

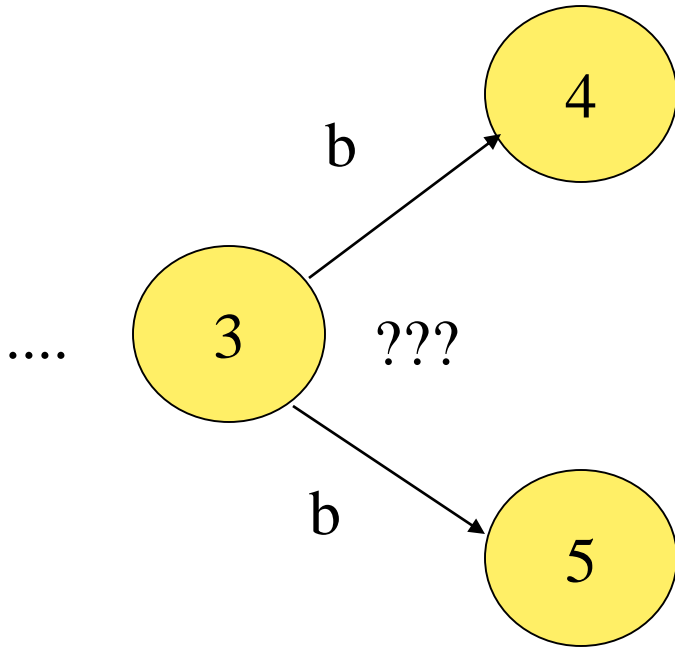


# Deterministik Olmayan Sonlu Otomata (Non-Deterministic Finite Automata)

Otomata Teorisi  
Konya Teknik Üniversitesi  
Bilgisayar Mühendisliği Bölümü



# "Non-Deterministic" Kavramı



“Path” belirlemesi sadece girdiye bağlı değildir. Makine hangi yolu izleyeceğine bir şekilde karar vermek zorundadır. Burada; insan faktörü veya yapay zeka teknikleri ile makineye karar verdirme gibi durumlar düşünülebilir.

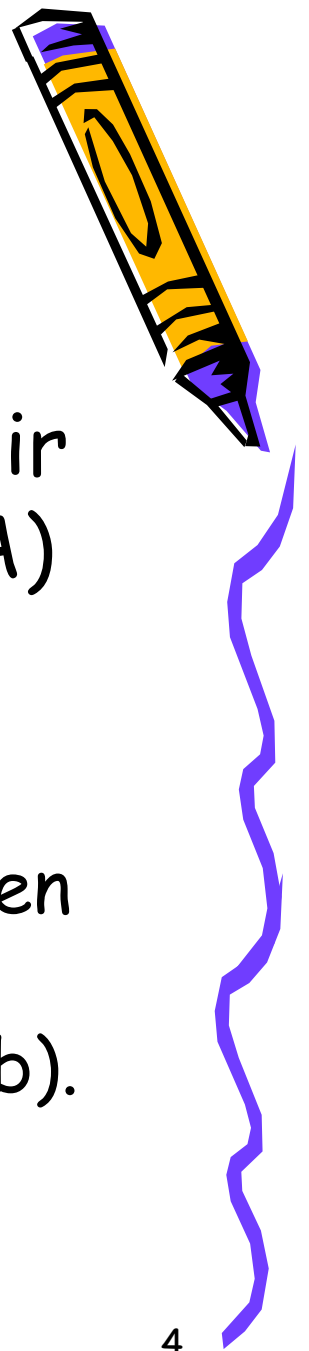
# Tanım



- Bir non-deterministik sonlu otomata (NFA) 3 bileşenden oluşmaktadır:
  - Bir tanesi başlangıç, bir veya birden fazlası sonuç olabilen durumlar kümesi.
  - Girdi alfabesi.
  - Bir durumdan diğer duruma geçişler. Null ( $\Lambda$ ) ile geçiş yapılmaz.



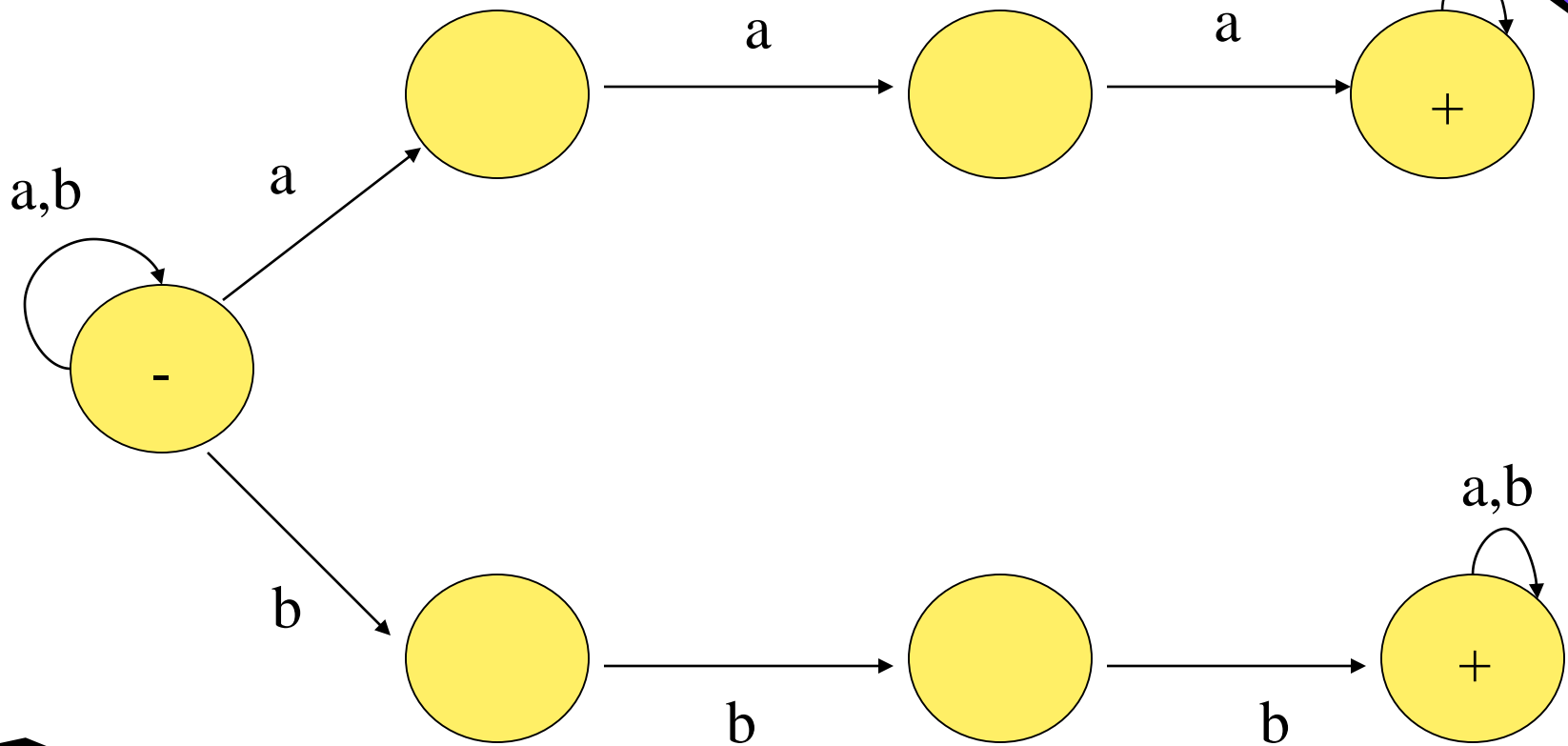
# Teorem



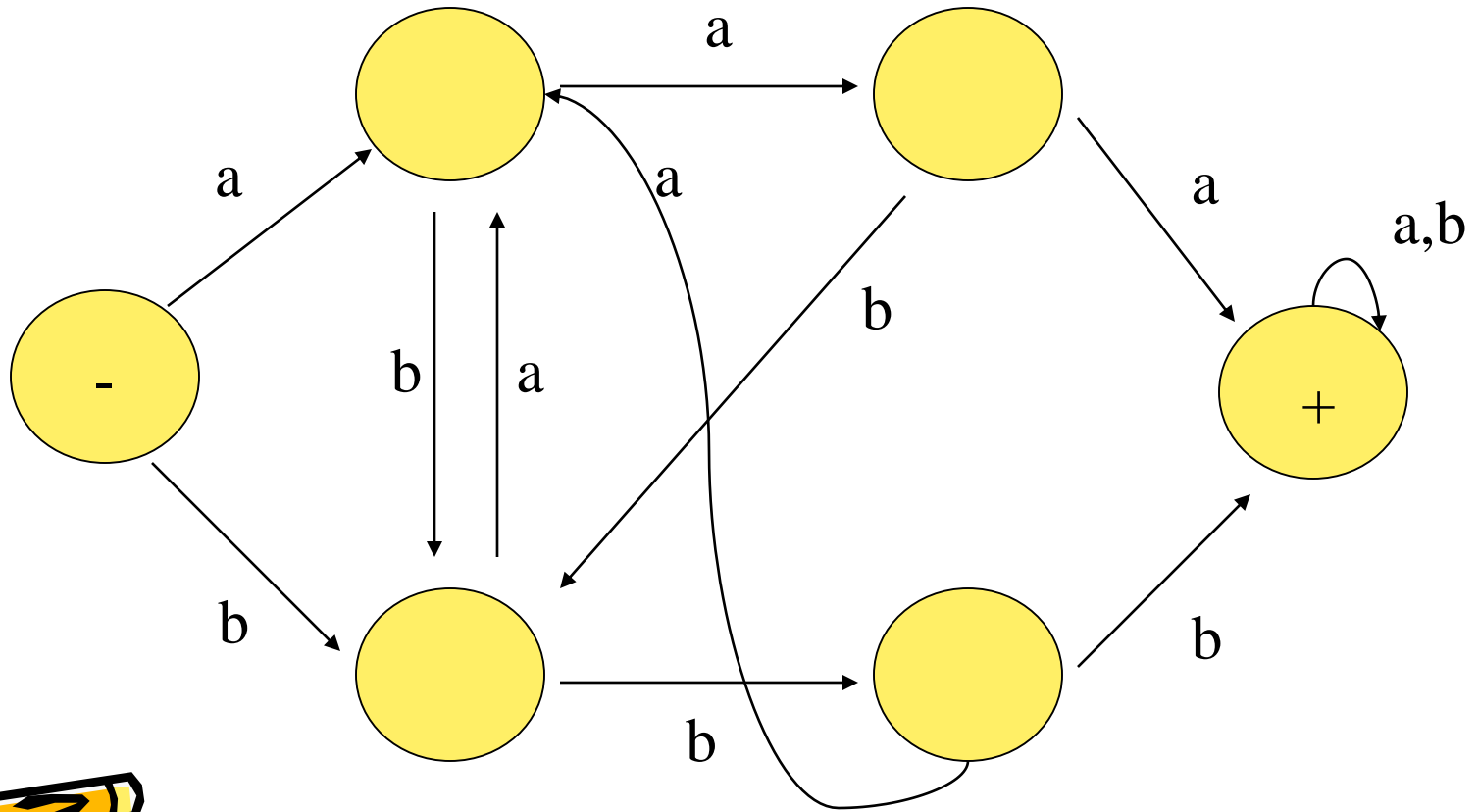
- Her NFA için aynı dili kabul eden bir Deterministik Sonlu Otomata (DFA) modeli çizilebilir.
- Örnek:
  - Yanyana 3 a (aaa) veya 3 b (bbb) içeren kelimelerin dili (language of all words that contain either triple a or triple b).
  - NFA ve DFA izleyen slaytlardadır.



# Örnek NFA



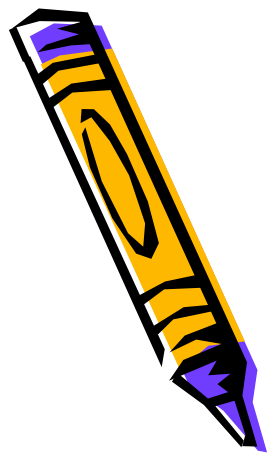
# Örnek DFA



Önceki slaytta görüldüğü gibi, NFA modelleri oluşturmak daha kolaydır.

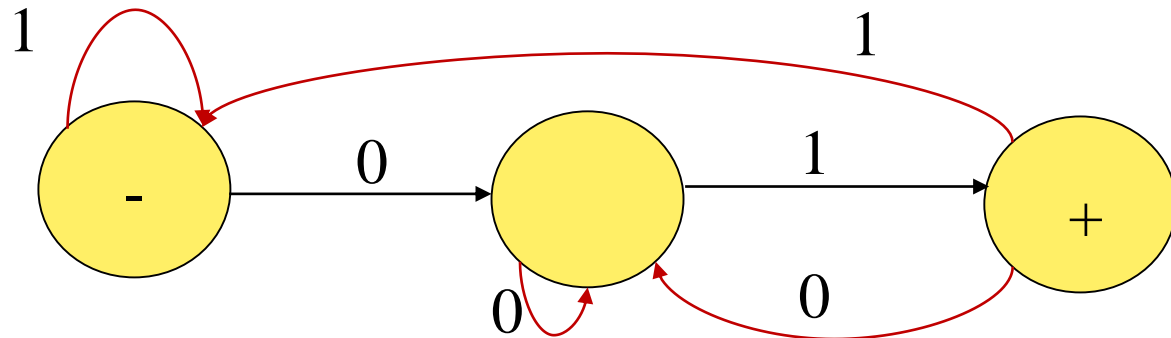
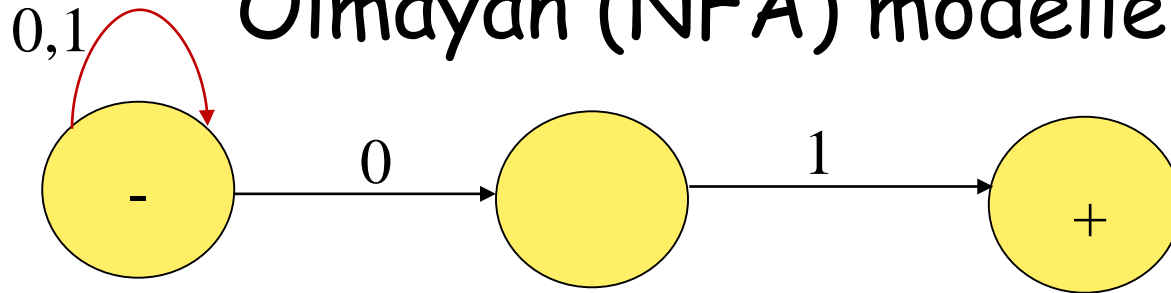
# Soru

- Alfabe =  $\{0,1\}$  olmak üzere 01 ile biten kelimelerin dili için Deterministik (DFA) ve Deterministik Olmayan (NFA) modelleri oluşturunuz.



# Soru

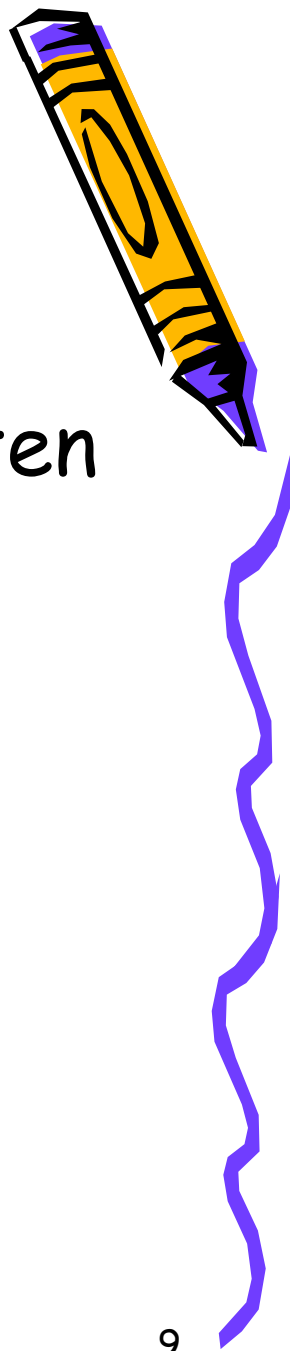
- Alfabe =  $\{0,1\}$  olmak üzere 01 ile biten kelimelerin dili için Deterministik (DFA) ve Deterministik Olmayan (NFA) modelleri oluşturunuz.





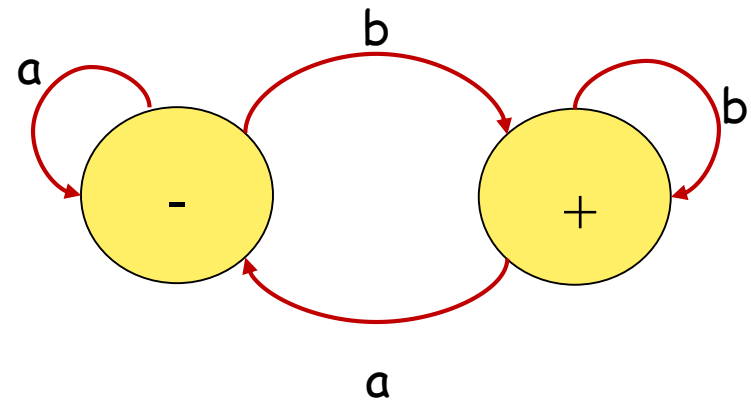
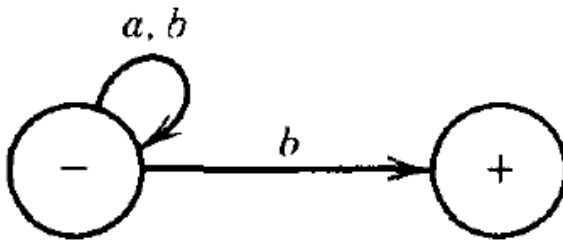
# Soru

- Alfabe =  $\{a,b\}$  olmak üzere b ile biten kelimelerin dili için Deterministik (DFA) ve Deterministik Olmayan (NFA) modelleri oluşturunuz.



# Soru

- Alfabe =  $\{a,b\}$  olmak üzere  $b$  ile biten kelimelerin dili için Deterministik (DFA) ve Deterministik Olmayan (NFA) modelleri oluşturunuz.

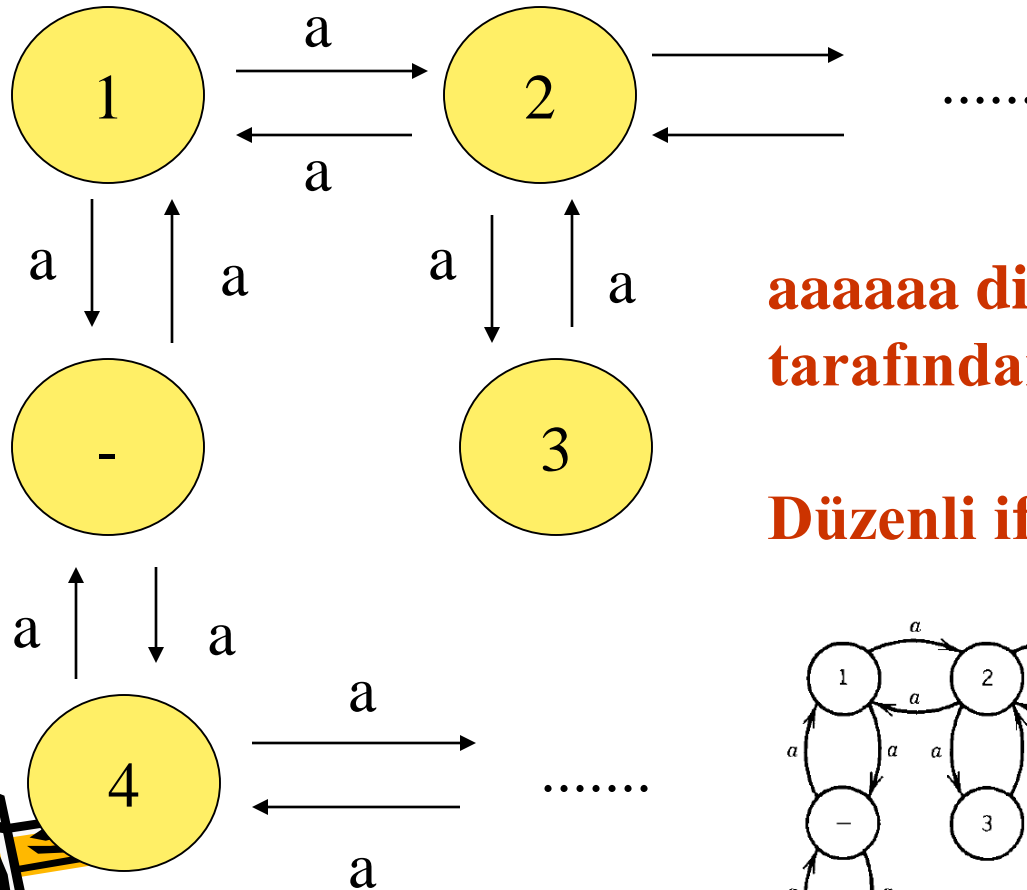


# Labirent Örneđi

- Aşağıdaki labirentten 6 adımda çıkmak mümkün müdür?

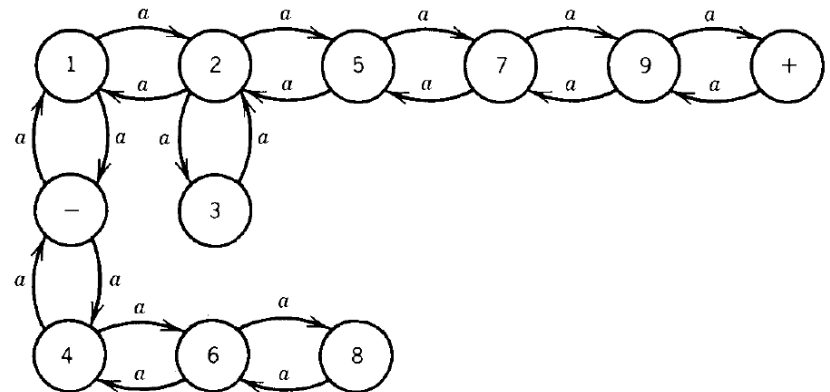
-	1	2	3
4		5	
6		7	
8		9	+

# Labirent Örneği



**aaaaaa dizisi bu makine tarafından kabul edilir mi?**

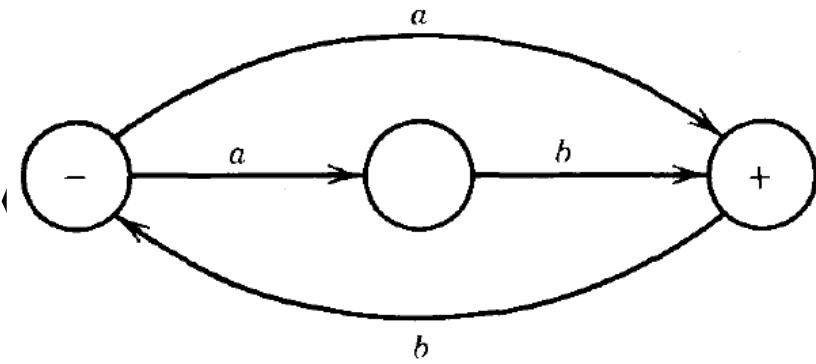
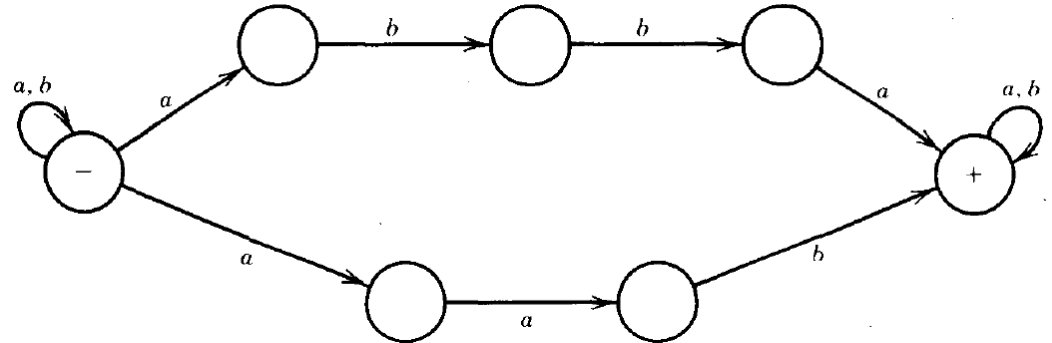
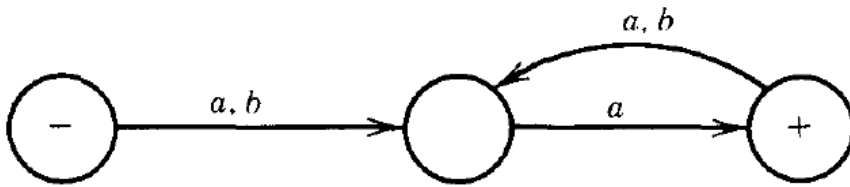
**Düzenli ifade yazabilir misiniz?**

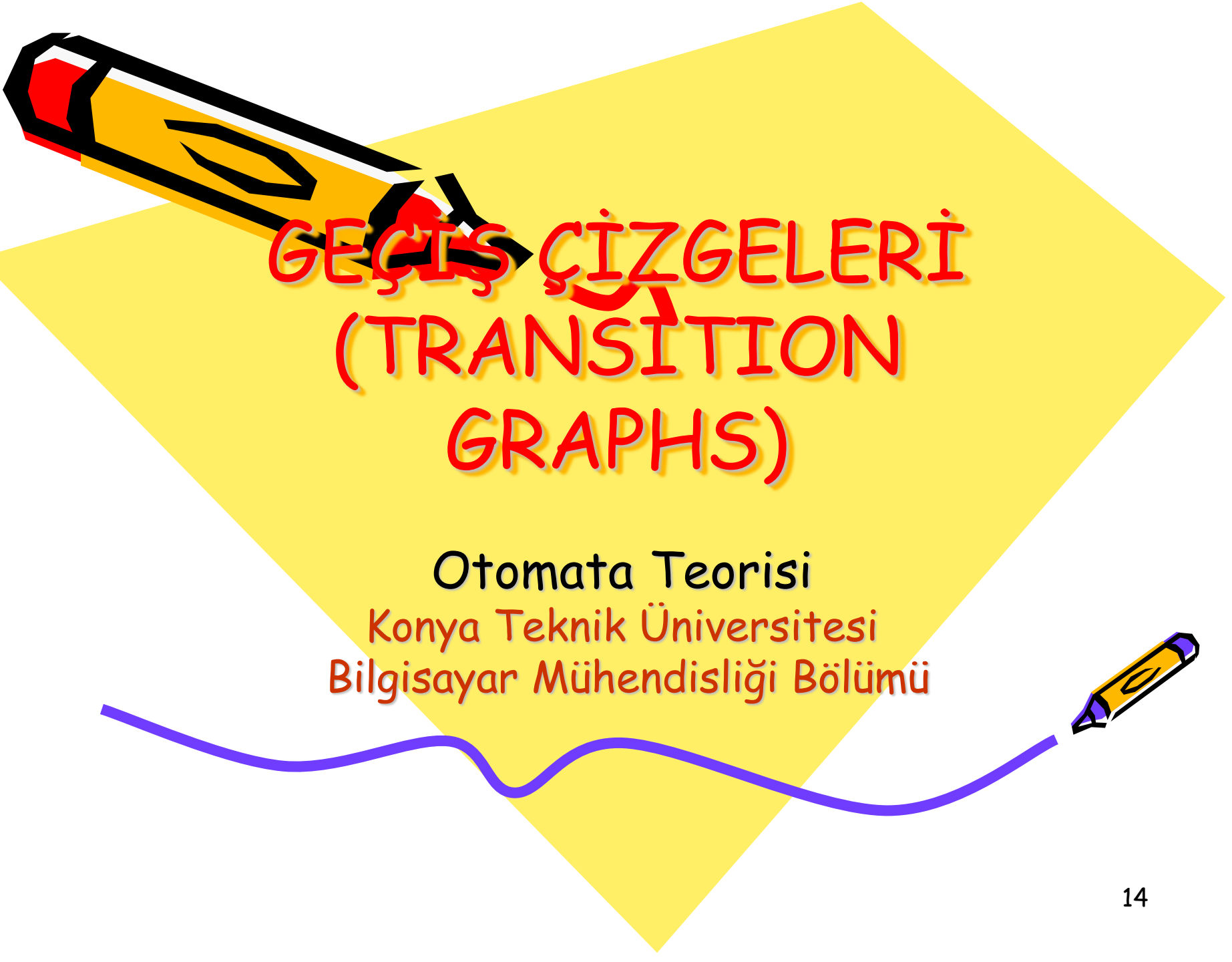


# Soru



- Alfabe =  $\{a, b\}$  olmak üzere aşağıdaki Deterministik Olmayan (NFA) modellerin dil tanımını yapınız





# GEÇİŞ ÇİZGELERİ (TRANSITION GRAPHS)

Otomata Teorisi  
Konya Teknik Üniversitesi  
Bilgisayar Mühendisliği Bölümü



# Tanım



- Geçiş çizgeleri (Transition Graphs - TG), girdi olarak verilen kelimededen bir seferde birden fazla sembol okumaya izin vermekte ve bu okunan bilgiye göre durum değiştirmektedir.
- Bir TG bir durumdan diğerine **NULL girdisi ile geçişe izin** verir.



# Tanım



- TG 3 bileşenden oluşmaktadır:
  - Bir veya birden fazlası başlangıç durumu ve yine bir veya birden fazlası sonuç durumu olabilen "durumlar kümesi".
  - Girdi olarak verilen kelimelerin oluşturduğu "alfabe".
  - NULL dahil olmak üzere, belli sembol veya "substring" lere bağlı olarak bir durumdan diğer duruma geçişi tanımlayan "geçişler".



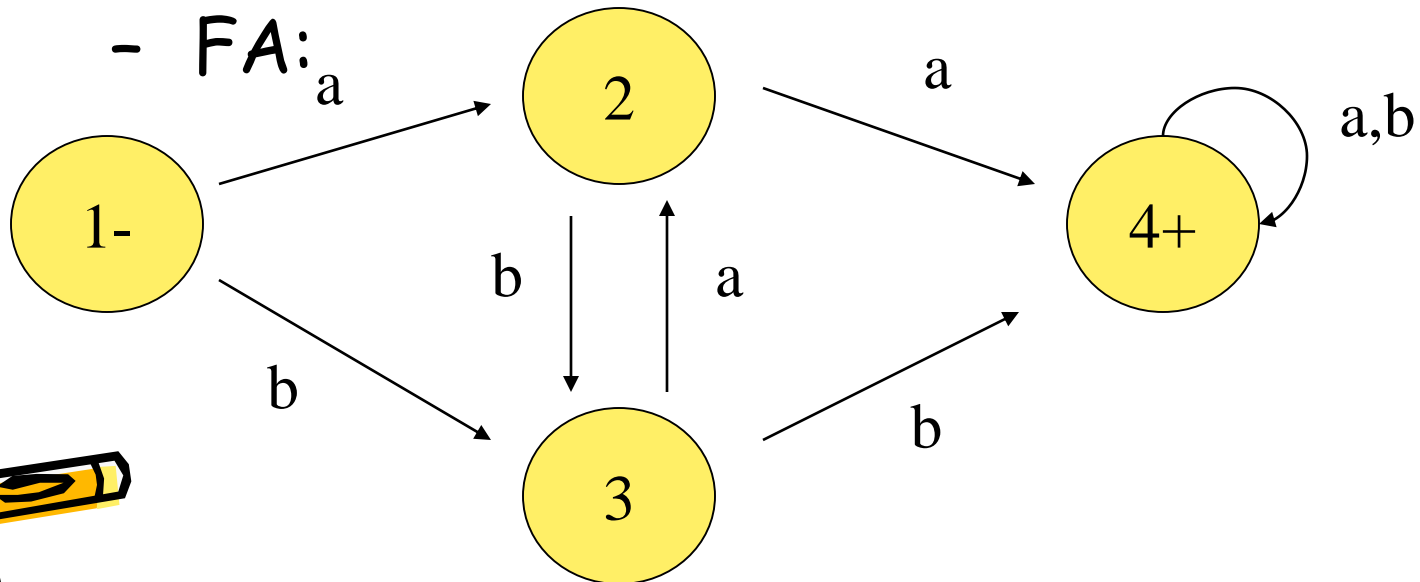


# Örnek -1

- Yanyana a veya b içeren kelimelerin dili (words that contain a doubled letter).

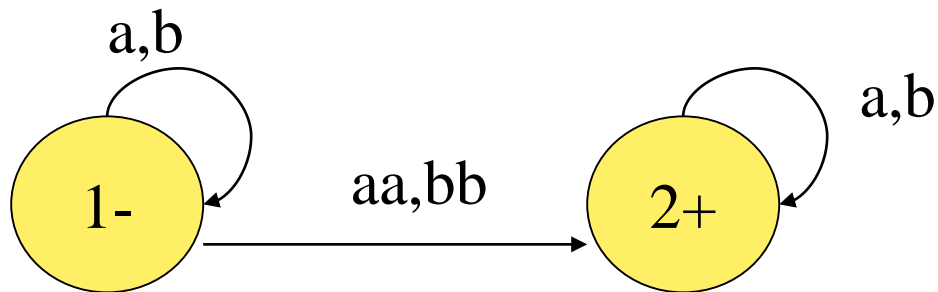
- R.E. :  $(a+b)^* (aa + bb) (a+b)^*$

- FA:



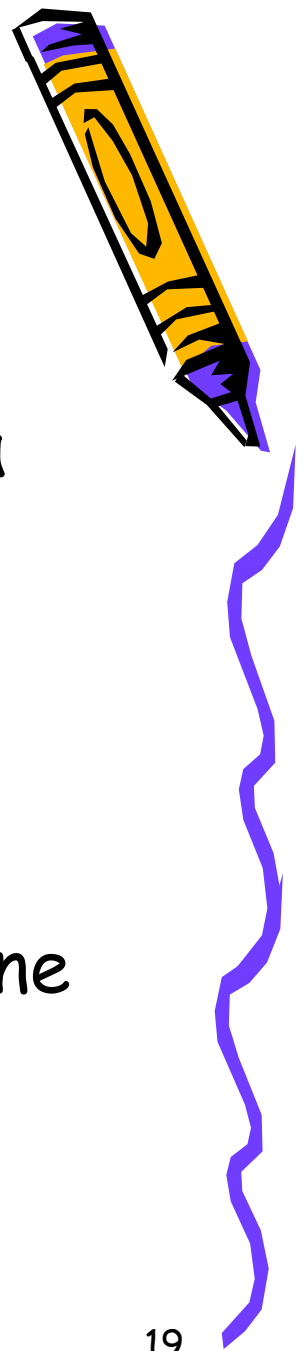
# Örnek -1 devam...

- TG: Daha az durumla modellenebilmektedir.

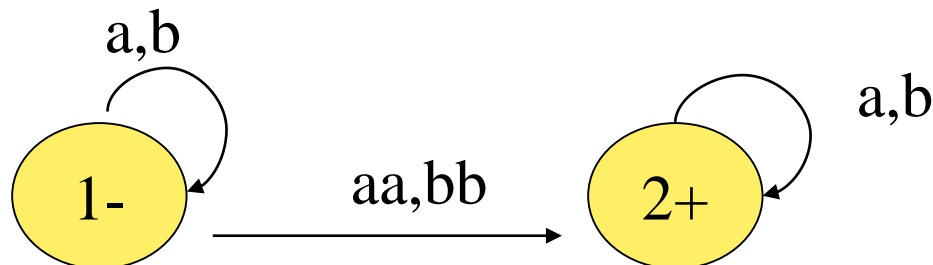


«baa» kelimesi?

# "Successfull Path" Kavramı



- Örnek-1 deki TG göz önüne alınırsa "baa" sembolü;
  - Başlangıç durumundaki döngüden
  - Başlangıçtan b tüketip, aa ile sonuç duruma geçerek elde edilebilir.
  - Eğer "ba" ile başlanacak olursa, makine takılır.



# "Successfull Path" Kavramı

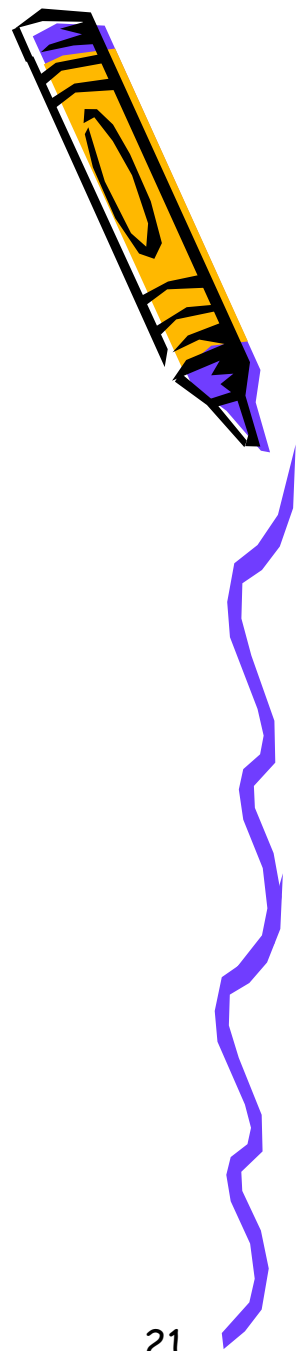
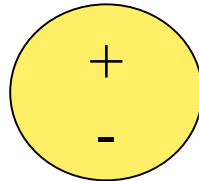


- Verilen bir kelime, ancak sonuç durumuna götüren en az bir yol olması durumunda TG tarafından kabul edilir. Sonuca götüren yol "successfull path" (başarılı yol) olarak adlandırılmaktadır.
- TG Deterministik Olmayan (Non-Deterministic) bir modeldir.



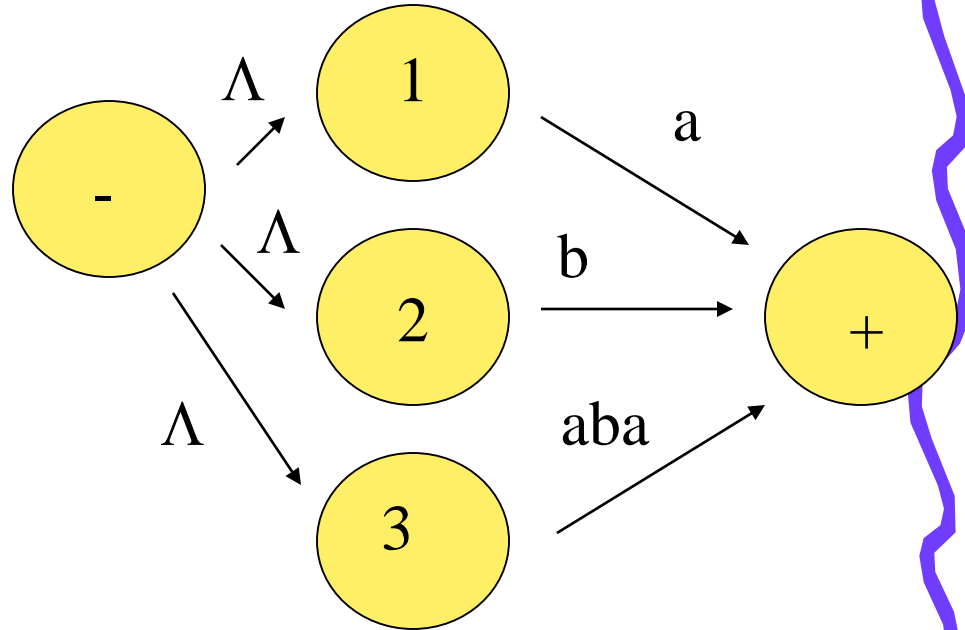
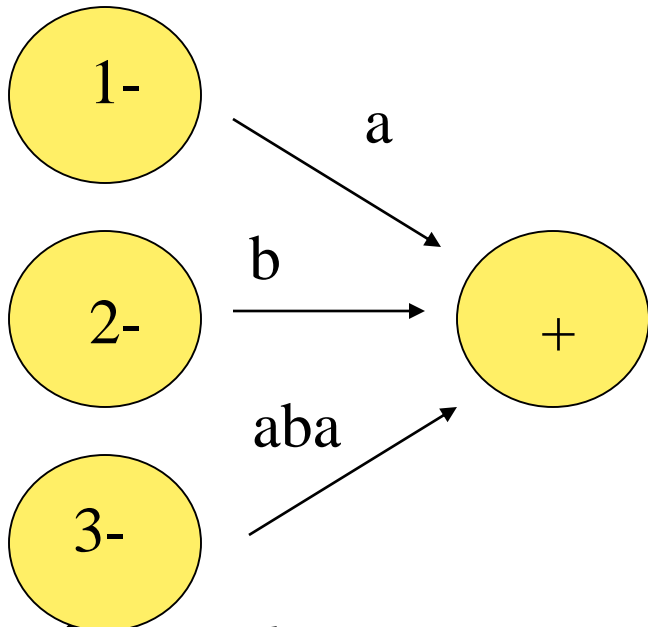
## Örnek -2

- Yalnızca NULL kabul eden T.G.



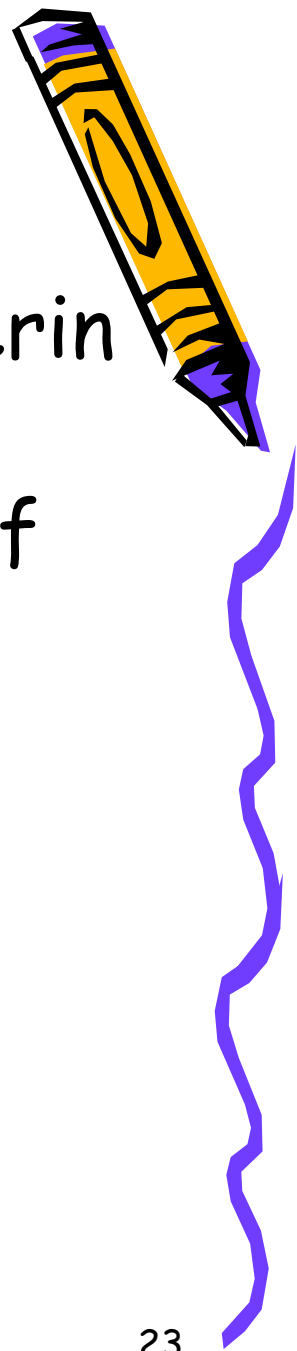
## Örnek -3

- Birden fazla başlangıç durumuna izin vermenin mantığı nedir?



$\Lambda$  geçişi izin verildiği hatırlanmalıdır.

# Soru



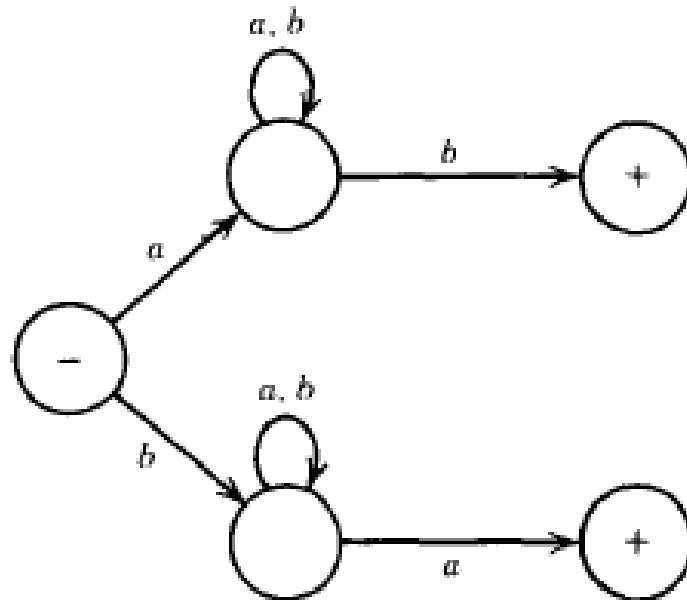
- Alfabe={a,b} olmak üzere, farklı sembollerle başlayıp biten kelimelerin dili için FA (deterministik) ve TG Modelleri oluşturunuz. (Language of all words that begin and end with different letters).



# Soru



- Alfabe={a,b} olmak üzere, farklı sembollerle başlayıp biten kelimelerin dili için FA (deterministik) ve TG Modelleri oluşturunuz. (Language of all words that begin and end with different letters).





# Soru

- Alfabe={a,b} olmak üzere, aşağıdaki dil için TG çiziniz.

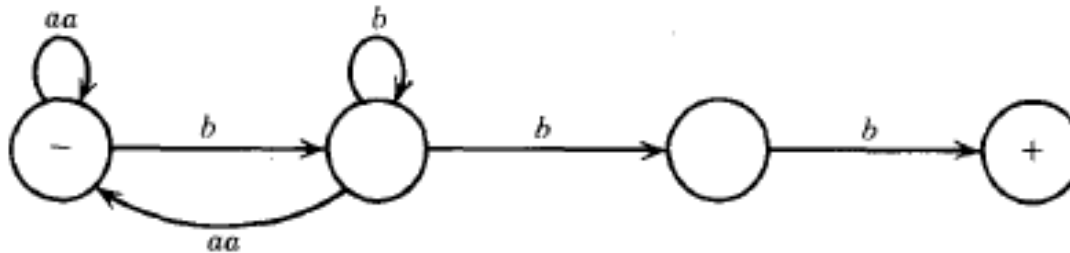
Bu TG, a'ların yalnızca çift kümeler halinde geçtiği ve üç veya daha fazla b ile biten tüm sözcüklerin dilini kabul eder



# Soru

- Alfabe={a,b} olmak üzere, aşağıdaki dil için TG çiziniz.

Bu TG, a'ların yalnızca çift kümeler halinde geçtiği ve üç veya daha fazla b ile biten tüm sözcüklerin dilini kabul eder

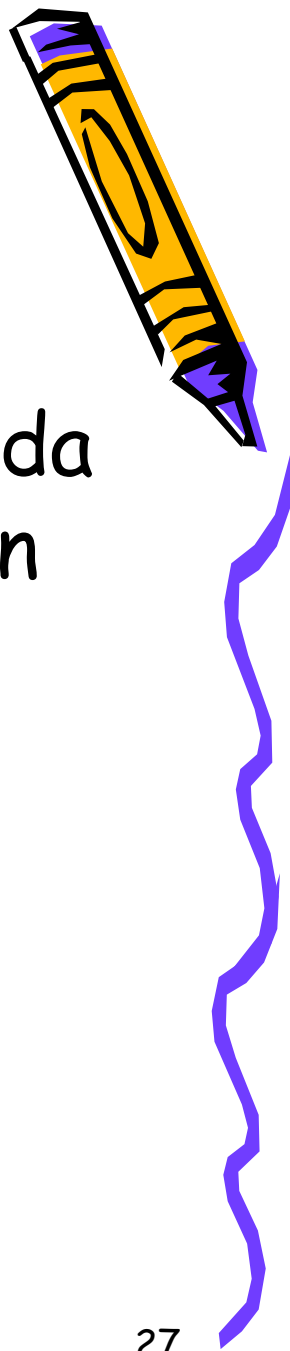


accepts the language of all words in which the *a*'s occur only in even clumps and that end in three or more *b*'s. ■

Alfabe={a,b} olmak üzere, «aa» ları en az 3 tane «b» lerin takip ettiği kelimelerin dili için TG çiziniz?

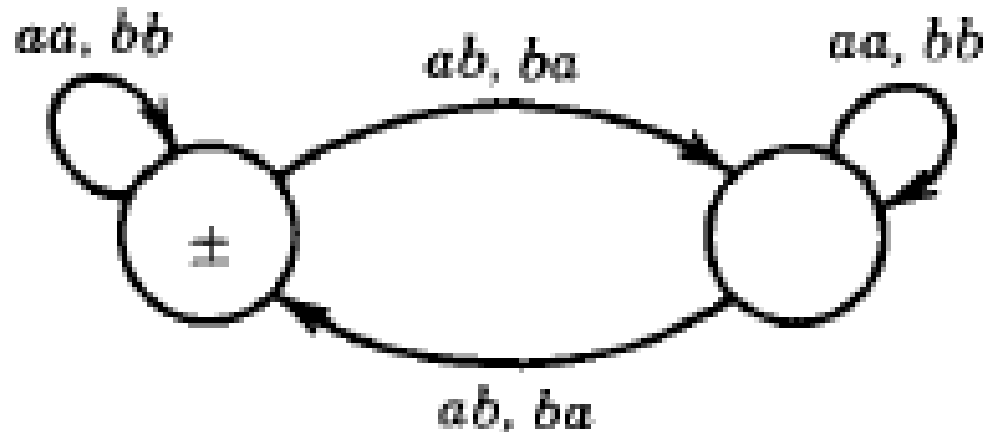
# Soru

- Alfabe={a,b} olmak üzere, çift sayıda a ve çift sayıda b içeren kelimelerin dili (EVEN-EVEN) için TG Modeli oluşturunuz.

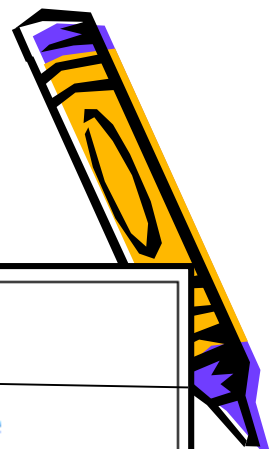


# Soru

- Alfabe={a,b} olmak üzere, çift sayıda a ve çift sayıda b içeren kelimelerin dili (EVEN-EVEN) için TG Modeli oluşturunuz.



# DFA, NFA ve TG Karşılaştırması



	<b>FA</b>	<b>NFA</b>	<b>TG</b>
<b>Start States</b>	<b>One</b>	<b>One</b>	<b>One or more</b>
<b>Final States</b>	<b>Some or none</b>	<b>Some or none</b>	<b>Some or none</b>
<b>Edge Labels</b>	<b>Letters from <math>\Sigma</math></b>	<b>Letters from <math>\Sigma</math></b>	<b>Words from <math>\Sigma^*</math></b>
<b>Number of Edges From Each State</b>	<b>One for each letter in <math>\Sigma</math></b>	<b>Arbitrary</b>	<b>Arbitrary</b>
<b>Deterministic?</b>	<b>Yes</b>	<b>Not necessarily</b>	<b>Not necessarily</b>
<b>Every Path Represents One Word?</b>	<b>Yes</b>	<b>Yes</b>	<b>Yes</b>





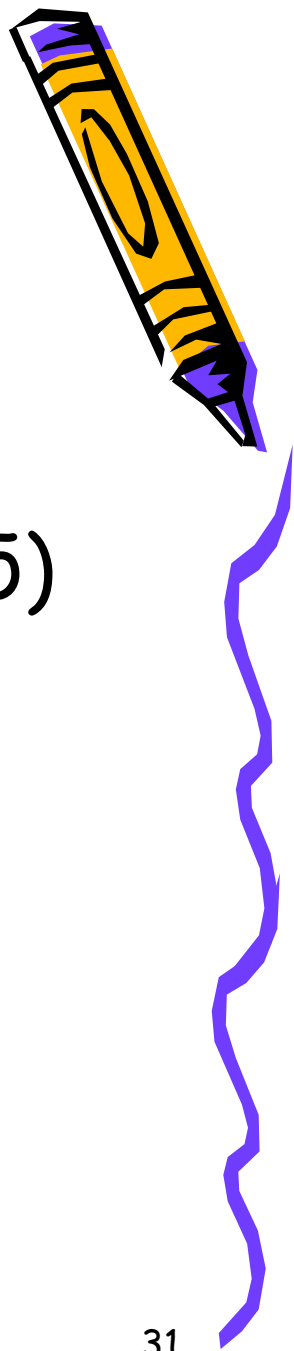
# ÇIKTILI SONLU OTOMATA (Finite Automata with Output)

Otomata Teorisi  
Konya Teknik Üniversitesi  
Bilgisayar Mühendisliği Bölümü



# Çıktı Veren F.A. Türleri

- Moore Makinesi (E.F.Moore,1956)
- Mealey Makinesi (G.H.Mealey, 1955)



# Moore Makinesi

- Moore makinesi aşağıda verilen beşli ile tanımlanabilmektedir.
  - $Q$  : Sonlu durumlar kümesi:  $q_0, q_1, q_2, \dots$  ( $q_0$  başlangıç durumudur).
  - $\Sigma$  : Girdi dizisini oluşturmak için semboller alfabeti.
  - $\Gamma$  : Olası **çıkı**tı sembolleri alfabeti.
  - Her girdi ile durumlar arası geçişin nasıl olduğunu gösteren geçiş tablosu.
  - Her **varılan** durumda hangi çıkı

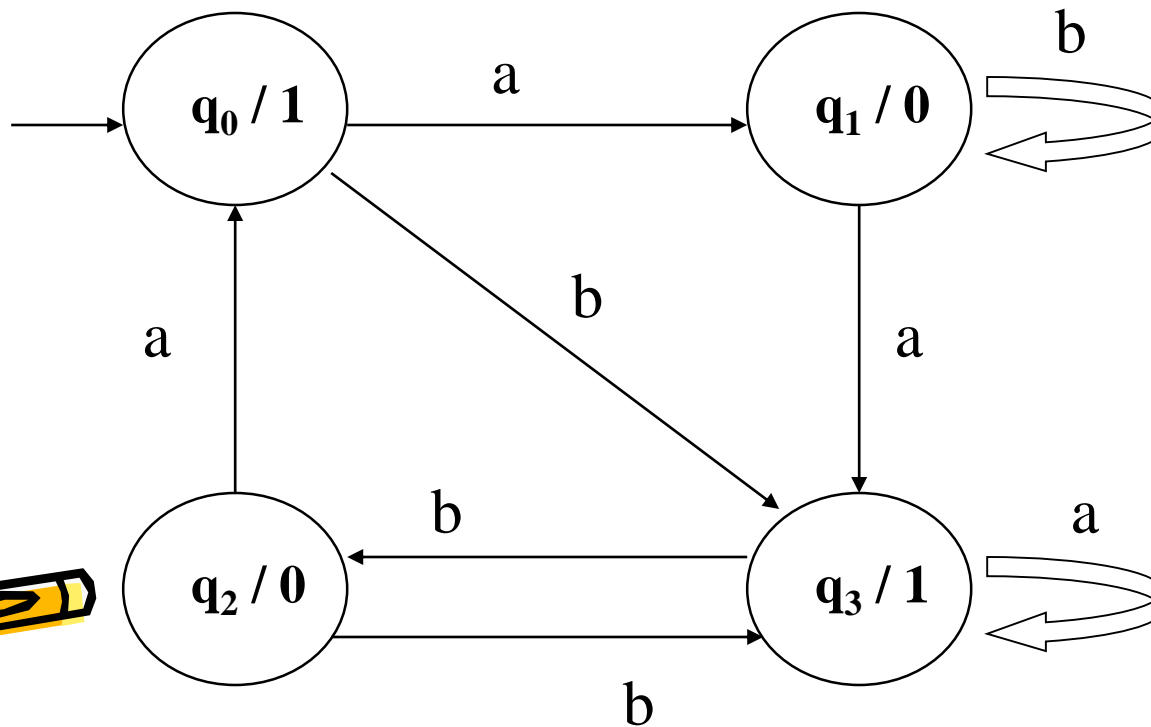




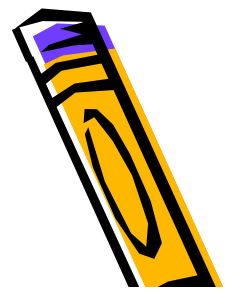
$Q : \{ \dots \}$   
 $\Sigma : \{ \dots \}$   
 $\Gamma : \{ \dots \}$

# Moore Örnek-1

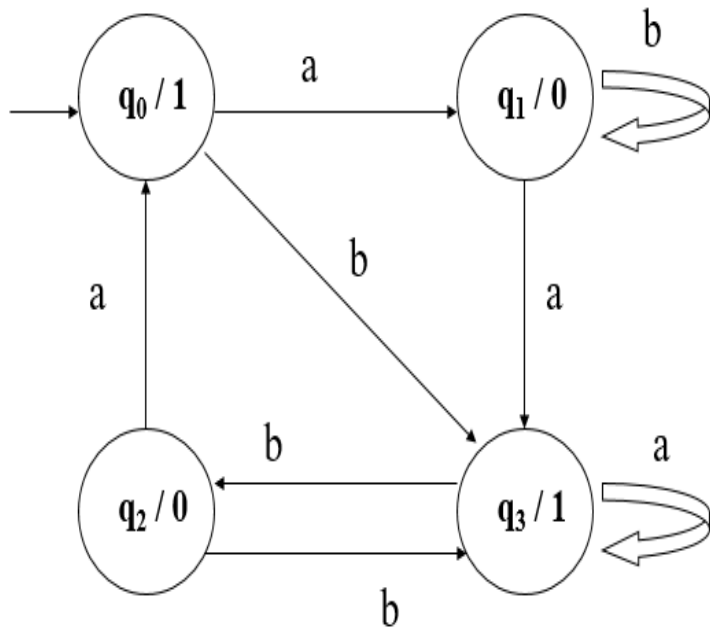
- Input: abab Output: 10010



# Moore Örnek-1



- Input: abab Output: 10010



Input alphabet:  $\Sigma = \{a, b\}$

Output alphabet:  $\Gamma = \{0, 1\}$

Names of states:  $q_0, q_1, q_2, q_3$ , ( $q_0$  = start state)

Transition Table

New State

Output Table

(The Character Printed in the Old State)

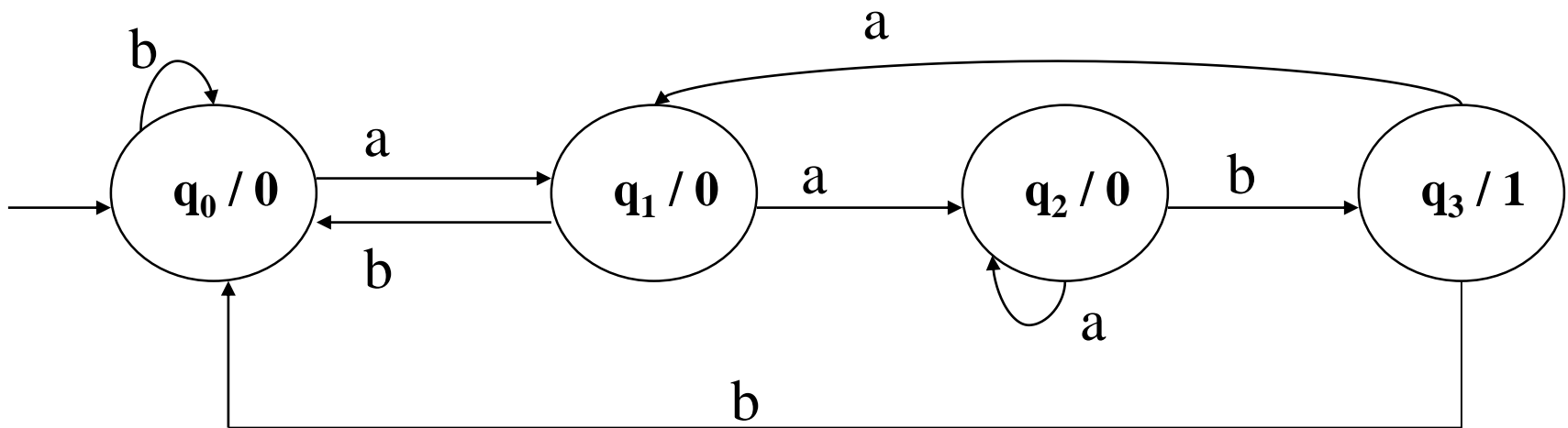
Old State	After Input <i>a</i>	After Input <i>b</i>	
$q_0$	$q_1$	$q_3$	1
$q_1$	$q_3$	$q_1$	0
$q_2$	$q_0$	$q_3$	0
$q_3$	$q_3$	$q_2$	1

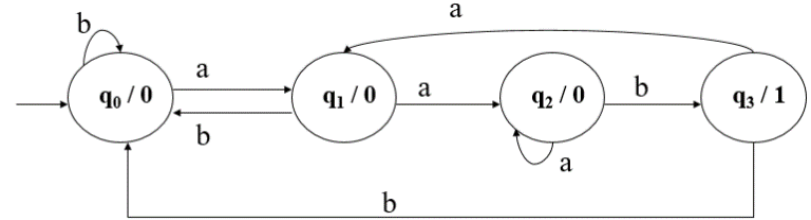


# Moore Örnek-2



- Verilen bir string içinde "aab" nin kaç defa bulunduğunu sayan "Moore Makinesi":





# Moore Örnek-2



- Örnek-2 için "Transition" ve "Output" Tablosu:

Transition Table			Output Table
Old State	After input a	After input b	Character printed
— q <sub>0</sub>	q <sub>1</sub>	q <sub>0</sub>	0
q <sub>1</sub>	q <sub>2</sub>	q <sub>0</sub>	0
q <sub>2</sub>	q <sub>2</sub>	q <sub>3</sub>	0
q <sub>3</sub>	q <sub>1</sub>	q <sub>0</sub>	1

Input String

State

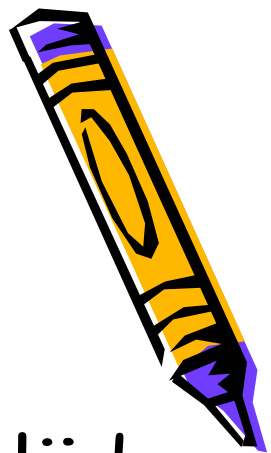
Output

	a	a	a	b	a	b	b	a	a	b	b	
State	q <sub>0</sub>	q <sub>1</sub>	q <sub>2</sub>	q <sub>2</sub>	q <sub>3</sub>	q <sub>1</sub>	q <sub>0</sub>	q <sub>0</sub>	q <sub>1</sub>	q <sub>2</sub>	q <sub>3</sub>	q <sub>0</sub>
Output	0	0	0	0	1	0	0	0	0	0	1	0



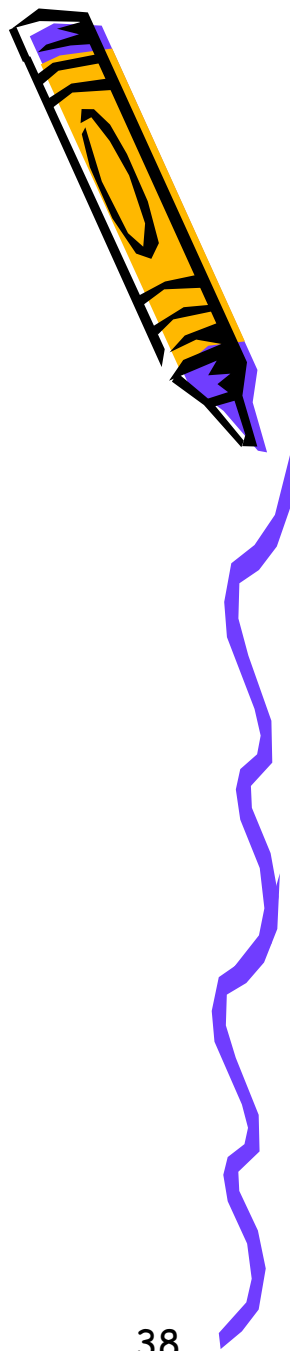
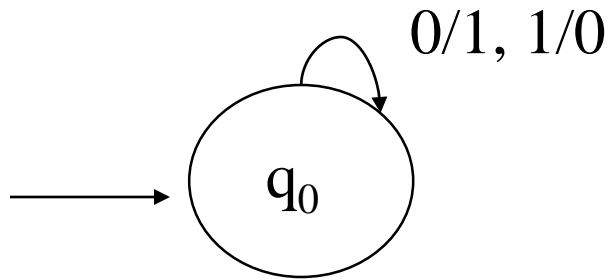
# Mealey Makinesi

- A Mealy Makinesi aşağıdaki dörtlünden oluşmaktadır:
  - Sonlu durumlar kümesi:  $q_0, q_1, q_2, \dots$  ( $q_0$  başlangıç durumudur).
  - Girdi dizisini oluşturmak için semboller alfabesi.
  - Olası çıktı sembolleri alfabesi.
  - Geçiş diyagramında her geçiş " $i/o$ " şeklinde gösterilir. Burada, " $i$ " girdi sembolünü, " $o$ " ise çıktı olarak verilen sembolü göstermektedir.



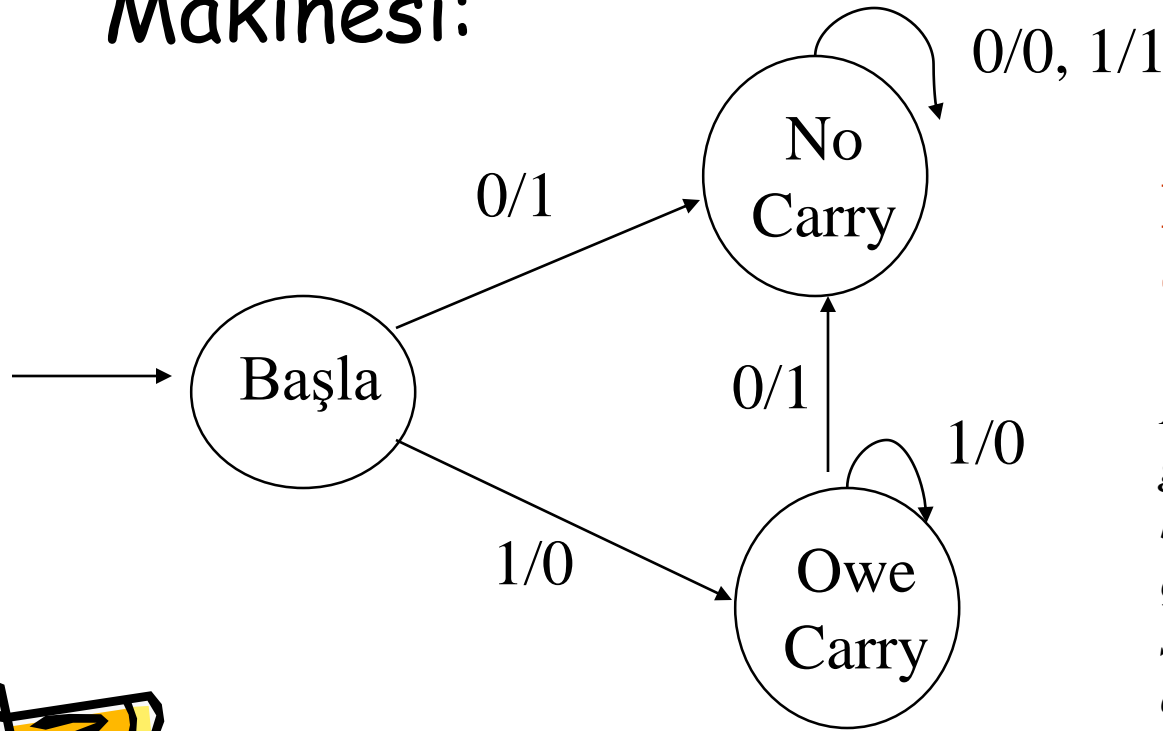
# Örnek-1

- 1'in tümleyenini bulan Mealey Makinesi:



## Örnek-2

- "1 Artırım" işlemi için Mealey Makinesi:



X0	X1
1	1
---	---
X1	X0
ELDE	VAR
YOK	X ?
0-->0	0-->1
1-->1	1-->0

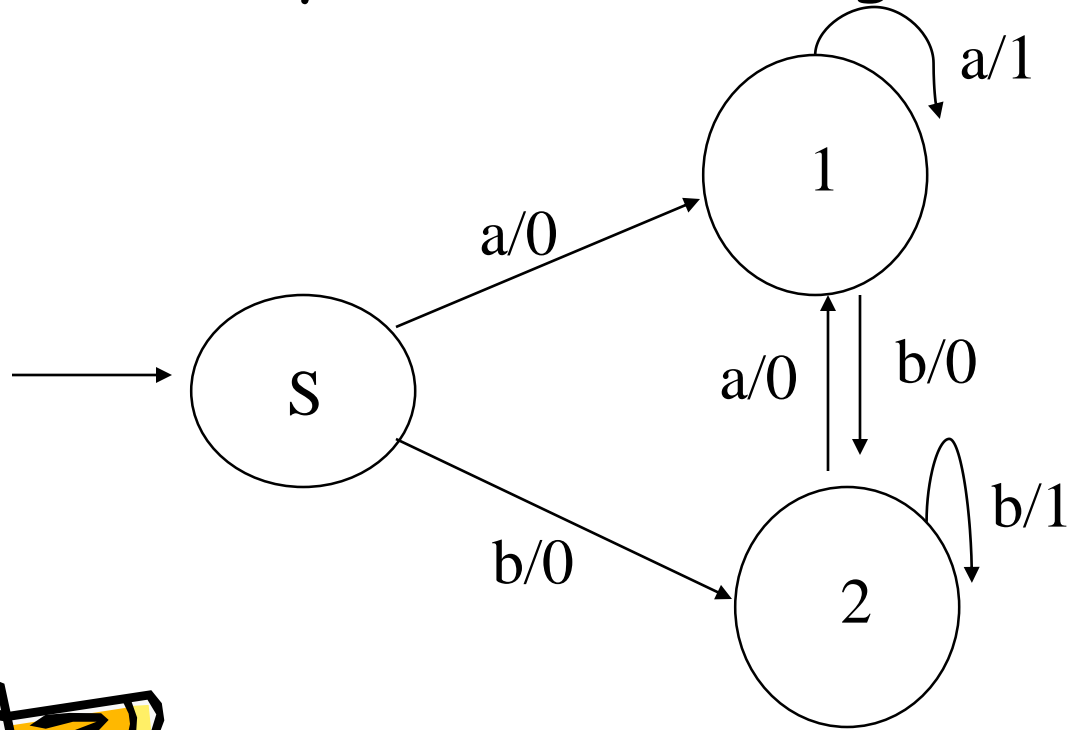
**Input:1011**

**Output:1100**

*Not: Bu örnek için, girdinin en sağ bit'ten başlayarak verildiği, çıktının da sağdan sola doğru oluştuğu dikkate alınmalıdır.*

# Örnek-3

- Kelime içindeki aa ve bb sayısını bulan Mealey Makinesi Örneği:



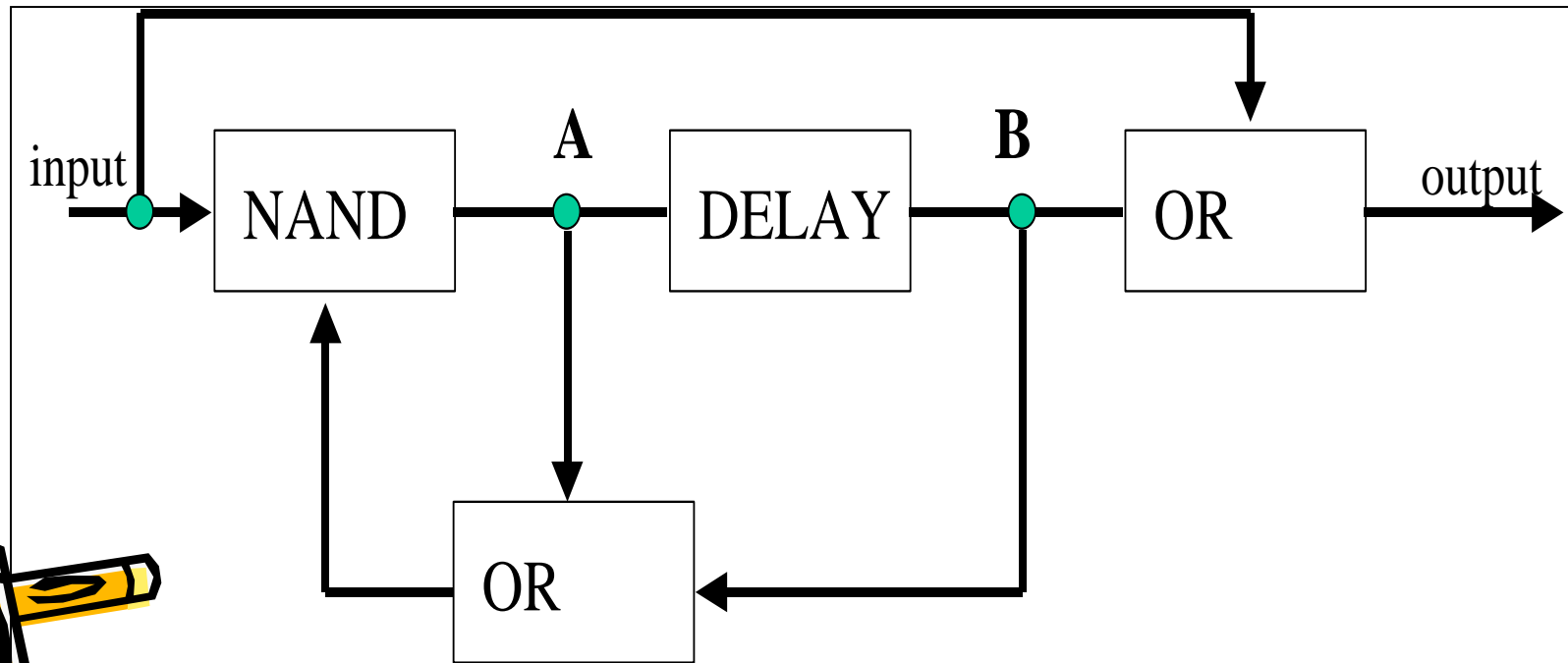
**Input:aaa**  
**Output:011**

**Input:ababbaab**  
**Output:00001010**



## Örnek -4

- “Sequential” mantık devrelerinin Mealey Makinesi ile modellenmesi:



NAND=NOT AND 41  
DELAY bir D Flip flopudur.

# Örnek- 4

Ders kitabından alıntı:

**Four states based on whether there is current at points A and B in the circuit or not? (Current in a wire is denoted by 1, no current by 0.)**

**$q_0$ : A=0 B=0**

**$q_1$ : A=0 B=1**

**$q_2$ : A=1 B=0**

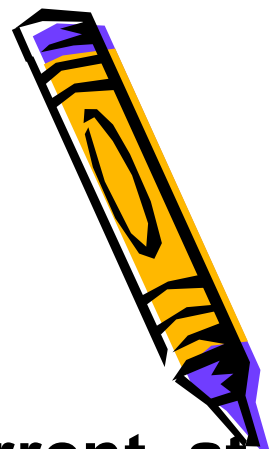
**$q_3$ : A=1 B=1**

The operation of this circuit is such that after an input of 0 or 1 the state changes according to the following rules:

new  $B$  = old  $A$

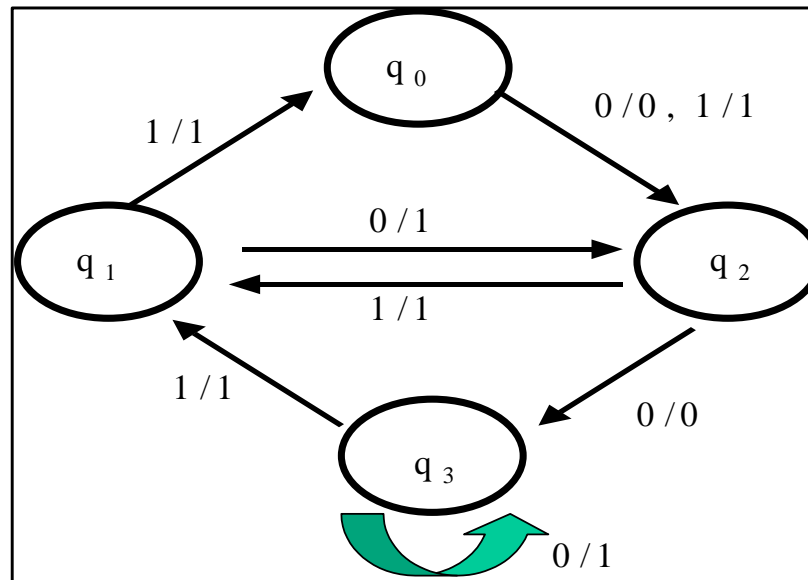
new  $A$  = (input) NAND (old  $A$  OR old  $B$ )

output = (input) OR (old  $B$ )



# Örnek-4

	After input 0		After input 1	
Old State	New State	Output	New State	Output
$q_0$	$q_2$	0	$q_2$	1
$q_1$	$q_2$	1	$q_0$	1
$q_2$	$q_3$	0	$q_1$	1
$q_3$	$q_3$	1	$q_1$	1



# Tüm FA Modelleri için Karşılaştırma

	<i>FA</i>	<i>TG</i>	<i>NFA</i>	<i>NFA-<math>\Lambda</math></i>	<i>Moore</i>	<i>Mealy</i>
<i>Start States</i>	One	One or more	One	One	One	One
<i>Final States</i>	Some or none	Some or none	Some or none	Some or none	None	None
<i>Edge Labels</i>	Letters from $\Sigma$	Words from $\Sigma^*$	Letters from $\Sigma$	Letters from $\Sigma$ and $\Lambda$	Letters from $\Sigma$	i/o i from $\Sigma$ o from $\Gamma$
<i>Number of Edges from Each State</i>	One for each letter in $\Sigma$	Arbitrary	Arbitrary	Arbitrary	One for each letter in $\Sigma$	One for each letter in $\Sigma$
<i>Deterministic?</i>	Yes	No	No	No	Yes	Yes
<i>Output</i>	No	No	No	No	Yes	Yes

# Ödev 26

Moore Makinesini simüle eden program kodunu üretiniz. Kullanıcıdan girdi olarak Sonlu durumlar kümesinin eleman sayısını (kümenin ilk elemanı her zaman başlangıç durumudur), Girdi dizisini oluşturmak için semboller alfabesini ve Olası çıktı sembolleri alfabesini alacaktır. Kullanıcıdan bilgi alışı arayüz aracılığıyla yapabileceğiniz gibi bir text (INPUT.TXT) dosyadan aşağıdaki formatta almasını sağlayabilirsiniz.

$$\begin{aligned} Q &= \{q_0, q_1, q_2, \dots, q_{10}\}. \\ \Sigma &= \{a, b\} \\ \Gamma &= \{0, 1\} \end{aligned}$$

Her girdi ile durumlar arası geçişin nasıl olduğunu gösteren geçiş tablosu ve Her varılan durumda hangi çıktının verildiğini gösteren çıktı tablosu bilgileri de kullanıcı tarafından organize edilecektir. Bunlar içinde arayüz tasarlanabileceği gibi iki ayrı text (GECISTABLOSU.TXT ve OUTPUT.TXT) dosyadan alınması da sağlanabilir. Text dosyada ilk satırlar başlık satırları ilk sütunlarda durum sütunları olacaktır. Her bir öge arası TAB karakteri ile ayrılmış olacaktır.

Programınız dışarıdan girilen Giriş sitringi için; durumlar arası geçişleri gösterebilmeli ve nihai çıktıyı da kullanıcıya göstermelidir. GÖRSELLİK ekstra değerlendirilecektir.



# Ödev 27

Mealey Makinesini simüle eden program kodunu üretiniz. Kullanıcıdan girdi olarak Sonlu durumlar kümesinin eleman sayısını (kümenin ilk elemanı her zaman başlangıç durumudur), Girdi dizisini oluşturmak için semboller alfabesini ve Olası çıktı sembolleri alfabesini alacaktır. Kullanıcıdan bilgi alışı arayüz aracılığıyla yapabileceğiniz gibi bir text (INPUT.TXT) dosyadan aşağıdaki formatta almasını sağlayabilirsiniz.

$Q: \{q_0, q_1, q_2, \dots, q_{10}\}.$

$\Sigma = \{a, b\}$

$\Gamma = \{0, 1\}$

Her girdi ile durumlar arası geçişin nasıl olduğunu gösteren Geçiş diyagramı bilgileri de kullanıcı tarafından organize edilecektir. Bunlar içinde arayüz tasarlanabileceği gibi tek bir text (GECISDIYAGRAMI.TXT) dosyadan alınması da sağlanabilir. Text dosyada ilk satırlar başlık satırları ilk sütunlarda durum sütunları olacaktır. Her bir öge arası TAB karakteri ile ayrılmış olacaktır. Her geçiş "i/o" şeklindedir.

Programınız dışarıdan girilen Giriş sitringi için; durumlar arası geçişleri gösterebilmeli ve nihai çıktıyı da kullanıcıya göstermelidir. GÖRSELLİK ekstra değerlendirilecektir.

