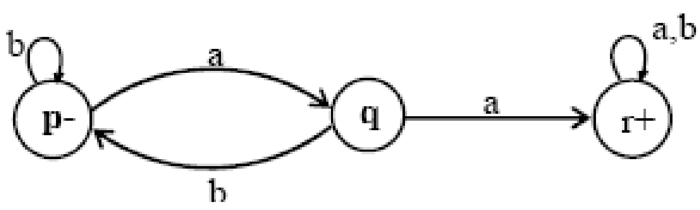


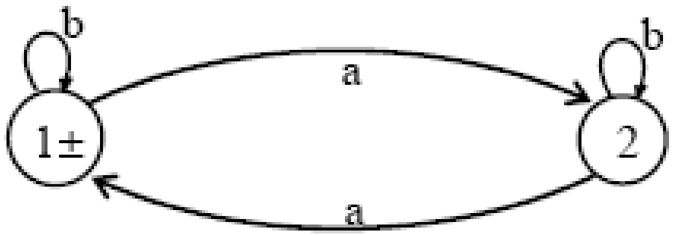
- also r2 = (b+ab*a)* and FA2 be



- An FA corresponding to $L_1 \cap L_2$ can be determined as follows

Example Cont….



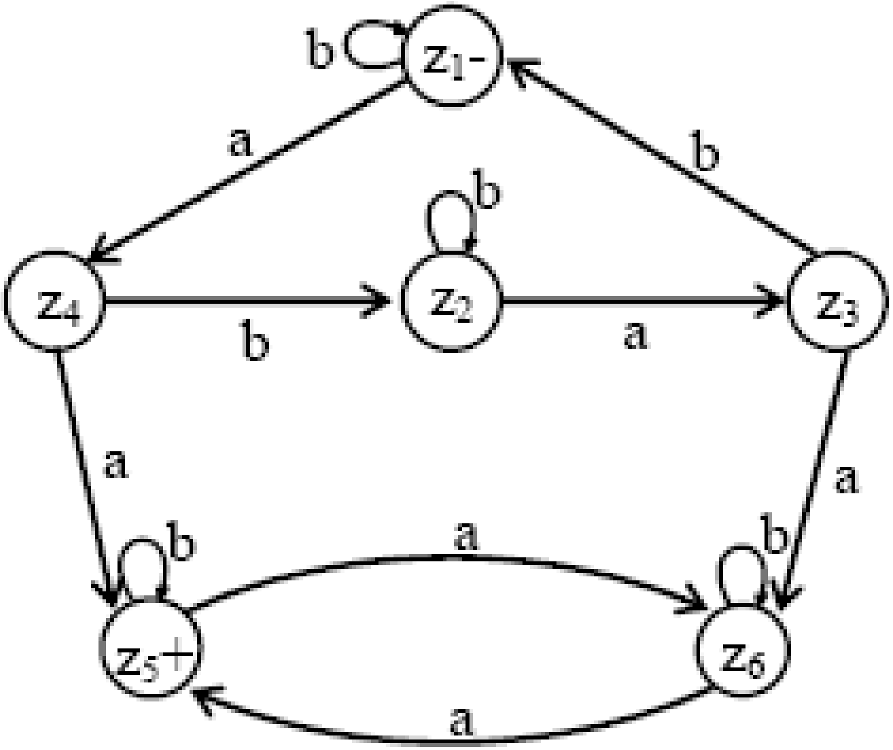


Old states	New states after reading	
	a	b
Z_1- ≡ (p, 1)	(q, 2) ≡ z_4	(p, 1) ≡ z_1
z_2 ≡ (p, 2)	(q, 1) ≡ z_3	(p, 2) ≡ z_2
z_3 ≡ (q, 1)	(r, 2) ≡ z_6	(p, 1) ≡ z_1
z_4 ≡ (q, 2)	(r, 1) ≡ z_5	(p, 2) ≡ z_2
z_5 + ≡ (r, 1)	(r, 2) ≡ z_6	(r, 1) ≡ z_5
z_6 ≡ (r, 2)	(r, 1) ≡ z_5	(r, 2) ≡ z_6

Example Cont….

- The corresponding transition diagram may be as follows:

Old states	New states after reading	
	a	b
Z_1- ≡ (p, 1)	(q, 2) ≡ z_4	(p, 1) ≡ z_1
z_2 ≡ (p, 2)	(q, 1) ≡ z_3	(p, 2) ≡ z_2
z_3 ≡ (q, 1)	(r, 2) ≡ z_6	(p, 1) ≡ z_1
z_4 ≡ (q, 2)	(r, 1) ≡ z_5	(p, 2) ≡ z_2
z_5 + ≡ (r, 1)	(r, 2) ≡ z_6	(r, 1) ≡ z_5
z_6 ≡ (r, 2)	(r, 1) ≡ z_5	(r, 2) ≡ z_6



For each of the following pairs of regular languages, find an FA that each define L1 ∩ L2 and convert into a Regular Expression

	L1	L2
1	(a+b)*a	b(a+b)*
2	(a+b)*a	(a+b)*aa(a+b)*
3	(a+b)*a	(a+b)*b
4	(a+b)b(a+b)*	b(a+b)*
5	(a+b)b(a+b)*	(a+b)*aa(a+b)*
6	(a+b)b(a+b)*	(a+b)*b
7	(b+ab)*b(a+Λ)	(a+b)*aa(a+b)*
8	(b+ab)*b(a+Λ)	(b+ab*a)*ab*
9	(b+ab)*b(a+Λ)	(a+ba)*a
10	(ab*)*	b(b+ab)*
11	(ab*)*	a(b+ab)*
12	(ab*)*	(a+b)*aa(a+b)*
13	(aa+ab+ba+bb)*	b(b+ab)*
14	(aa+ab+ba+bb)*	(a+b)*aa(a+b)*
15	(aa+ab+ba+bb)*	(b+ab)*a(a+Λ)
16	(aa+ab+ba+bb)*	a(a+b)*