Biçimsel Diller ve Otomata Teorisi

Sunu VII
Kleene Kuramı II

İZZET FATİH ŞENTÜRK

Kleene's Theorem

- Any language that can be defined by regular expression, or finite automaton, or transition graph can be defined by all three methods
- Proof
  - ✓Part 1: Every language that can be defined by a FA can also be defined by a TG
  - ✓Part 2: Every language that can be defined by a TG can also be defined by a RE
  - **Part 3: Every language that can be defined by a RE can also be defined by a FA**

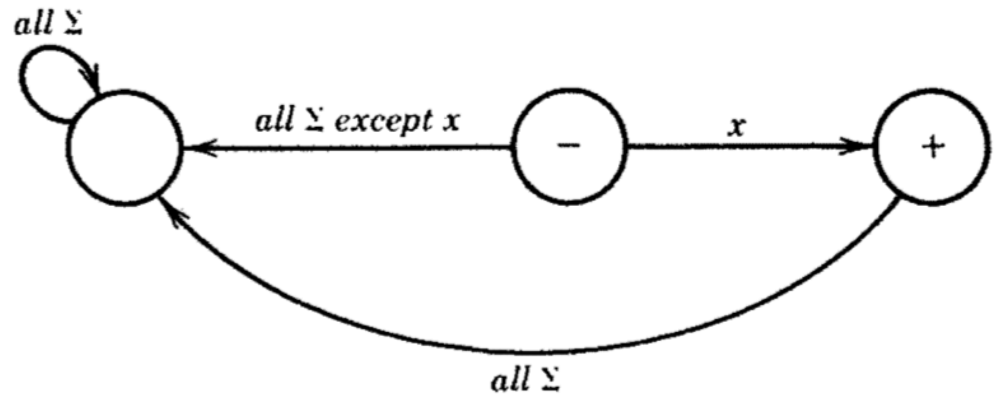
Proof, Part 3: Converting REs into FAs

- This is the hardest part of the whole theorem
- Every RE can be built up from the letters of the alphabet Σ and Λ by repeated application of certain rules:
  - Addition, concatenation, and closure
- When we build up a RE, we could <u>at the same time</u> be building up an FA that accepts the same language

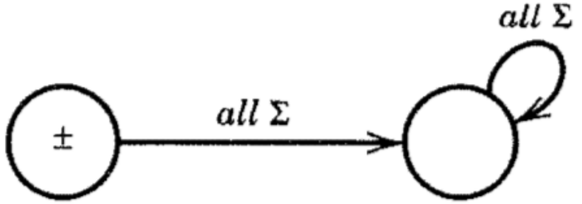
Proof of Part 3, Rule 1

- **Rule 1**: There is an FA that accepts any particular letter of the alphabet. There is an FA that accepts only the word Λ

- **Proof of Rule 1:** If x is in Σ, then the FA accepts only the word x



- **Proof of Rule 1:** One FA that accepts only Λ



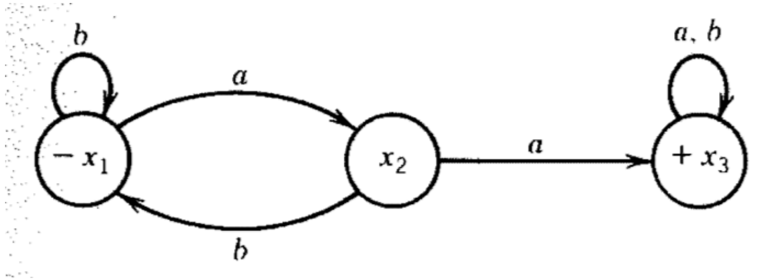
Proof of Part 3, Rule 2

- If there is an FA called FA_1 that accepts the language defined by the RE $\mathbf{r_1}$ and there is an FA called FA_2 that accepts the language defined by the RE $\mathbf{r_2}$, then there is an FA that we shall call FA_3, that accepts the language defined by the RE $(\mathbf{r_1} + \mathbf{r_2})$
  -> ! Union !

- We will prove Rule 2 by showing how to construct the new machine from the two old machines
- Before stating the general principals, we will demonstrate them in a specific example first

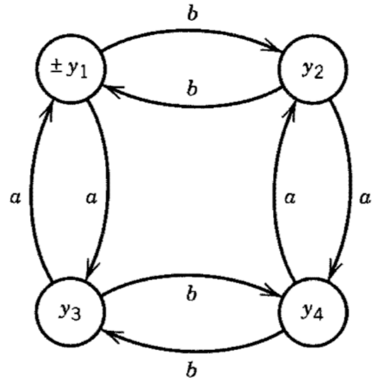
Proof of Part 3, Rule 2

- FA_1: The language of all words over $\Sigma=\{a\ b\}$ that have a double a somewhere in them



	a	b
$-x_1$	x_2	x_1
x_2	x_3	x_1
$+x_3$	x_3	x_3

- FA_2: EVEN-EVEN (even number of a's and an even number of b's)



	a	b
$\pm y_1$	y_3	y_2
y_2	y_4	y_1
y_3	y_1	y_4
y_4	y_2	y_3

Proof of Part 3, Rule 2

- FA_3: The language of all words that either have an aa or are in EVEN-EVEN and rejects all other strings with neither characteristics
  - The language of the new machine is the union of these two languages
  - We shall call the states in this new machine z_1, z_2, z_3 … for as many as needed
  - We shall define this machine by its transition table
  - We will keep track of where the input would be if it were running on FA_1 alone and where the input would be if it were running on FA_2 alone

Proof of Part 3, Rule 2

- First, we need a start state z_1
  - z_1 combines x_1 (if running on FA_1) and y_1 (if running on FA_2)
- All z-states in FA_3 machine carry with them a double meaning
  - It is running on both FA_1 and FA_2 and we keep track of both games simultaneously
- What new states can occur if the input letter a is read?

Proof of Part 3, Rule 2

- What new states can occur if the input letter a is read?
  - For FA_1, it would put the machine into state x_2
  - For FA_2, it would put the machine into state y_3
- On FA_3, letter a puts the machine into state z_2 which means either x_2 or y_3

	a	b
$-x_1$	x_2	x_1
x_2	x_3	x_1
$+x_3$	x_3	x_3

	a	b
$\pm y_1$	y_3	y_2
y_2	y_4	y_1
y_3	y_1	y_4
y_4	y_2	y_3

$$\pm z_1 = x_1 \quad \text{or} \quad y_1$$
$$z_2 = x_2 \quad \text{or} \quad y_3$$

Proof of Part 3, Rule 2

- If we are in z_1 and read the letter b
  - For FA_1, it would put the machine into state x_1 (from state x_1)
  - For FA_2, it would put the machine into state y_2 (from state y_1)

	a	b
$-x_1$	x_2	x_1
x_2	x_3	x_1
$+x_3$	x_3	x_3

	a	b
$\pm y_1$	y_3	y_2
y_2	y_4	y_1
y_3	y_1	y_4
y_4	y_2	y_3

$$\pm z_1 = x_1 \quad \text{or} \quad y_1$$
$$z_2 = x_2 \quad \text{or} \quad y_3$$
$$z_3 = x_1 \quad \text{or} \quad y_2$$

- The beginning of the transition table for FA_3

	a	b
$\pm z_1$	z_2	z_3

Proof of Part 3, Rule 2

- If we are in z_2 and read the letter a
  - For FA_1, it would put the machine into state x_3 (final state)
  - For FA_2, it would put the machine into state y_1
- If we are in z_2 and read the letter b
  - For FA_1, it would put the machine into state x_1
  - For FA_2, it would put the machine into state y_4

	a	b
$-x_1$	x_2	x_1
x_2	x_3	x_1
$+x_3$	x_3	x_3

	a	b
$\pm y_1$	y_3	y_2
y_2	y_4	y_1
y_3	y_1	y_4
y_4	y_2	y_3

$+z_4 = x_3$ or y_1

$z_5 = x_1$ or y_4

	a	b
$\pm z_1$	z_2	z_3
z_2	z_4	z_5

- Acceptance by either machine FA_1 or FA_2 is enough for acceptance by FA_3

Proof of Part 3, Rule 2

- If we are in z_3 and read the letter a
  - For FA_1, it would put the machine into state x_2
  - For FA_2, it would put the machine into state y_4
- If we are in z_3 and read the letter b
  - For FA_1, it would put the machine into state x_1
  - For FA_2, it would put the machine into state y_1

	a	b
$-x_1$	x_2	x_1
x_2	x_3	x_1
$+x_3$	x_3	x_3

	a	b
$\pm y_1$	y_3	y_2
y_2	y_4	y_1
y_3	y_1	y_4
y_4	y_2	y_3

$z_6 = x_2$ or y_4

	a	b
$\pm z_1$	z_2	z_3
z_2	z_4	z_5
z_3	z_6	z_1

Proof of Part 3, Rule 2

- If we are in z_4 and read the letter a
  - For FA_1, it would put the machine into state x_3 (final state)
  - For FA_2, it would put the machine into state y_3
- If we are in z_4 and read the letter b
  - For FA_1, it would put the machine into state x_3 (final state)
  - For FA_2, it would put the machine into state y_2

	a	b
$-x_1$	x_2	x_1
x_2	x_3	x_1
$+x_3$	x_3	x_3

	a	b
$\pm y_1$	y_3	y_2
y_2	y_4	y_1
y_3	y_1	y_4
y_4	y_2	y_3

$+z_7 = x_3$ or y_3

$+z_8 = x_3$ or y_2

	a	b
$\pm z_1$	z_2	z_3
z_2	z_4	z_5
z_3	z_6	z_1

Proof of Part 3, Rule 2

If we are in z_5 and we read an a, we go to x_2 or y_2, which we shall call z_9.
If we are in z_5 and we read a b, we go to x_1 or y_3, which we shall call z_{10}.

$$z_9 = x_2 \quad \text{or} \quad y_2$$
$$z_{10} = x_1 \quad \text{or} \quad y_3$$

If we are in z_6 and we read an a, we go to x_3 or y_2, which is our old z_8.
If we are in z_6 and we read a b, we go to x_1 or y_3, which is z_{10} again.
If we are in z_7 and we read an a, we go to x_3 or y_1, which is z_4 again.
If we are in z_7 and we read a b, we go to x_3 or y_4, which is a new state, z_{11}.

$$+z_{11} = x_3 \quad \text{or} \quad y_4$$

If we are in z_8 and we read an a, we go to x_3 or $y_4 = z_{11}$.
If we are in z_8 and we read a b, we go to x_3 or $y_1 = z_4$.
If we are in z_9 and we read an a, we go to x_3 or $y_4 = z_{11}$.
If we are in z_9 and we read a b, we go to x_1 or $y_1 = z_1$.
If we are in z_{10} and we read an a, we go to x_2 or y_1, which is our last new state, z_{12}.

$$+z_{12} = x_2 \quad \text{or} \quad y_1$$

If we are in z_{10} and we read a b, we go to x_1 or $y_4 = z_5$.

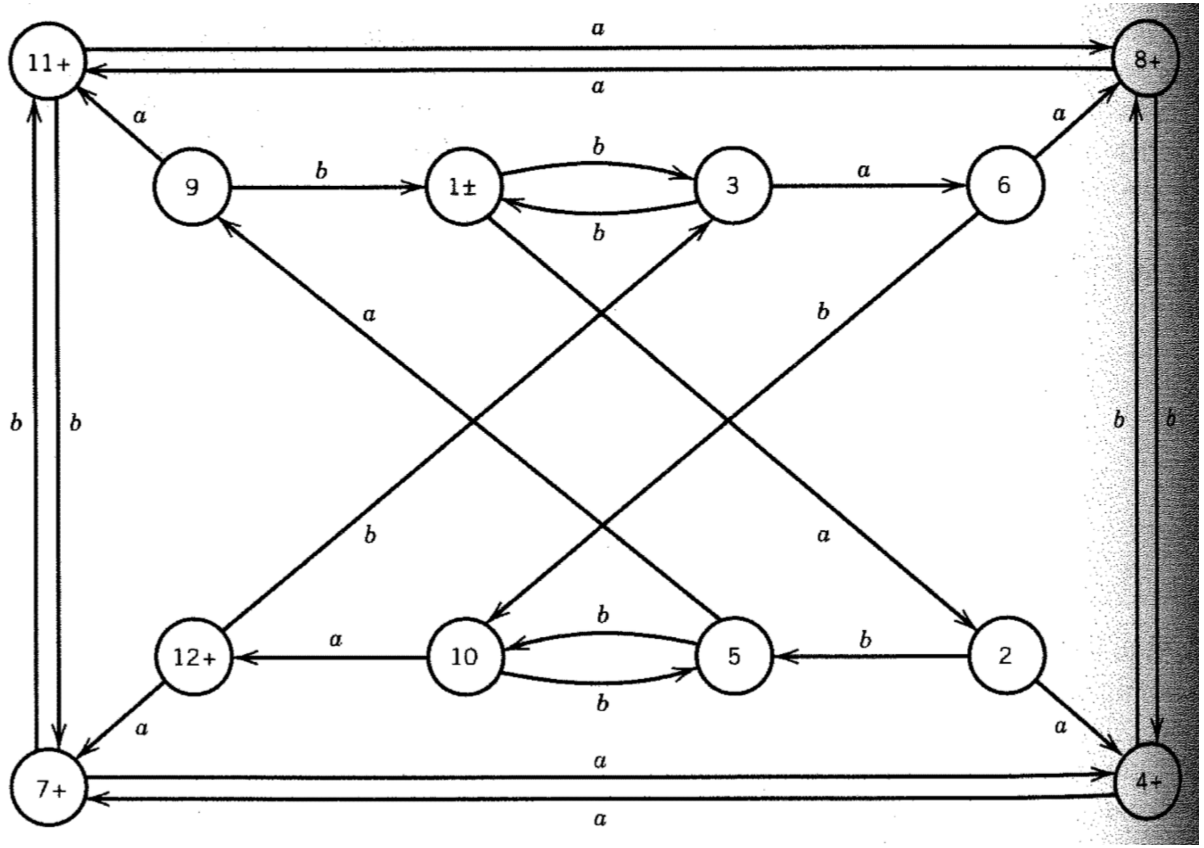
If we are in z_{11} and we read an a, we go to x_3 or $y_2 = z_8$.
If we are in z_{11} and we read a b, we go to x_3 or $y_3 = z_7$.
If we are in z_{12} and we read an a, we go to x_3 or $y_3 = z_7$.
If we are in z_{12} and we read a b, we go to x_1 or $y_2 = z_3$.

	a	b
$\pm z_1$	z_2	z_3
z_2	z_4	z_5
z_3	z_6	z_1
$+z_4$	z_7	z_8
z_5	z_9	z_{10}
z_6	z_8	z_{10}
$+z_7$	z_4	z_{11}
$+z_8$	z_{11}	z_4
z_9	z_{11}	z_1
z_{10}	z_{12}	z_5
$+z_{11}$	z_8	z_7
$+z_{12}$	z_7	z_3

Proof of Part 3, Rule 2

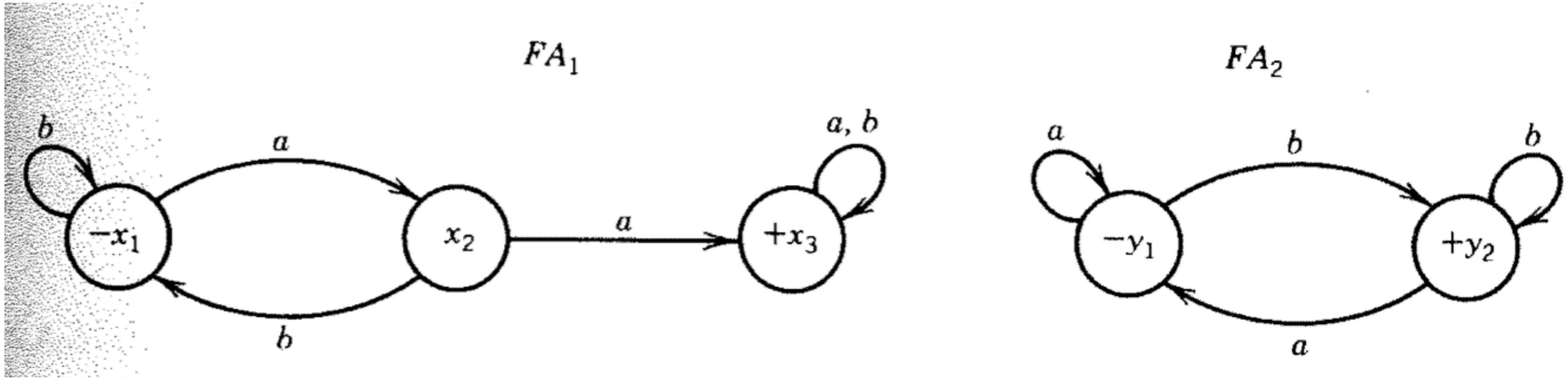
- FA_3

	a	b
$\pm z_1$	z_2	z_3
z_2	z_4	z_5
z_3	z_6	z_1
$+z_4$	z_7	z_8
z_5	z_9	z_{10}
z_6	z_8	z_{10}
$+z_7$	z_4	z_{11}
$+z_8$	z_{11}	z_4
z_9	z_{11}	z_1
z_{10}	z_{12}	z_5
$+z_{11}$	z_8	z_7
$+z_{12}$	z_7	z_3

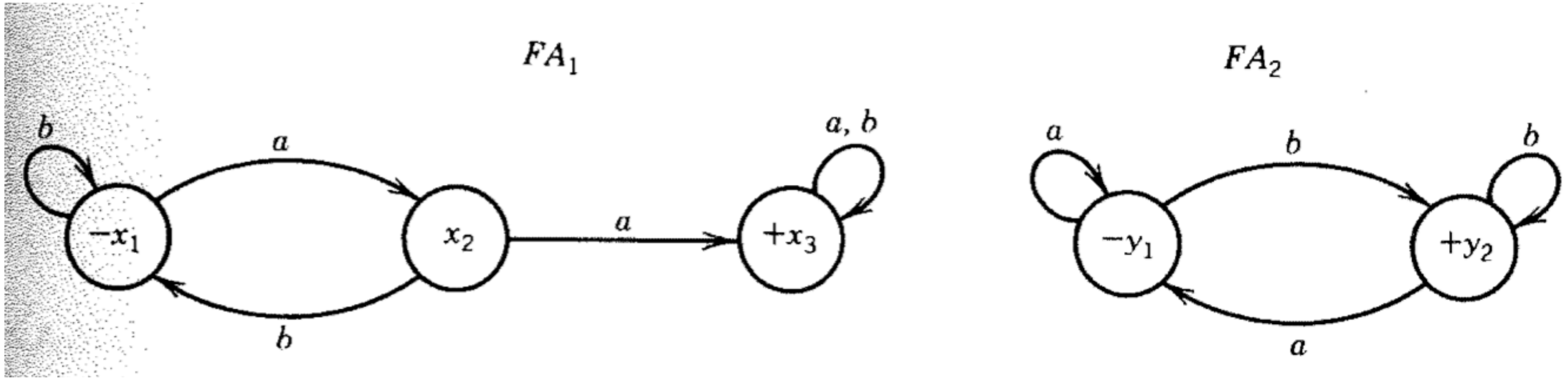


Example

- FA_1 accepts all words with a double a in them
- FA_2 accepts all words ending in b

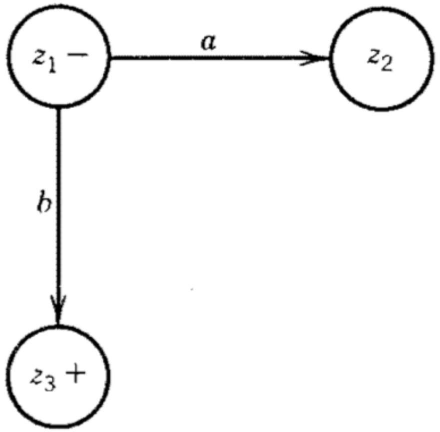


Example

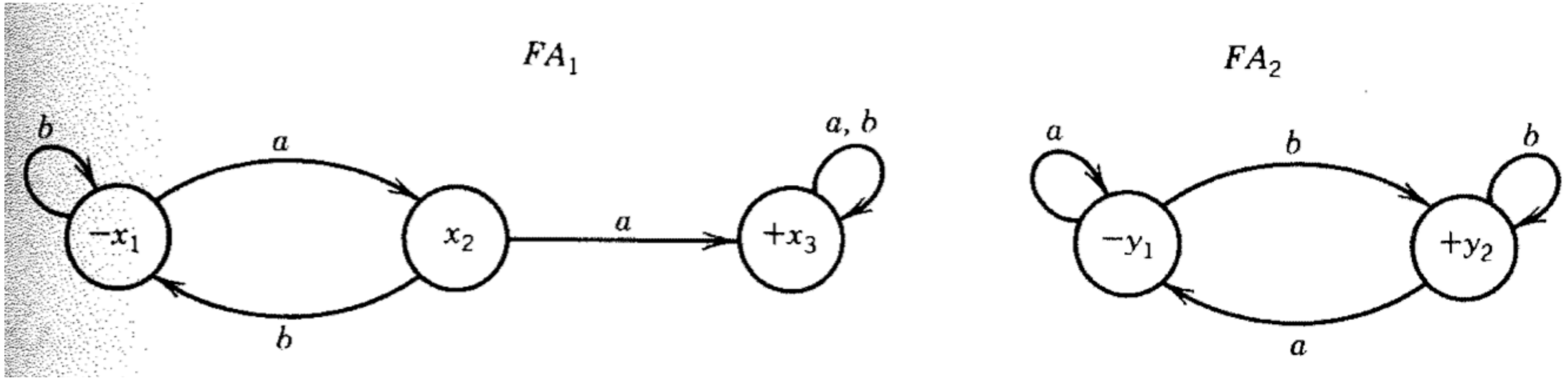


$-z_1 = x_1$ or y_1

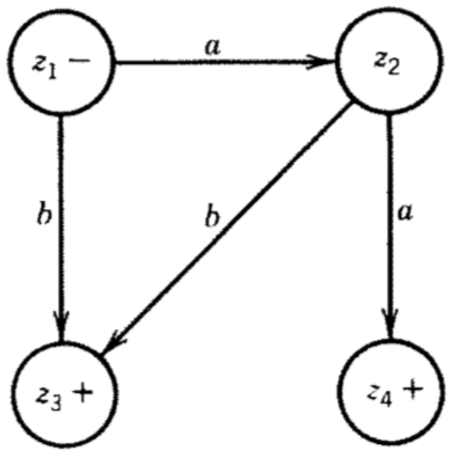
In z_1 if we read an a, we go to x_2 or $y_1 = z_2$
In z_1 if we read a b, we go to x_1 or $y_2 = z_3$, which is a final state since y_2 is.



Example

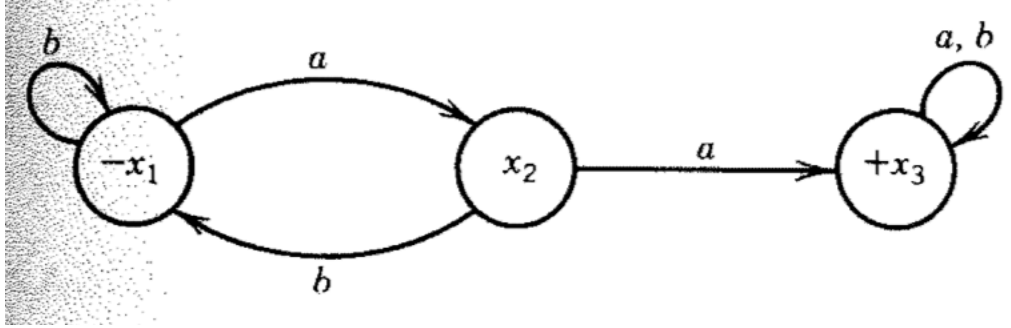


In z_2 if we read an a, we go to x_3 or $y_1 = z_4$, which is a final state because x_3 is.
In z_2 if we read a b, we go to x_1 or $y_2 = z_3$.

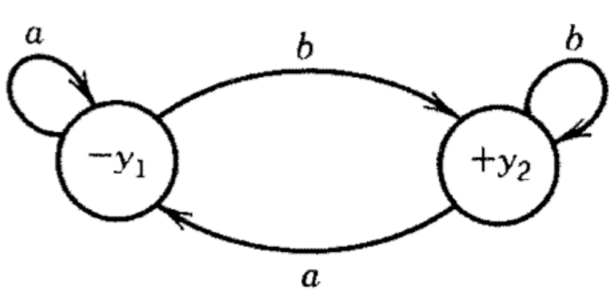


Example

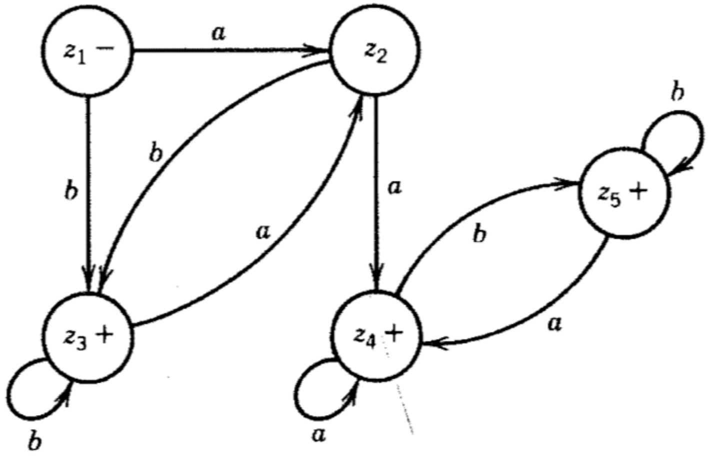
FA_1



FA_2



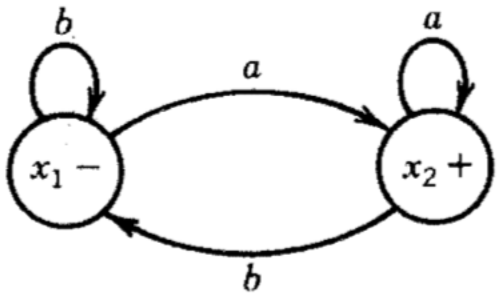
In z_3 if we read an a, we go to x_2 or $y_1 = z_2$.
In z_3 if we read a b, we go to x_1 or $y_2 = z_3$.
In z_4 if we read an a, we go to x_3 or $y_1 = z_4$.
In z_4 if we read a b, we go to x_3 or $y_2 = z_5$, which is a final state.
In z_5 if we read an a, we go to x_3 or $y_1 = z_4$.
In z_5 if we read a b, we go to x_3 or $y_2 = z_5$.

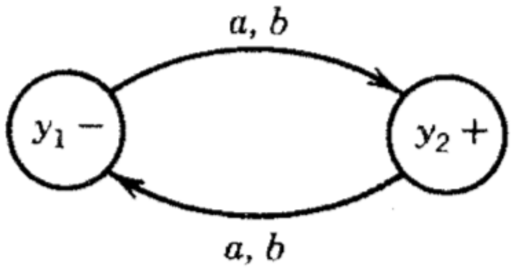


- $z_6 = x_2$ or y_2 does not arise, why?

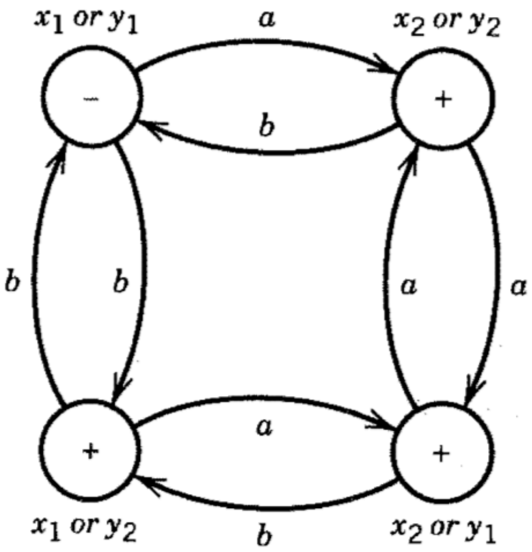
Example

- FA_1 accepts all words that end in a
- FA_2 accepts all words with an odd number of letters



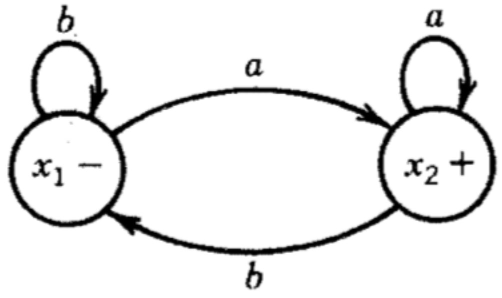


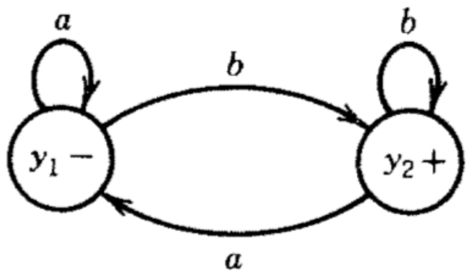
- FA_3 accepts all words that either have an odd number of letters or end in a
  - The only state that is not a + state is the - state



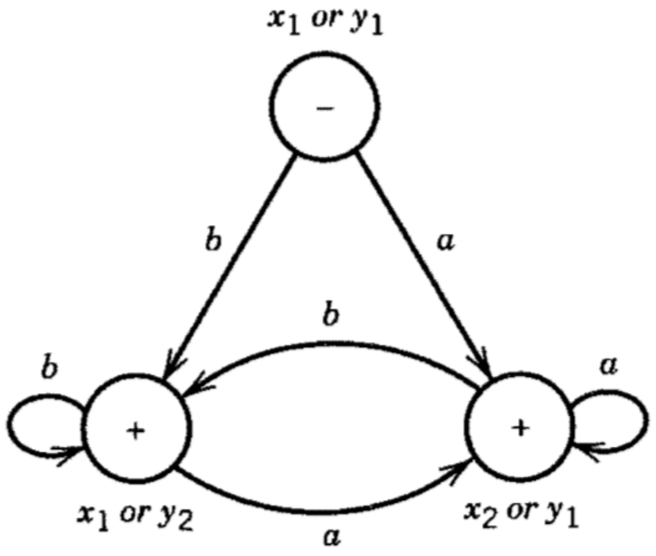
Example

- FA_1 accepts all words that end in a
- FA_2 accepts all words that end in b





- FA_3 accepts all words that end in a or b (all words except Λ)
  - State x_2 or y_2 cannot be reached

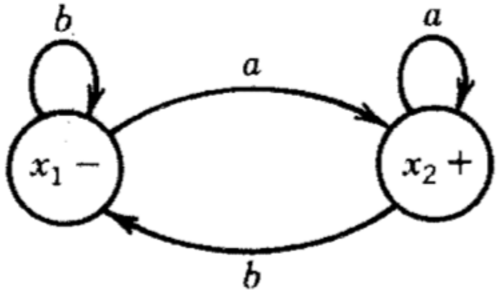


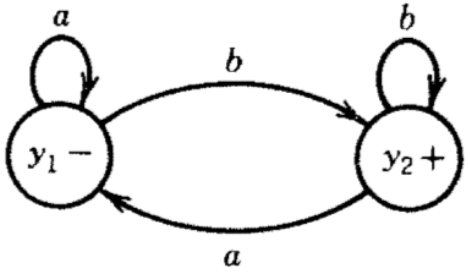
An Alternate Procedure for Producing the Union-Machine

- Let FA_1 have states x_1, x_2, …
- Let FA_2 have states y_1, y_2, …
- We can define the union machine (FA_3) initially as having all the possible states x_1 or y_1 for all combinations of i and j
- The number of states in FA_3 would always be the product of the number of states in FA_1 and FA_2
- For each state in FA_3 we could draw it's a-edge and b-edge in any order
- What we have done before is create new z states when needed

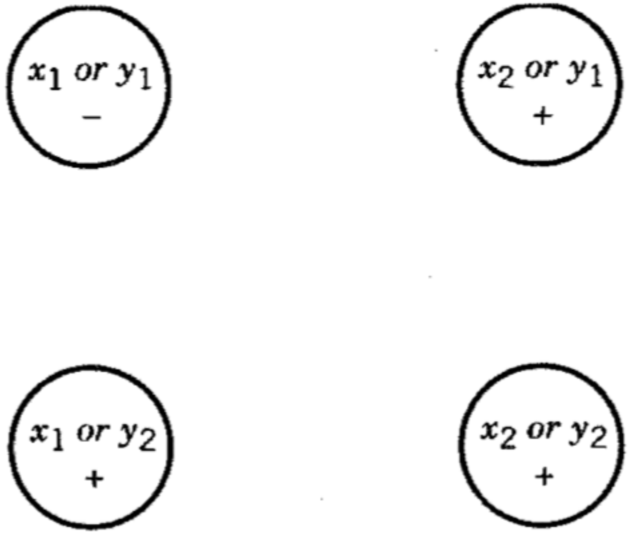
An Alternate Procedure for Producing the Union-Machine

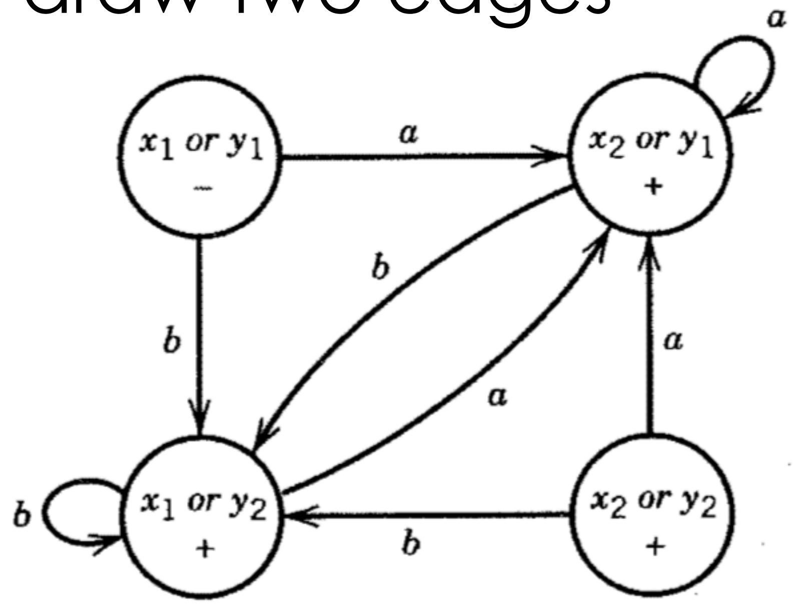
- We could start with four possible states for the example before





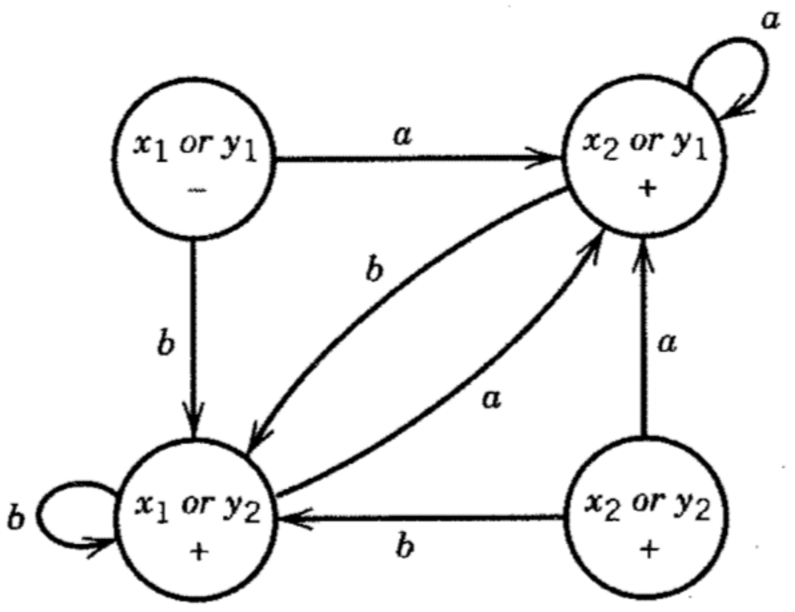
- For each of these four states we would draw two edges





An Alternate Procedure for Producing the Union-Machine

- This is a perfectly possible FA for the union language $FA_1 + FA_2$
- However, we see that its lower right-hand side state is completely useless
  - It can never be entered by any string starting at –
  - It is not against the definition of an FA to have a useless state

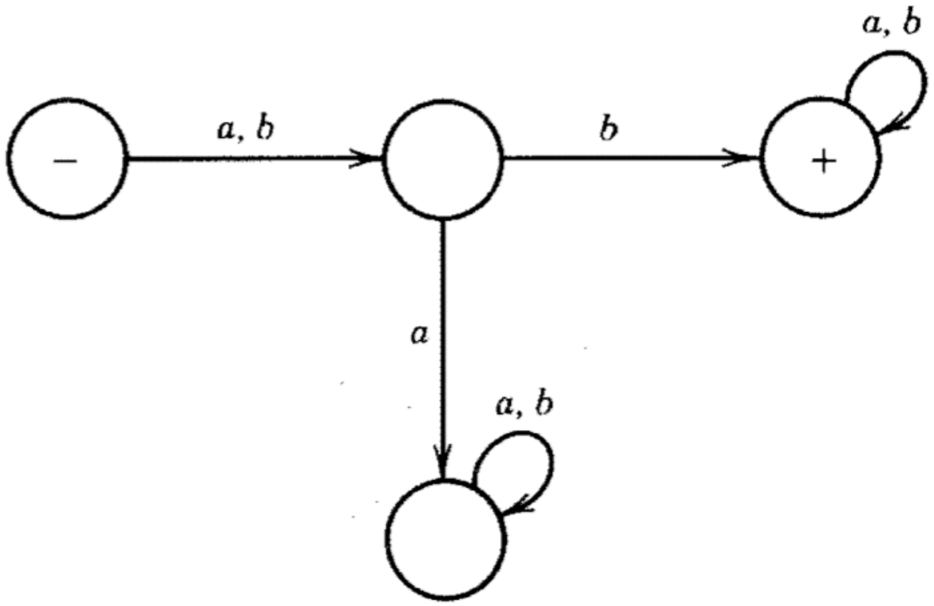


Proof of Part 3, Rule 3

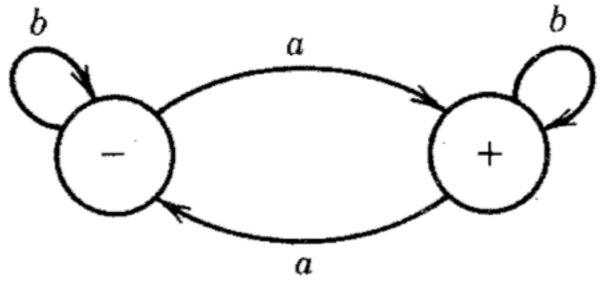
- Rule 3
  - If there is an FA_1 that accepts the language defined by the regular expression $\mathbf{r}_1$ and an FA_2 that accepts the language defined by the regular expression $\mathbf{r}_2$, then there is an FA_3 that accepts the language defined by the concatenation $\mathbf{r}_1\mathbf{r}_2$, the product language

Example

- L1: The language of all words with b as the second letter



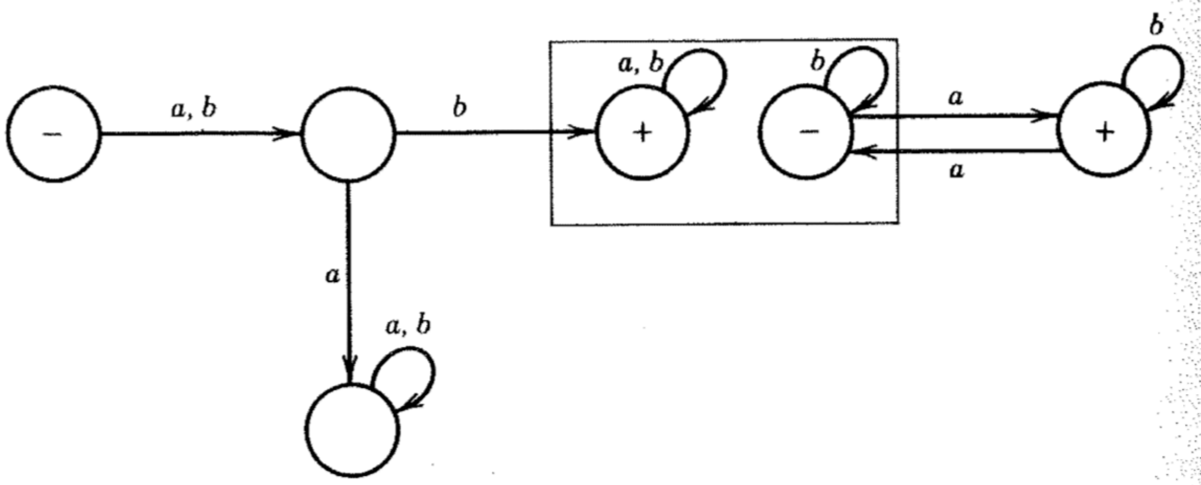
- L2: The language of all words that have an odd number of a's



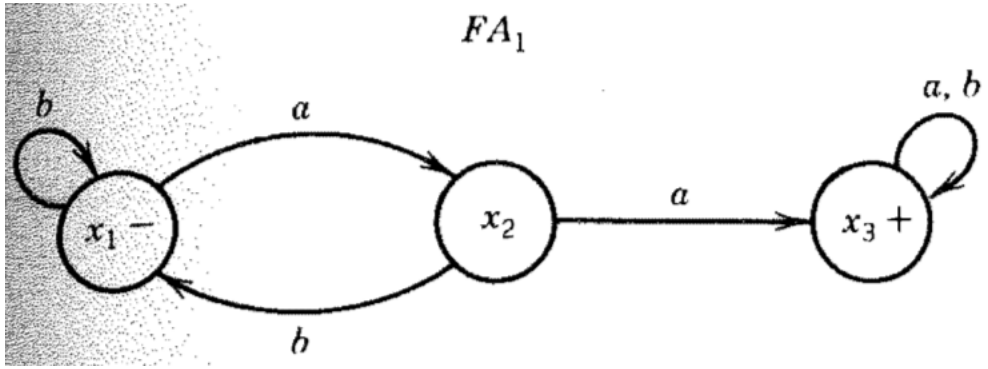
- Consider the input string (ab)(abbaa)
  - Begin on FA1 and finish on +
  - Jump to FA with the remaining string and finish on +

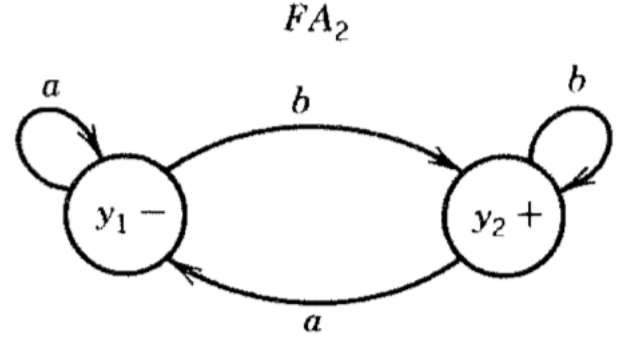
Example

- This simple idea does not work
- Consider a different input string for the same product language: ababbab
- (abab)(bab) is accepted
- (ab)(abbab) is rejected
- How do we know when to jump?

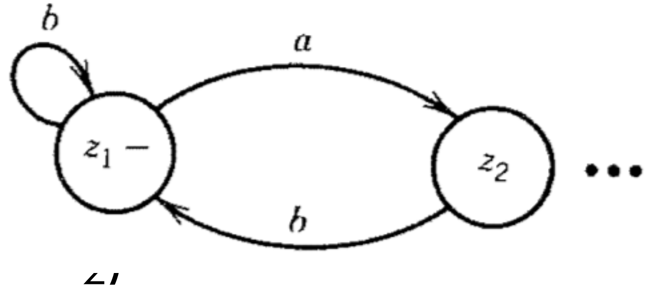


Example





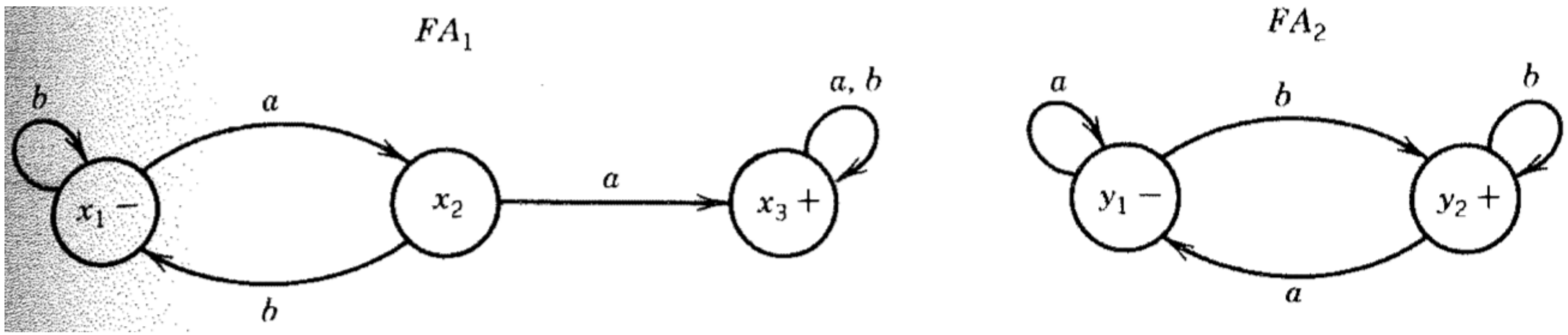
- Start with the state z_1, which is exactly like x_1
  - The input string is being run on FA_1 alone
  - From z_1, if read a *b*, we must return to x_1
  - From z_1, if read an *a*, we must go to x_2 (z_2 is same
  - From z_2, if read an *a*, we must go to z_3 (z_3 is same as x_3)



- X3 has a dual identity
  - Either it means that we have reached the final state in FA1
  - Or else we pass through

$$z_3 = \begin{cases} x_3, \text{ and we are still running on } FA_1 \\ \text{or} \\ y_1, \text{ and we have begun to run on } FA_2 \end{cases}$$

Example



- We are in z_3 and we read an *a*, we have three options

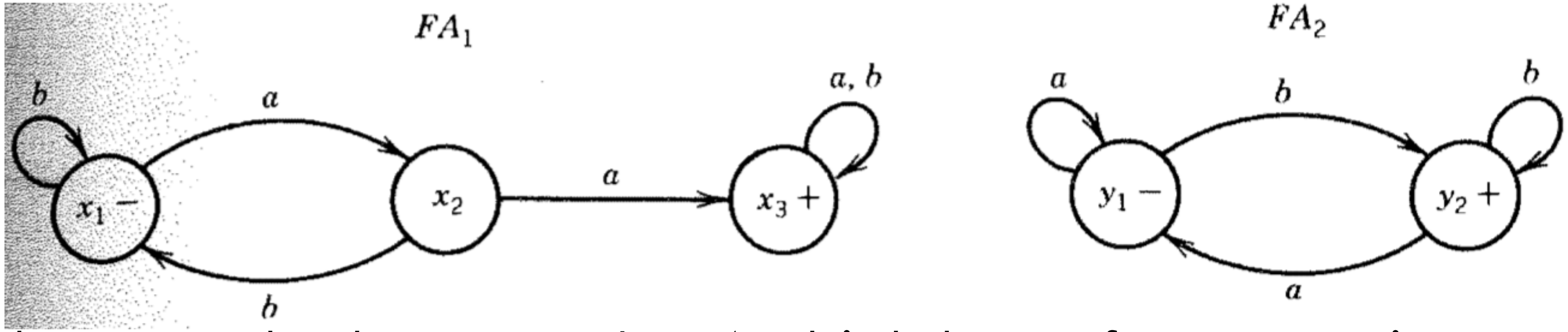
$$\begin{cases} \text{We are back in } x_3 \text{ continuing to run the string on } FA_1 \\ \text{or} \\ \text{we have just finished on } FA_1 \text{ and we are now in } y_1 \\ \text{beginning to run on } FA_2 \\ \text{or} \\ \text{we have looped from } y_1 \text{ back to } y_1 \text{ while already running on} \\ FA_2 \end{cases}$$

$= x_3$ or y_1
(because being in y_1 is the same whether we are there for the first time or not)

$= z_3$

- Reading an *a* takes us back to z_3 from z_3

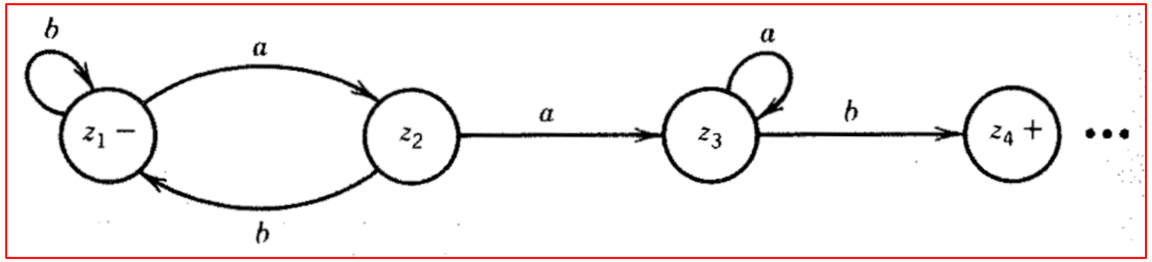
Example



- We are in z_3 and we read a *b*, we go to z4 which have four meanings

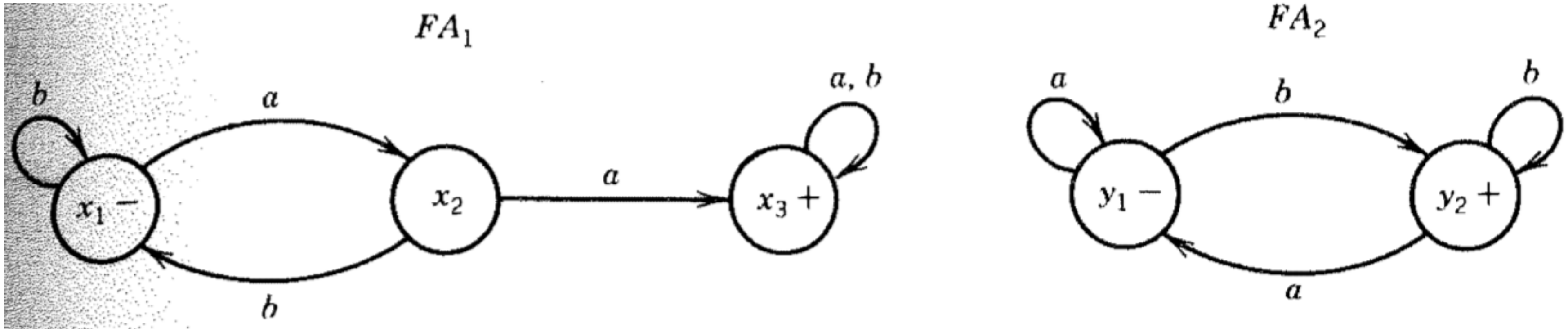
$$+z_4 = \begin{cases} \text{We are still in } x_3 \text{ continuing to run on } FA_1 \\ \text{or} \\ \text{we have just finished running on } FA_1 \text{ and are now in } y_1 \text{ on } FA_2 \\ \text{or} \\ \text{we are now in } y_2 \text{ on } FA_2\text{, having reached there via } y_1 \end{cases}$$

$$= \quad x_3 \quad \text{or} \quad y_1 \quad \text{or} \quad y_2$$



- If a path ends in z_4, this path can be broken into two parts:
  - The first part: From x_1 to x_3
  - The second part: From y_1 to y_2
  - Therefore, it must be accepted. z_4 is a final state

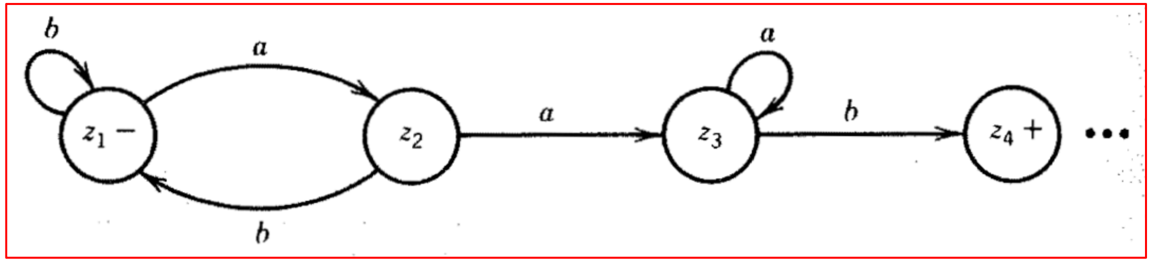
Example



- We are in z_4 and we read an *a*, our choices are:

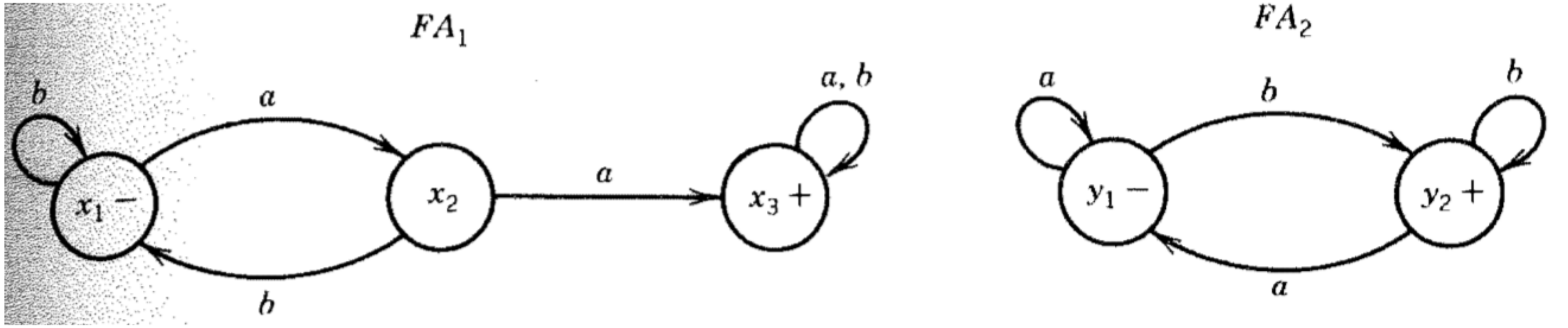
$$= \begin{cases} \text{remaining in } x_3 \text{ and continuing to run on } FA_1 \\ \text{or} \\ \text{having just finished } FA_1 \text{ and beginning at } y_1 \\ \text{or} \\ \text{having moved from } y_2 \text{ back to } y_1 \text{ in } FA_2 \end{cases}$$

$= x_3 \quad \text{or} \quad y_1$



- This is exactly the definition of z_3
    - If we are in z_4 and read an *a*, we go back to z_3

Example



- We are in z_4 and we read a *b*, our choices are:

remaining in x_3 and continuing to run on FA_1

or

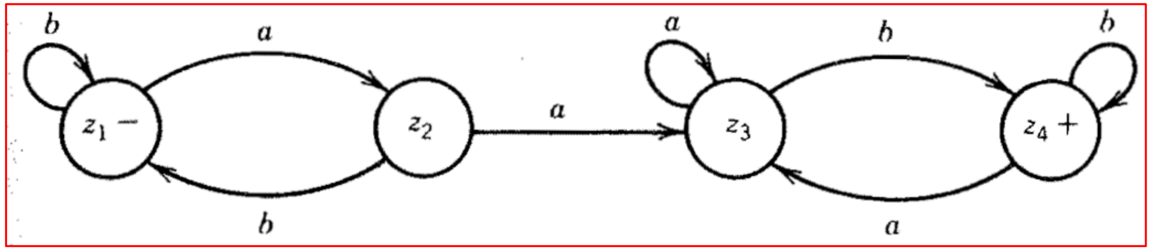
having just finished FA_1 and beginning at y_1

or

having looped back from y_2 to y_2 running on FA_2

$= x_3 \quad \text{or} \quad y_1 \quad \text{or} \quad y_2$

$= z_4$



- This is the definition of z_4
  - If we are in z_4 and read a *b*, we go loop back to z_4

Proof of Part 3, Rule 4

- Rule 4
  - If **r** is a regular expression and FA_1 is a finite automaton that accepts exactly the language defined by **r**, then there is an FA called FA_2 that will accept exactly the language defined by **r***