

# FINITE STATE AUTOMATA DETERMINISTIC FINITE STATE (DFA)

---



DR. ŞAFAK KAYIKÇI


# Sonlu Durumlar Makinesi (FSA)

---

Sembol	= a, b, c, 0, 1, 2, .....	
Alfabe	= $\Sigma$ (sigma) semboller topluluğu	Örn = {a, b}, {d, e, g}, {0, 1, 2}....
String(Katar)	= semboller (karakterler) dizesi	Örn = a, b, 0, 1, aa, bb, 01, ...
Dil	= stringler kümesi	

$\Sigma = \{0,1\} \rightarrow$  alfabe bu olsun. Bu alfabe üzerinden yapılabilecek diller :

L1 = Boyutu 2 olan bütün stringler = {00, 01, 10, 11}  sonlu küme  
L2 = Boyutu 3 olan bütün stringler = {000, 001, 010, 011, 100, 101, 110, 111} 

L3 = 0 ile başlayan bütün stringler = {0, 00, 01, 0001, 010, 01111, .... }  sonsuz küme

# $\Sigma$ kuvvetleri

---

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

$$\Sigma^0 = \text{Boyutu 0 olan bütün stringler} = \Sigma^0 = \{\epsilon\} \text{ ( bazı yerlerde } \lambda \text{ ile gösterilir)}$$

$$\Sigma^1 = \text{Boyutu 1 olan bütün stringler} = \Sigma^1 = \{0, 1\}$$

$$\Sigma^2 = \text{Boyutu 2 olan bütün stringler} = \Sigma^2 = \{00, 01, 10, 11\}$$

$$\Sigma^3 = \text{Boyutu 3 olan bütün stringler} = \Sigma^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$$

.....

$$\Sigma^n = \text{Boyutu n olan bütün stringler} = \Sigma^n$$

$$\text{Kümenin eleman sayısı (cardinality) } \Sigma^n = 2^n$$

Sigma star (Kleene Yıldızı)

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \dots$$

$$= \{\epsilon\} \cup \{0,1\} \cup \{00, 01, 10, 11\} \cup \dots$$

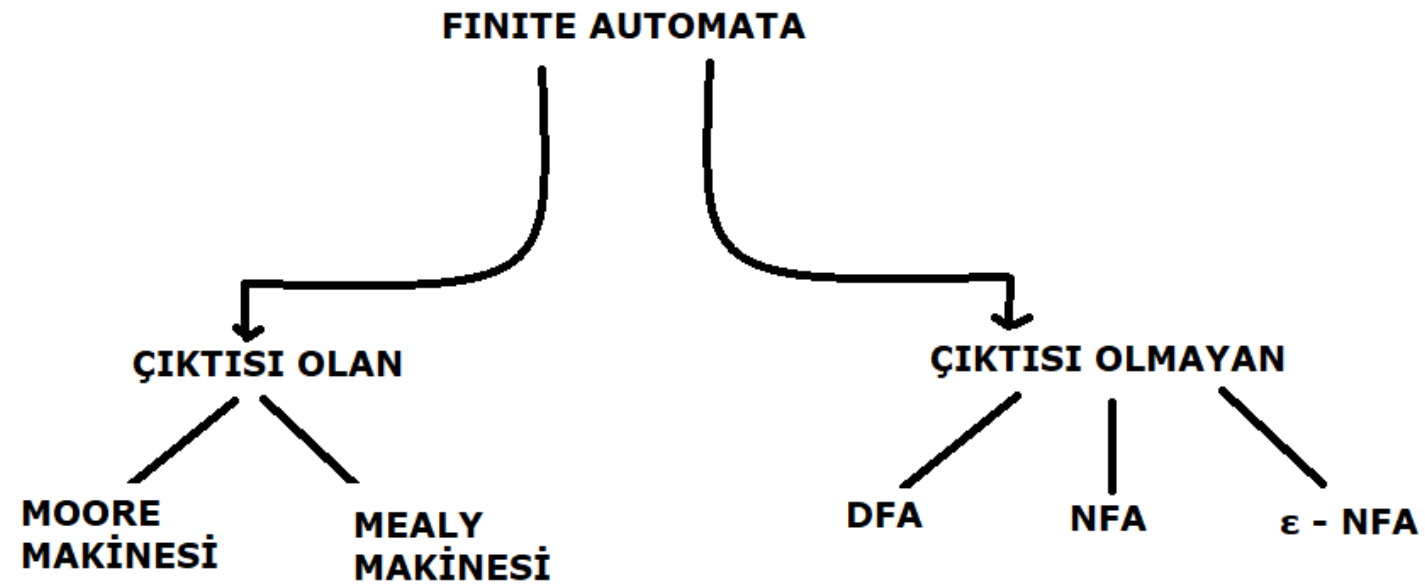
= bütün olasılıklar

Kleene Kapaması (Artı)

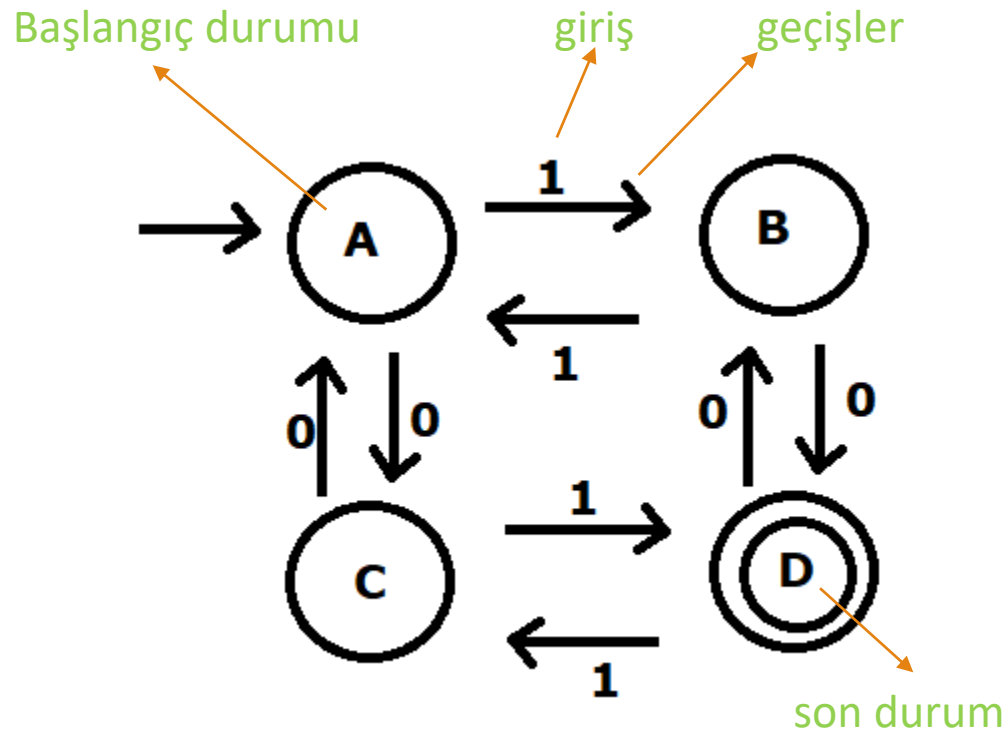
$$\Sigma^+ = \Sigma^* - \{\epsilon\}$$

# Sonlu Durumlar Makinesi

---



# DETERMINISTIC FINITE AUTOMATA (DFA)



$(Q, \Sigma, q_0, F, \delta)$

$Q$  = tüm durumların kümesi

$\Sigma$  = giriş alfabesi

$q_0$  = başlangıç durumu

$F$  = son durumlar kümesi

$\delta$  = geçiş fonksiyonları  $Q \times \Sigma \rightarrow Q$

$Q = \{A, B, C, D\}$

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

$q_0 = A$

$\delta$  = geçiş fonksiyonları  $Q \times \Sigma \rightarrow Q$

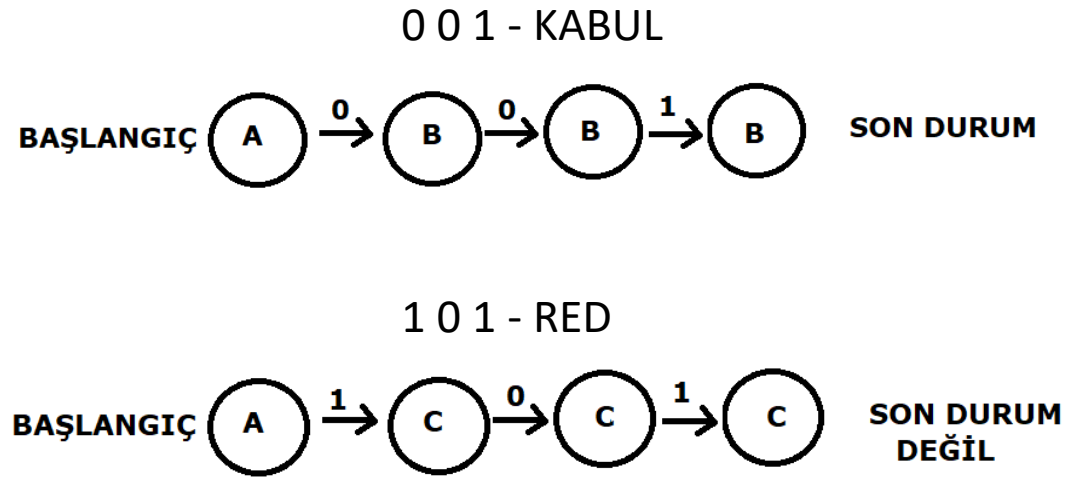
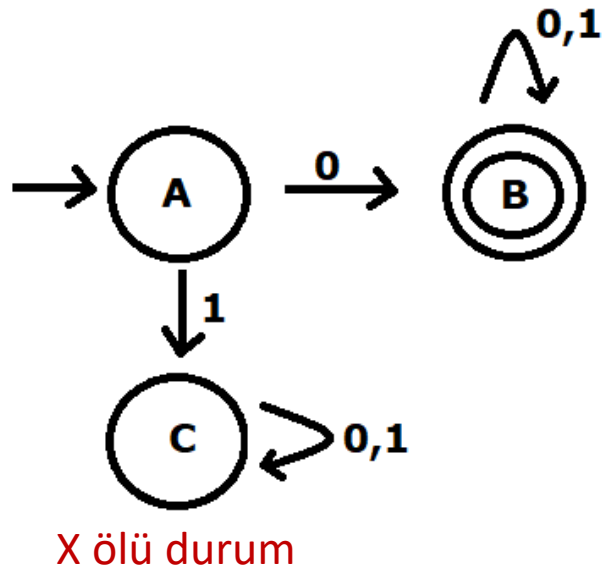
$F = \{D\}$

$\delta$  tablosu

	0	1
A	C	B
B	D	A
C	A	D
D	B	C

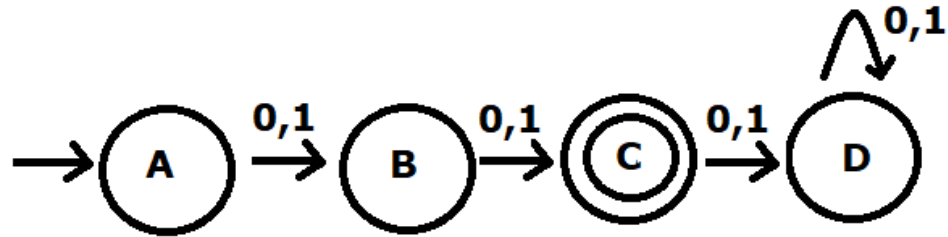
# DFA – örnek 1

$L_1 = 0$  ile başlayan bütün stringler =  $\{0, 00, 01, 000, 010, 011, \dots\}$

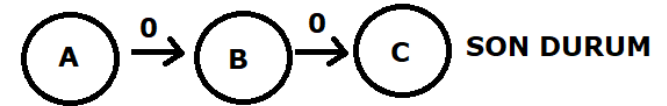


# DFA – örnek 2

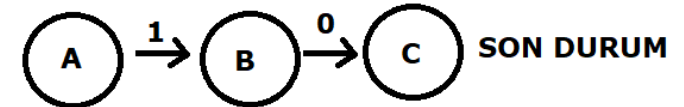
L2 = boyutu iki olan bütün stringler = {00,01,10,11}



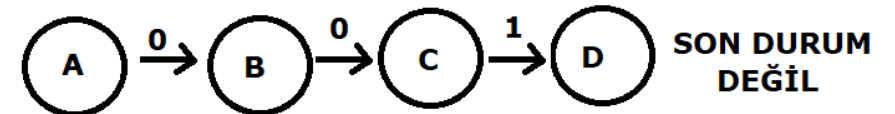
0 0 - KABUL



1 0 - KABUL



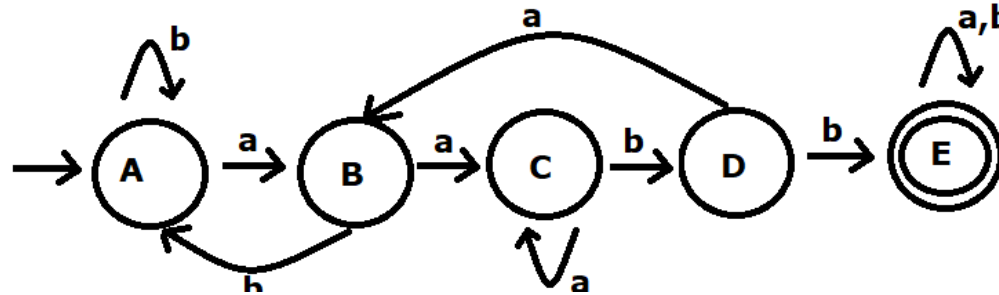
0 0 1 - RED



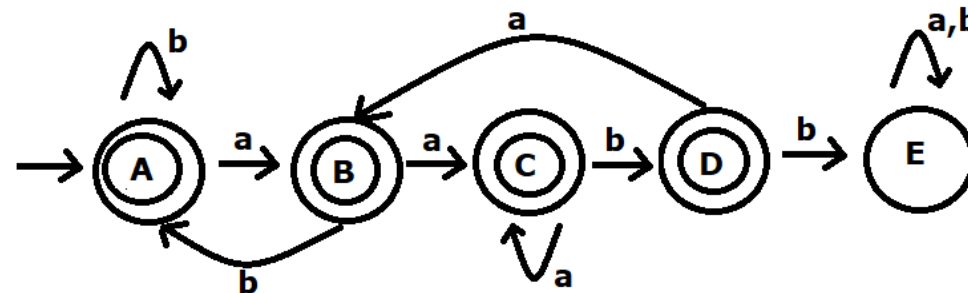
# DFA – örnek 3

$\Sigma = \{a, b\}$  üzerinde aabb içermeyen bütün stringleri kabul eden DFA tasarlayınız.

Daha basit olarak ilk başta aabb içeren DFA'yı tasarlayalım



Sonrasında son durumların tam terslerini alalım





# DETERMINISTIC FINITE AUTOMATA (DFA)'nın indirgenmesi -1

---

DR. ŞAFAK KAYIKÇI

# DFA indirgenmesi

---

DFA'nın minimum sayıda durumla ifade edilecek hale getirilmesidir.



$$\delta(A, x) \rightarrow F \text{ ve } \delta(B, x) \rightarrow F$$

veya

$$\delta(A, x) \nrightarrow F \text{ ve } \delta(B, x) \nrightarrow F$$

$$|x| = 0 \rightarrow A \text{ ve } B \text{ '0' denklik}$$

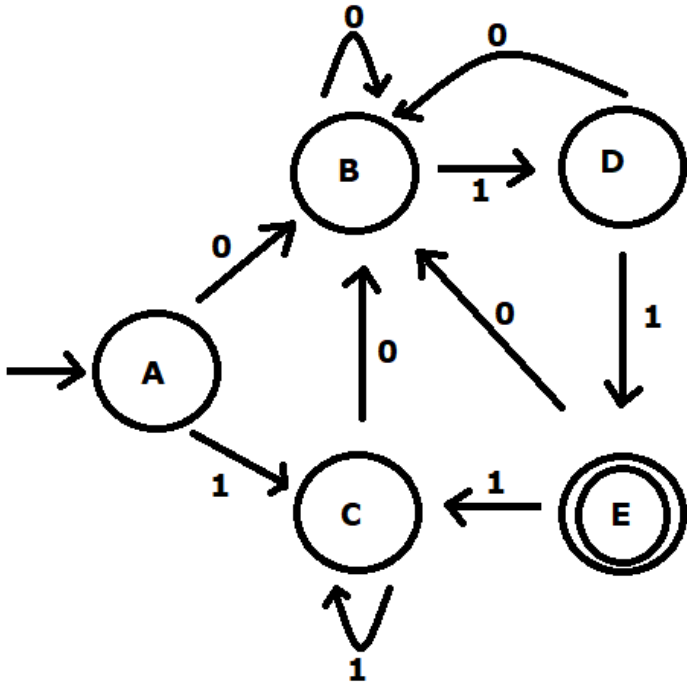
$$|x| = 1 \rightarrow A \text{ ve } B \text{ '1' denklik}$$

.....

$$|n| = n \rightarrow A \text{ ve } B \text{ 'n' denklik}$$

# Örnek

Aşağıdaki DFA'yı indirgeyiniz?



	0	1
→A	B	C
B	B	D
C	B	C
D	B	E
E	B	C

5 durum

# Örnek - çözüm

	0	1
→A	B	C
B	B	D
C	B	C
D	B	E
E	B	C

	0	1
→AC	B	AC
B	B	D
D	B	E
E	B	AC

0 denklik = son durum olanları bir, olmayanları bir grupluyoruz  
 $\{A,B,C,D\} \{E\}$

İkişer kontrol ediyoruz.

Çıktılar aynı ise sorun yok.

Çıktılar farklı ama bir önceki denklikte aynı kümede ise yine sorun yok.

A,B ✓    A,C ✓    C,D ✗

1 denklik =  $\{A,B,C\} \{D\} \{E\}$

2 denklik =  $\{A,C\} \{B\} \{D\} \{E\}$

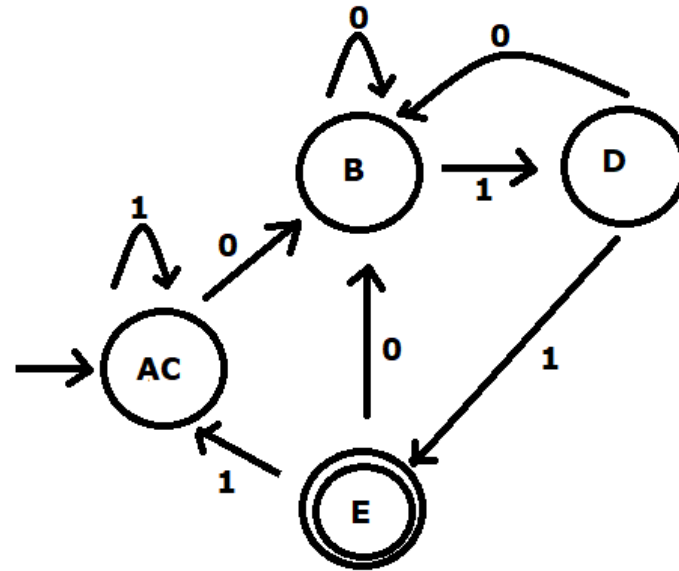
3 denklik =  $\{A,C\} \{B\} \{D\} \{E\}$

2. ve 3. derece aynı olduğunda işlemi bitirebiliriz.

# Örnek - çözüm

	0	1
→AC	B	AC
B	B	D
D	B	E
E	B	AC

4 durum



# Örnek

	0	1
→q0	q1	q5
q1	q6	q2
q2	q0	q2
q3	q2	q6
q4	q7	q5
q5	q2	q6
q6	q6	q4
q7	q6	q2

8 durum

0 denklik = {q0,q1,q3,q4,q5,q6,q7} {q2}

1 denklik = {q0, q4,q6} {q1,q7} {q2} {q3,q5}

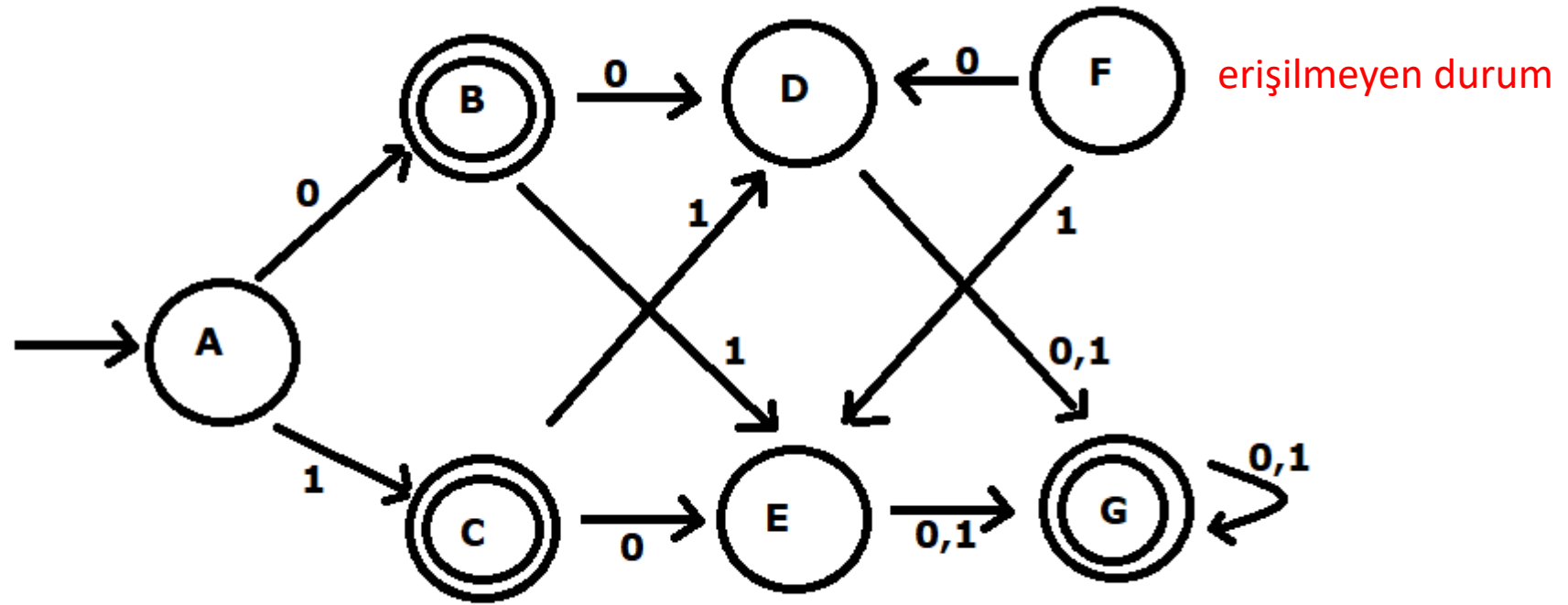
2 denklik = {q0, q4} {q6} {q1,q7} {q2} {q3,q5}

3 denklik = {q0, q4} {q6} {q1,q7} {q2} {q3,q5}

	0	1
→{q0,q4}	{q1,q7}	{q3,q5}
{q6}	{q6}	{q0,q4}
{q1,q7}	{q6}	{q2}
{q2}	{q2}	{q6}
{q3,q5}	{q0,q4}	{q2}

5 durum

# Örnek – (erişilmeyen durum ?)



# Çözüm

Erişilmeyen durum varsa onu çıkarıp yapıyoruz.

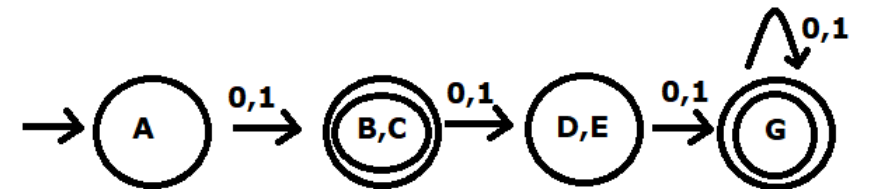
	0	1
→A	B	C
B	D	E
C	E	D
D	G	G
E	G	G
G	G	G

6 durum

0 denklik = {A,D,E} {B,C,G}  
1 denklik = {A,D,E} {B,C} {G}  
2 denklik = {A} {D,E} {B,C} {G}  
3 denklik = {A} {D,E} {B,C} {G}

	0	1
→{A}	{B,C}	{B,C}
{D,E}	{G}	{G}
{B,C}	{D,E}	{D,E}
{G}	{G}	{G}

4 durum





# DETERMINISTIC FINITE AUTOMATA (DFA)'nın indirgenmesi -2

---

DR. ŞAFAK KAYIKÇI

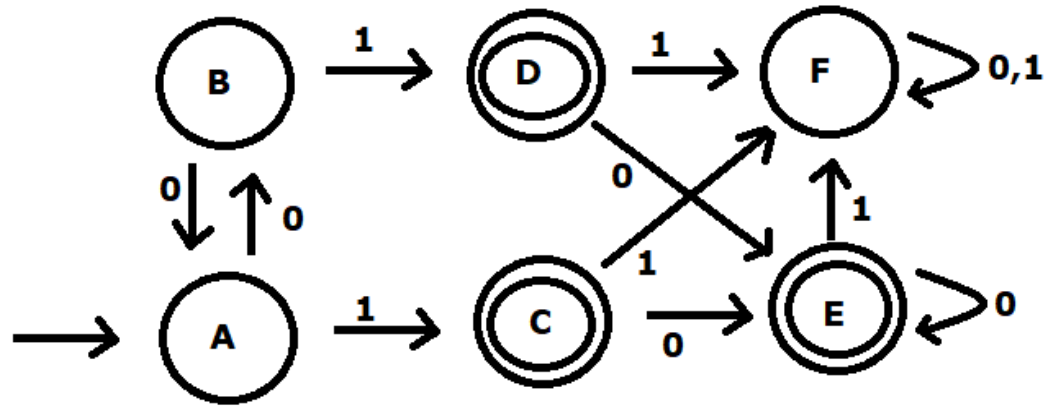
# Myhill – Nerode Teoremi (Doldurma Metodu)

---

Adımlar :

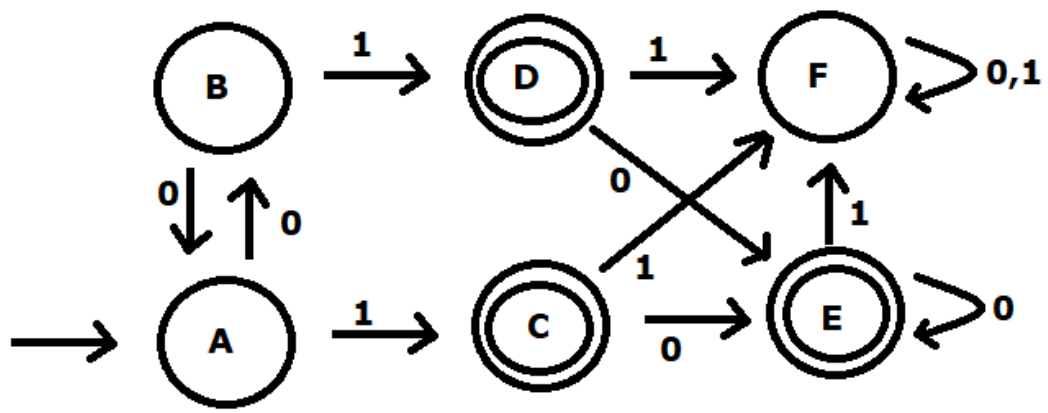
1. Her  $(P,Q)$  çifti için durumlar tablosu oluşturulur.
2.  $P \in F$  ve  $Q \notin F$  olan çiftler işaretlenir.
3. İşaretlenmemiş  $(P,Q)$  çiftleri arasında  $[\delta(P, x), \delta(Q, x)]$  işaretlenmiş ise bunlarda işaretlenir.
4. Daha fazla işlem yapılamayacak duruma kadar döngü devam eder.
5. İşaretlenmemiş bütün durumlar birleştirilip tek bir durum olarak gösterilir.

# Örnek -1



	A	B	C	D	E	F
A						
B						
C	$\sqrt{1}$	$\sqrt{1}$				
D	$\sqrt{1}$	$\sqrt{1}$				
E	$\sqrt{1}$	$\sqrt{1}$				
F			$\sqrt{1}$	$\sqrt{1}$	$\sqrt{1}$	

Çiftlerden biri son durum olup, diğeri olmayanlar işaretlenir.



	A	B	C	D	E	F
A						
B						
C	$\sqrt{1}$	$\sqrt{1}$				
D	$\sqrt{1}$	$\sqrt{1}$				
E	$\sqrt{1}$	$\sqrt{1}$				
F			$\sqrt{1}$	$\sqrt{1}$	$\sqrt{1}$	

(B,A) için

$\delta(B,0):A$

$\delta(A,0):B$

AB işaretli, bir şey yapma

$\delta(B,1):D$

$\delta(A,1):C$

DC işaretli, bir şey yapma

(D,C) için

$\delta(D,0):E$

$\delta(C,0):E$

EE tabloda yok, bir şey yapma

$\delta(D,1):F$

$\delta(C,1):F$

FF tabloda yok, bir şey yapma

(E,C) için

$\delta(E,0):E$

$\delta(C,0):E$

EE tabloda yok, bir şey yapma

$\delta(E,1):F$

$\delta(C,1):F$

FF tabloda yok, bir şey yapma

(E,D) için

$\delta(E,0):E$

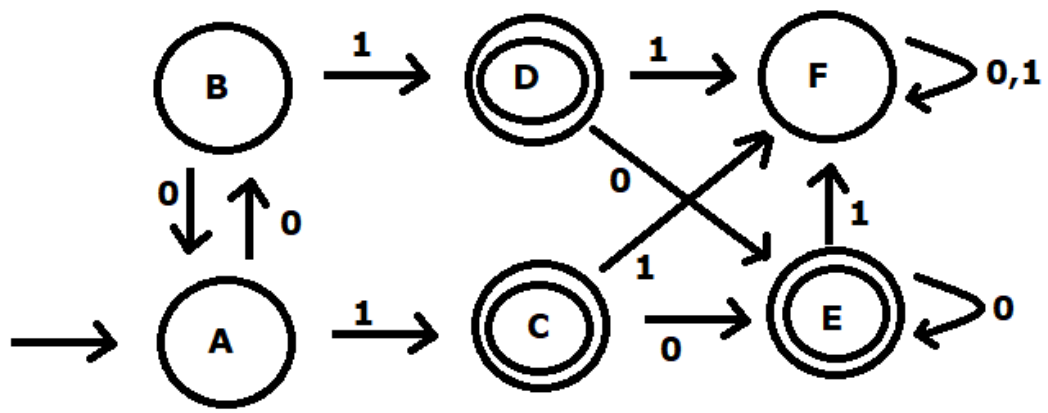
$\delta(D,0):E$

EE tabloda yok, bir şey yapma

$\delta(E,1):F$

$\delta(D,1):F$

FF tabloda yok, bir şey yapma



	A	B	C	D	E	F
A						
B						
C	$\sqrt{1}$	$\sqrt{1}$				
D	$\sqrt{1}$	$\sqrt{1}$				
E	$\sqrt{1}$	$\sqrt{1}$				
F			$\sqrt{1}$	$\sqrt{1}$	$\sqrt{1}$	

(F,A) için

$\delta(F,0) : F$

$\delta(A,0) : B$

FB işaretli, bir şey yapma

$\delta(F,1) : F$

$\delta(A,1) : C$

FC işaretli, dolayısıyla FA işaretliyoruz

(F,B) için

$\delta(F,0) : F$

$\delta(B,0) : A$

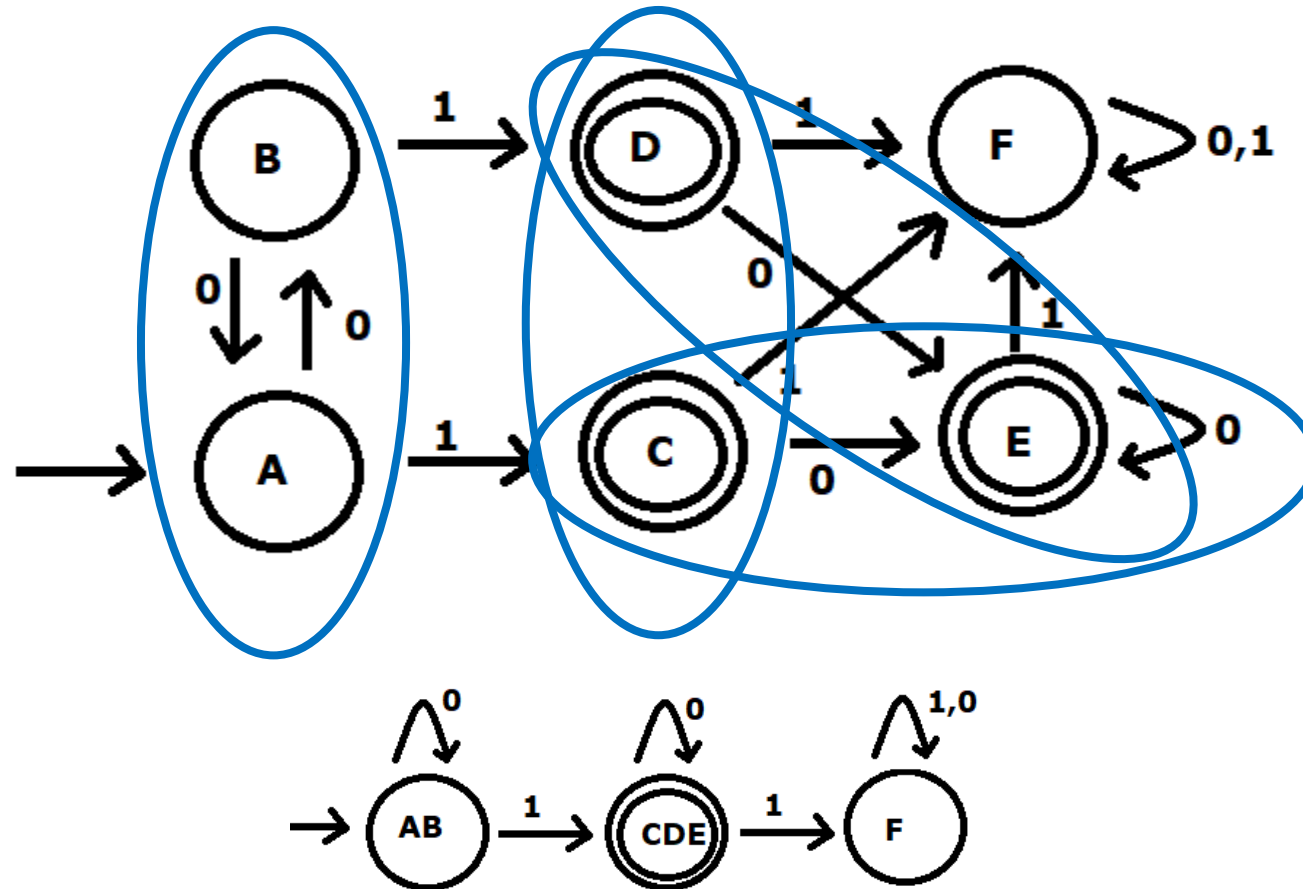
FA tabloda işaretli, dolayısıyla FB işaretliyoruz

$\delta(F,1) : F$

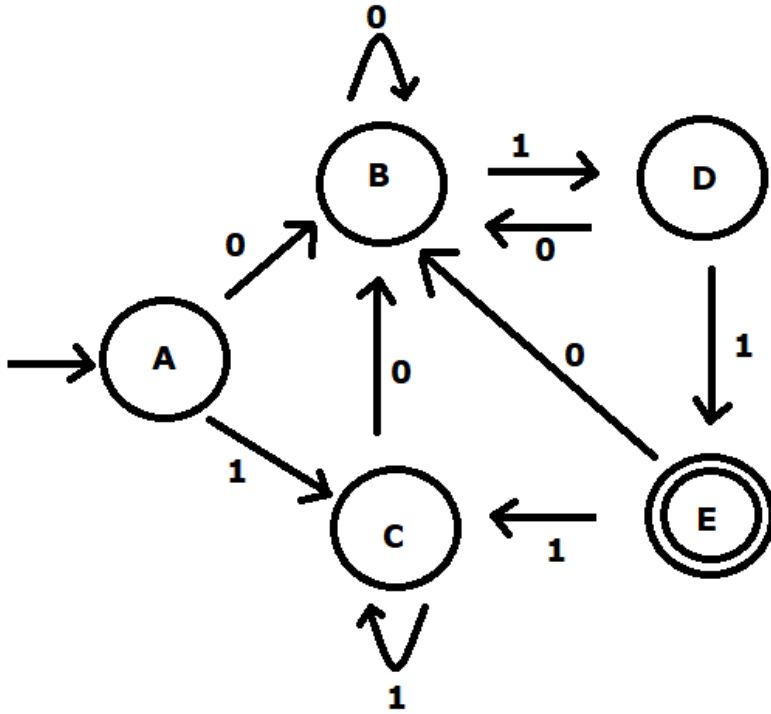
$\delta(B,1) : D$

	A	B	C	D	E	F
A						
B						
C	$\sqrt{1}$	$\sqrt{1}$				
D	$\sqrt{1}$	$\sqrt{1}$				
E	$\sqrt{1}$	$\sqrt{1}$				
F	$\sqrt{2}$	$\sqrt{2}$	$\sqrt{1}$	$\sqrt{1}$	$\sqrt{1}$	

İşaretlenmemiş olan ikilileri birleştiriyoruz. (A,B) (D,C) (E,C) (E,D)

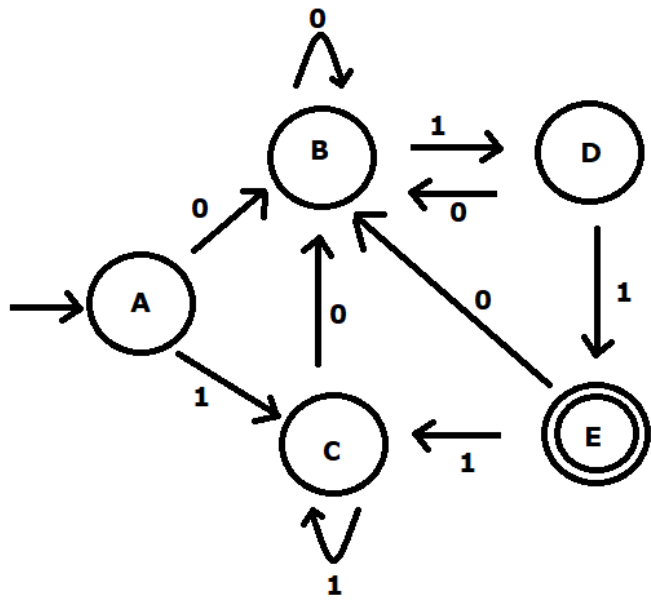


## Örnek 2



	A	B	C	D	E
A					
B					
C					
D					
E	$\sqrt{1}$	$\sqrt{1}$	$\sqrt{1}$	$\sqrt{1}$	

Çiftlerden biri son durum olup, diğeri olmayanlar işaretlenir.



	A	B	C	D	E
A					
B					
C					
D	$\sqrt{2}$				
E	$\sqrt{1}$	$\sqrt{1}$	$\sqrt{1}$	$\sqrt{1}$	

(B,A) için

$\delta(B,0):B$   
 $\delta(A,0):B$

BB tabloda yok, bir şey yapma

$\delta(B,1):D$   
 $\delta(A,1):C$

DC işaretli, bir şey yapma

(C,A) için

$\delta(C,0):B$   
 $\delta(A,0):B$

BB tabloda yok, bir şey yapma

$\delta(C,1):C$   
 $\delta(A,1):C$

CC tabloda yok, bir şey yapma

(C,B) için

$\delta(C,0):B$   
 $\delta(B,0):B$

BB tabloda yok, bir şey yapma

$\delta(C,1):C$   
 $\delta(B,1):D$

DC işaretli, bir şey yapma

(A,D) için

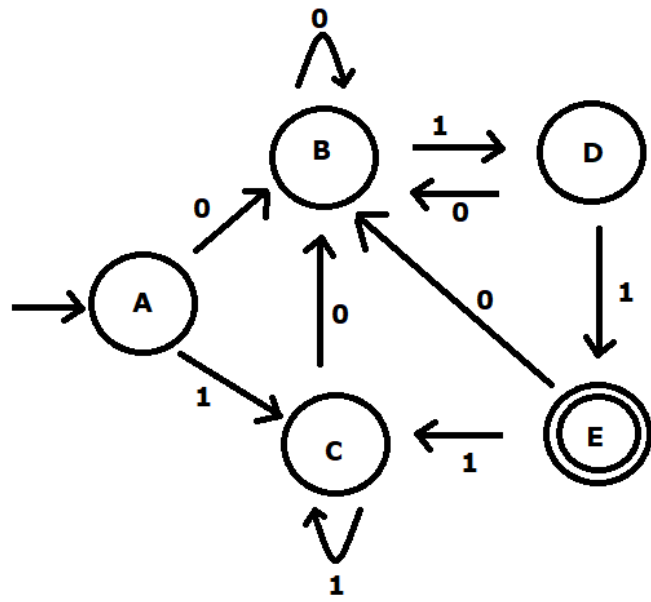
$\delta(A,0):B$   
 $\delta(D,0):B$

BB tabloda yok, bir şey yapma

$\delta(A,1):C$   
 $\delta(D,1):E$

CE işaretli, dolayısıyla AD işaretliyoruz





	A	B	C	D	E
A					
B					
C					
D	$\sqrt{2}$	$\sqrt{2}$	$\sqrt{2}$		
E	$\sqrt{1}$	$\sqrt{1}$	$\sqrt{1}$	$\sqrt{1}$	

(D,B) için

$\delta(D,0):B$

$\delta(B,0):B$

BB tabloda yok, bir şey yapma

$\delta(D,1):E$

$\delta(B,1):D$

ED işaretli, dolayısıyla DB işaretliyoruz

(D,C) için

$\delta(D,0):B$

$\delta(C,0):B$

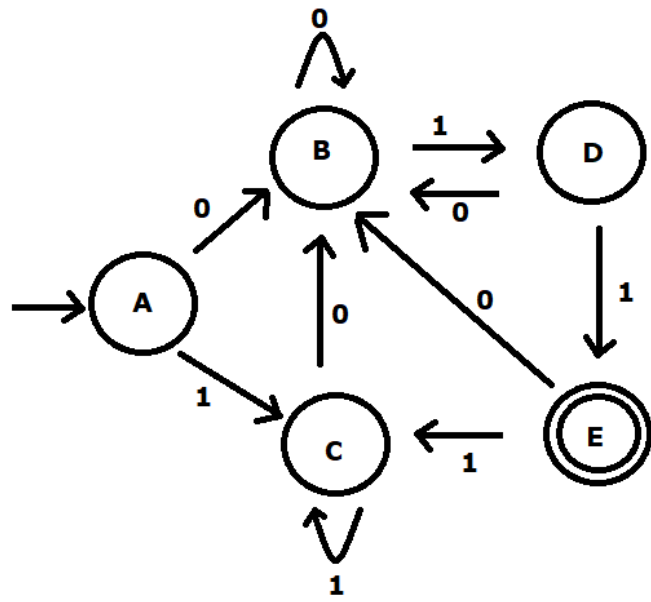
BB tabloda yok, bir şey yapma

$\delta(D,1):E$

$\delta(C,1):C$

EC işaretli, dolayısıyla DC işaretliyoruz

1. TUR BİTTİ, 2. TURA BAŞLAYACAĞIZ



	A	B	C	D	E
A					
B	$\sqrt{3}$				
C		$\sqrt{3}$			
D	$\sqrt{2}$	$\sqrt{2}$	$\sqrt{2}$		
E	$\sqrt{1}$	$\sqrt{1}$	$\sqrt{1}$	$\sqrt{1}$	

(A,B) için

$\delta(A,0):B$

$\delta(B,0):B$

BB tabloda yok, bir şey yapma

$\delta(A,1):C$

$\delta(B,1):D$

CD bu sefer işaretli, dolayısıyla AB işaretliyoruz

(A,C) için

$\delta(A,0):B$

$\delta(C,0):B$

BB tabloda yok, bir şey yapma

$\delta(A,1):C$

$\delta(C,1):C$

CC tabloda yok, bir şey yapma

(B,C) için

$\delta(B,0):B$

$\delta(C,0):B$

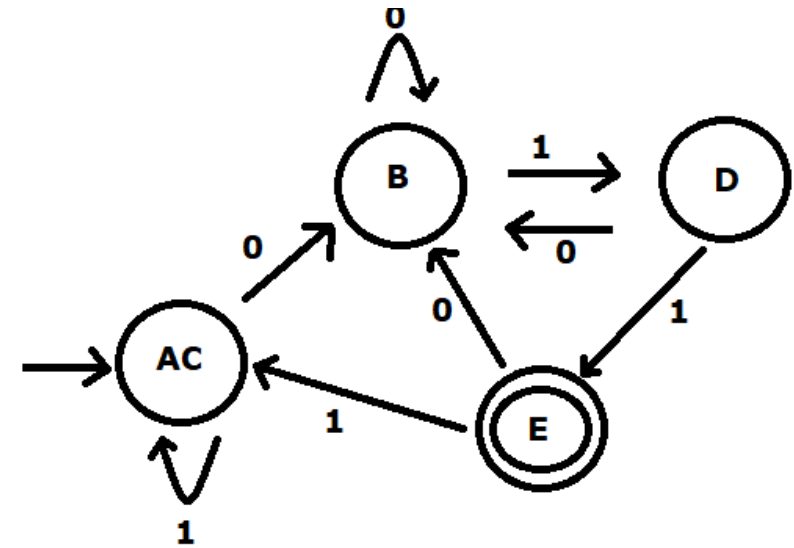
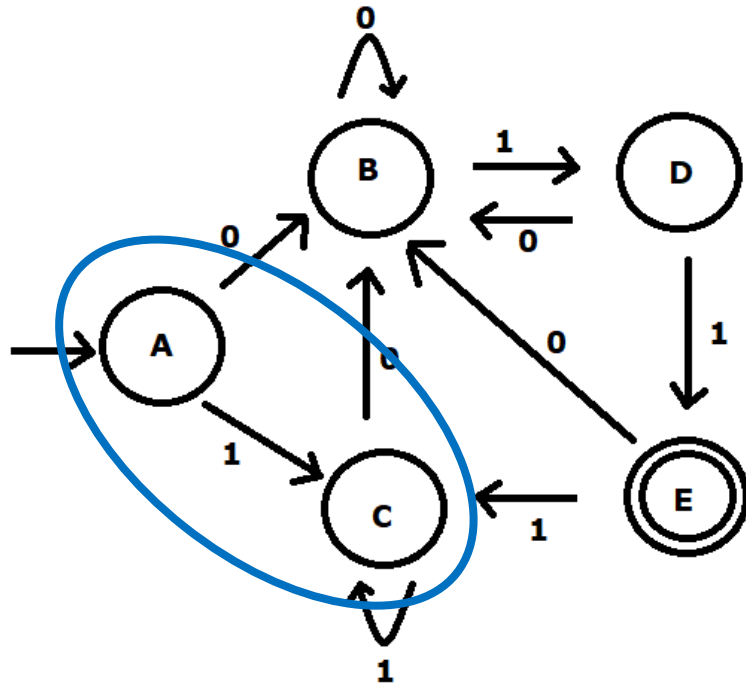
BB tabloda yok, bir şey yapma

$\delta(B,1):D$

$\delta(C,1):C$

CD bu sefer işaretli, dolayısıyla BC işaretliyoruz

İşaretlenmemiş olan ikilileri birleştiriyoruz. (A,C) B D E



# Epsilon NFA

---

DR. ŞAFAK KAYIKÇI

# $\epsilon$ - NFA

$\epsilon \rightarrow$  boş sembol ifadeleri gösterir ( $\lambda$  ile de gösterilebilir)

NFA =  $(Q, \Sigma, q_0, F, \delta)$

$Q$  = Tüm durumlar kümesi =  $\{A, B\}$

$\Sigma$  = giriş (input) alfabesi =  $\{0, 1\}$

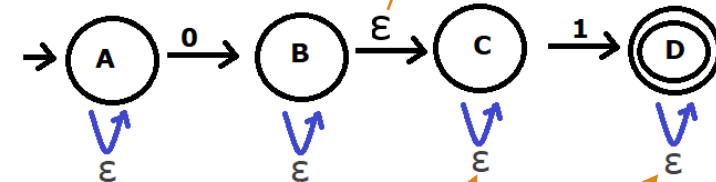
$q_0$  = başlangıç durumu =  $\{A\}$

$F$  = son durumlar kümesi =  $\{B\}$

$\delta = Q \times \Sigma \rightarrow 2^Q$

$$\delta = Q \times \Sigma \cup \epsilon \rightarrow 2^Q$$

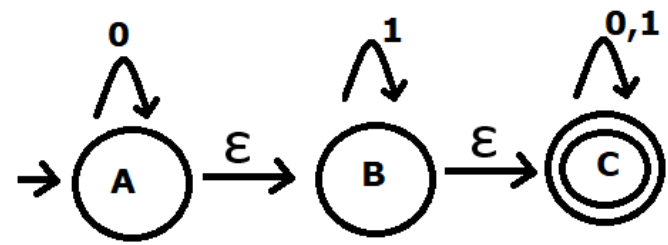
birleşim  $\epsilon$



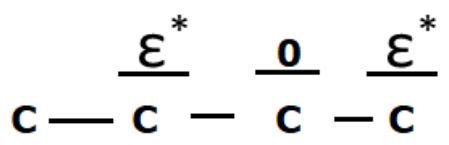
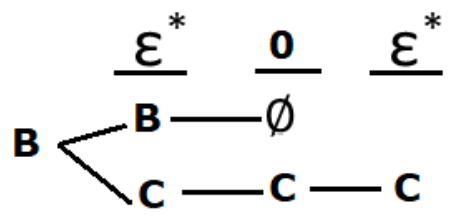
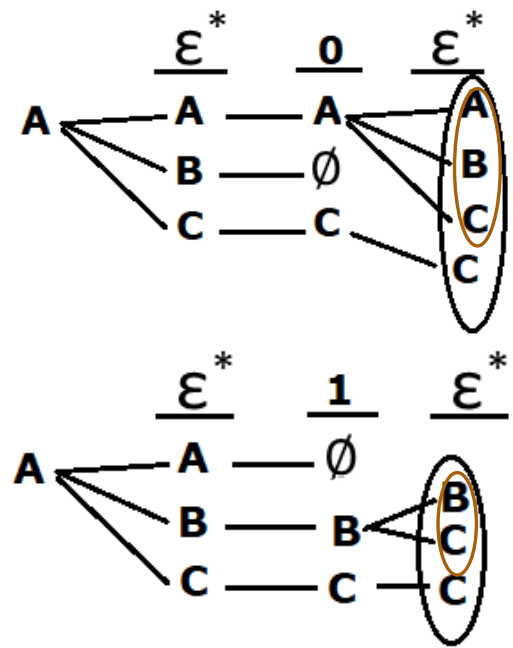
B hiçbir input gelmediği zaman C'ye gider

$\epsilon$  üzerindeki her durum kendisine gider

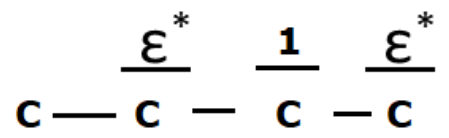
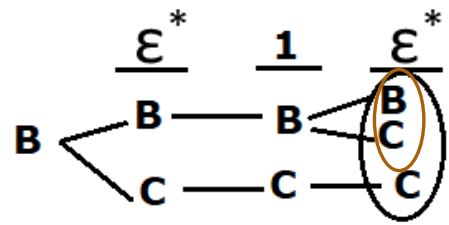
Örnek 1



$\epsilon^*$  : Bir durum üzerinden sadece  $\epsilon$  ile erişilebilen tüm durumlardır.



	0	1
A	{A,B,C}	{B,C}
B	{C}	{B,C}
C	{C}	{C}

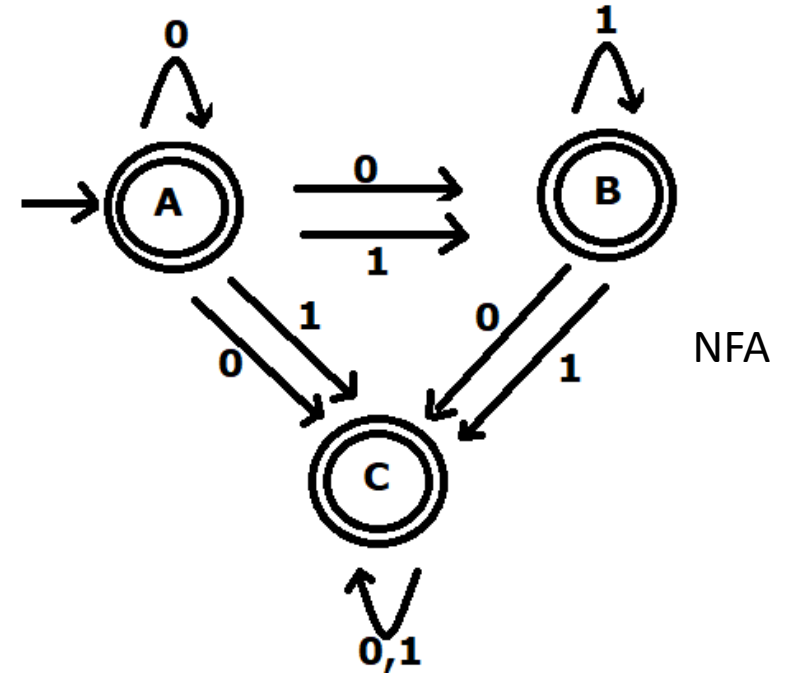


Son durumlar hesaplanırken,  
C zaten son durum. C'ye  $\epsilon$  üzerinden erişebilenlerde son durum olur (A,B)

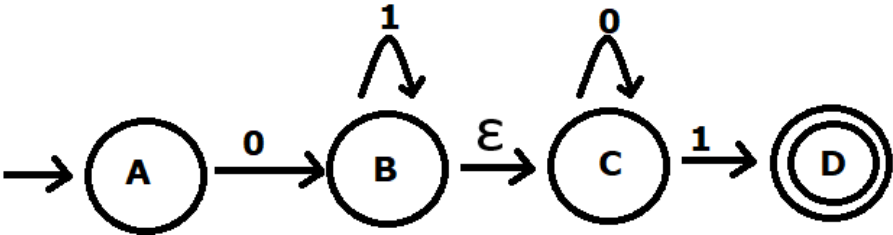
## $\epsilon$ 'dan – NFA çevirimi

	0	1
A	{A,B,C}	{B,C}
B	{C}	{B,C}
C	{C}	{C}

Son durumlar hesaplanırken,  
C zaten son durum. C'ye  $\epsilon$  üzerinden  
erişebilenlerde son durum olur (A,B)



# Örnek 2

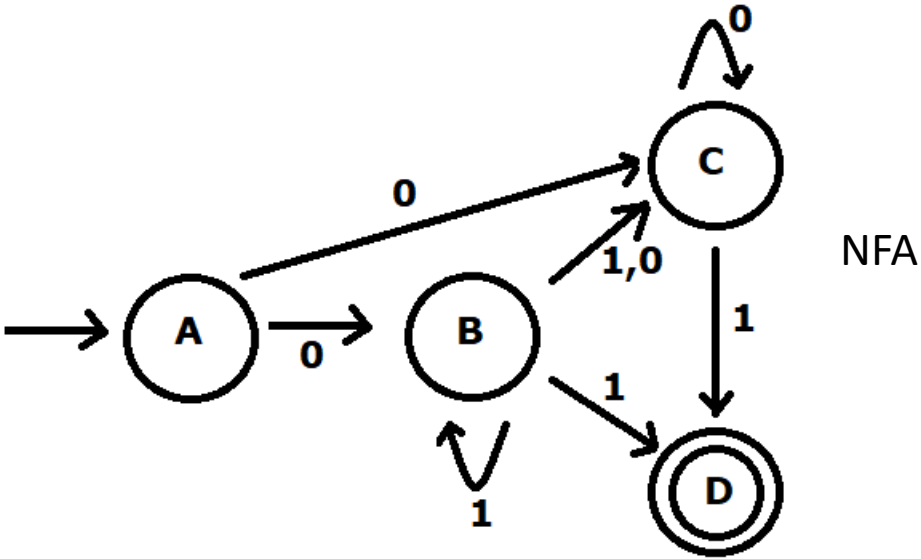


$\epsilon$  - NFA

	$\epsilon^*$	0	$\epsilon^*$
A	A	B	B C
B	B C	$\emptyset$ C	$\emptyset$ C
C	C	C	C
D	D	$\emptyset$	$\emptyset$

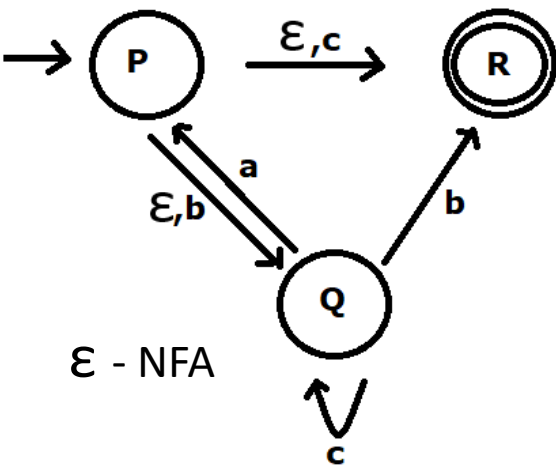
	$\epsilon^*$	1	$\epsilon^*$
A	A	$\emptyset$	$\emptyset$
B	B C	B D	B C D
C	C	D	D
D	D	$\emptyset$	$\emptyset$

	0	1
A	B,C	$\emptyset$
B	C	B,C,D
C	C	D
D	$\emptyset$	$\emptyset$





Örnek 3

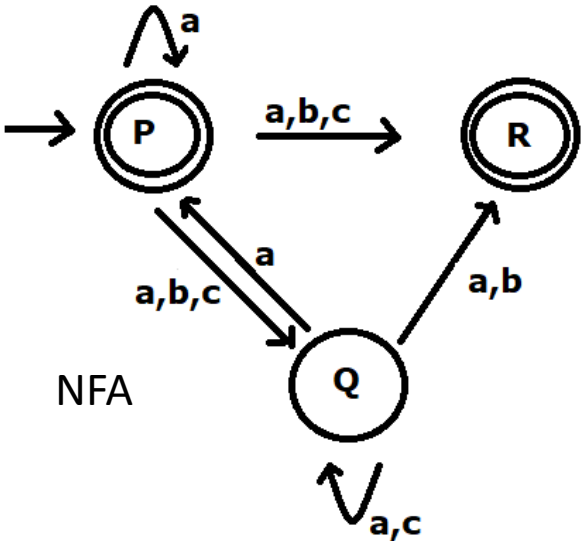


	$\epsilon^*$	a	$\epsilon^*$
P	P	$\emptyset$	$\emptyset$
	Q	P	P,Q,R
	R	$\emptyset$	$\emptyset$
Q	Q	P	P,Q,R
R	R	$\emptyset$	$\emptyset$

	$\epsilon^*$	b	$\epsilon^*$
P	P	Q	Q
	Q	R	R
	R	$\emptyset$	$\emptyset$
Q	Q	R	R
R	R	$\emptyset$	$\emptyset$

	$\epsilon^*$	c	$\epsilon^*$
P	P	R	R
	Q	Q	Q
	R	$\emptyset$	$\emptyset$
Q	Q	Q	Q
R	R	$\emptyset$	$\emptyset$

	a	b	c
P	P,Q,R	Q,R	R,Q
Q	P,Q,R	R	Q
R	$\emptyset$	$\emptyset$	$\emptyset$



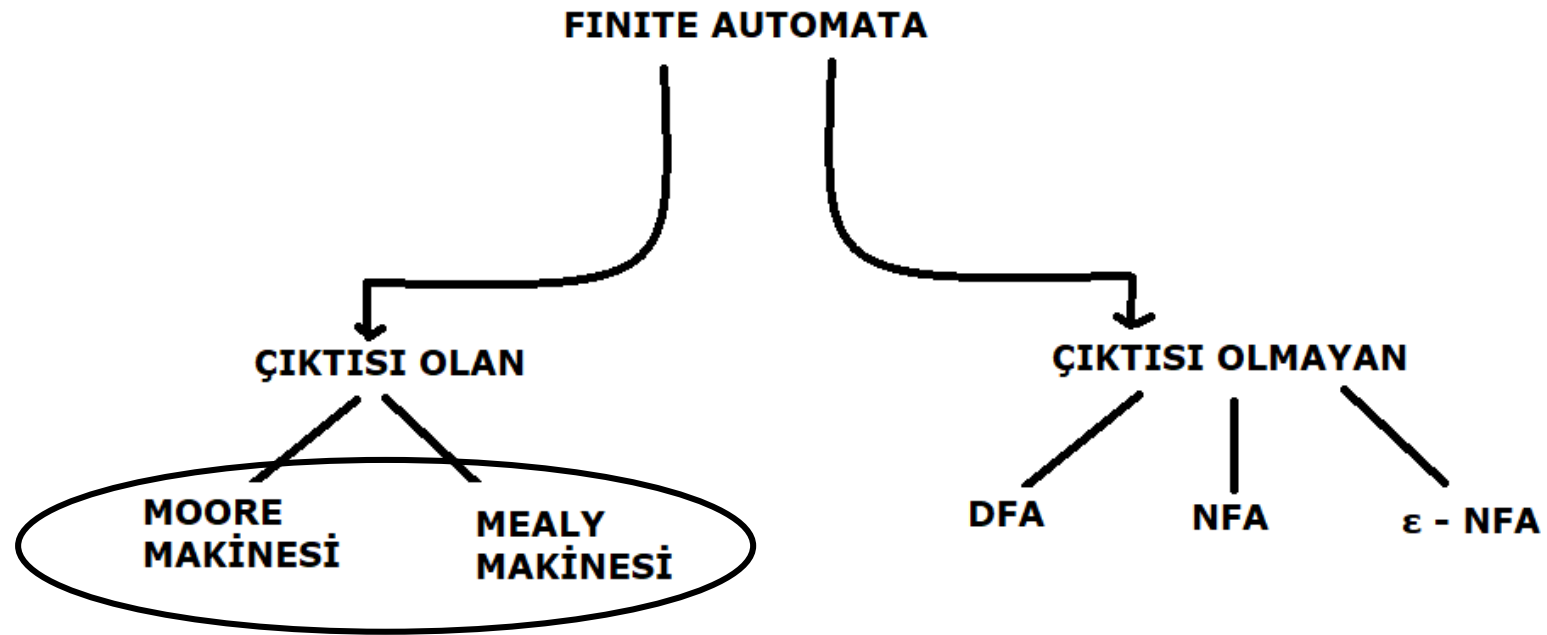
# Mealy – Moore Makineleri

---

DR. ŞAFAK KAYIKÇI

# Çıktısı olan Sonlu Durum Makineleri

---



# Mealy – Moore Makineleri

---

MEALY MAKİNESİ  
 $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$

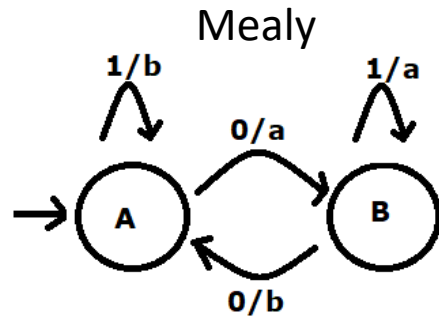
MOORE MAKİNESİ  
 $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$

$Q \times \Sigma \rightarrow \Delta$

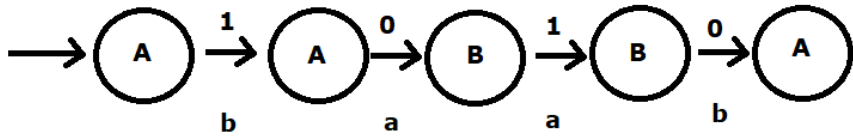
$Q$  = son durumlar kümesi  
 $\Sigma$  = sonlu boş olmayan giriş alfabe kümesi  
 $\Delta$  = çıkış alfabe kümesi  
 $\delta$  = geçiş fonksiyonu  $Q \times \Sigma \rightarrow Q$   
 $\lambda$  = çıkış fonksiyonu  
 $q_0$  = başlangıç durumu

$Q \rightarrow \Delta$

# Mealy – Moore Makineleri

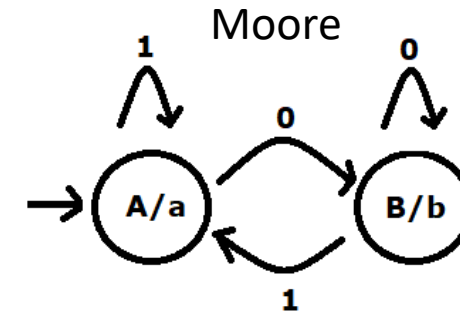


örn : 1 0 1 0

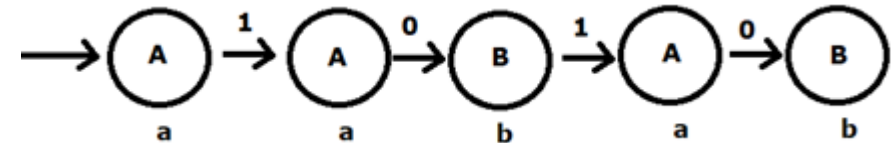


baab  
 $n \rightarrow n$

giriş uzunluğu      çıkış uzunluğu



örn : 1 0 1 0



aabab  
 $n \rightarrow n+1$

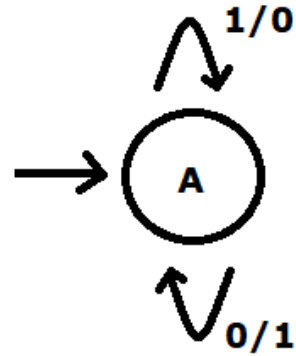
giriş uzunluğu      çıkış uzunluğu  
(başlangıç durumundan dolayı)

# Mealy Makinesi

---

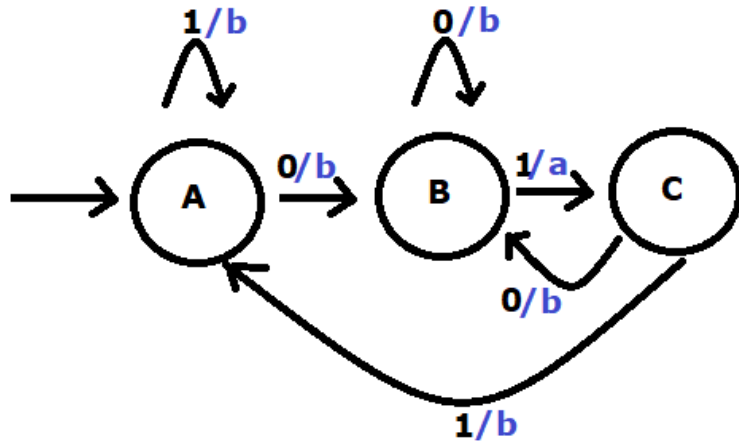
Örn : Herhangi bir binary input girişinin 1'e tümleyenini yapan mealy makinesini tasarlayınız.

1 0 1 0 0  
0 1 0 1 1     1'e tümleme



# Mealy Makinesi

Örn: içerisinde '01' geçen herhangi bir binary girişte, 'a' çıktısı veren makineyi tasarlayın



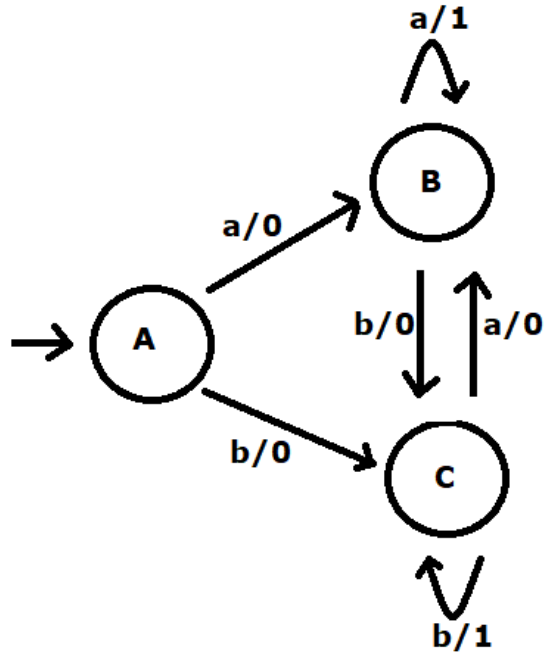
ilk başta DFA gibi tasarlıyoruz.

0110 → babb

1000 → bbbb

# Mealy Makinesi

Örn:  $\Sigma = \{a,b\}$  alfabesinde 'aa' veya 'bb' içeren giriş geldiğinde 1 basan makineyi tasarlayın.



aaa → 011  
aab → 010



# Mealy Makinesi

Örn : Herhangi bir binary girişin 2'ye tümleyenini yapan makineyi yapınız.

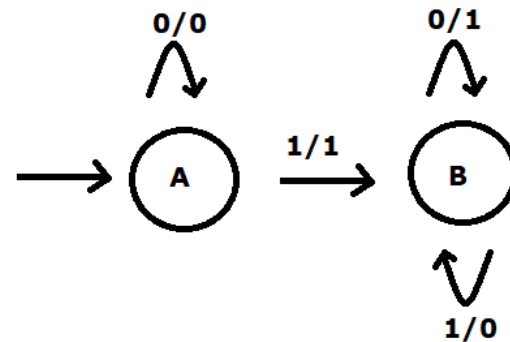
$$2\text{'ye tümleyen} = 1\text{'e tümleyen} + 1$$

\* sondan başlayarak ilk 1 görene kadar aynı, 1'i gördükten sonra tersini alınız

1 0 1 0 0  $\rightarrow$  0 1 1 0 0

1 1 1 0 0  $\rightarrow$  0 0 1 0 0

1 1 1 1 1  $\rightarrow$  0 0 0 0 1

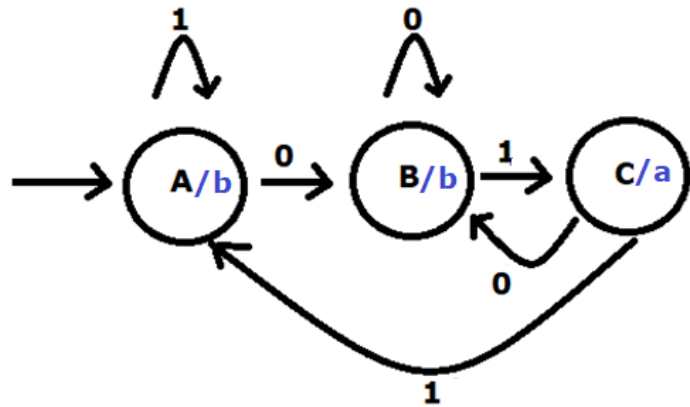


1	0	1	0	0
B	B	A	A	A
0	1	1	0	0

\* sondan başlayarak düşünerek

# Moore Makinesi

Örn: içerisinde '01' geçen herhangi bir binary girişte, 'a' çıktısı veren makineyi tasarlayın



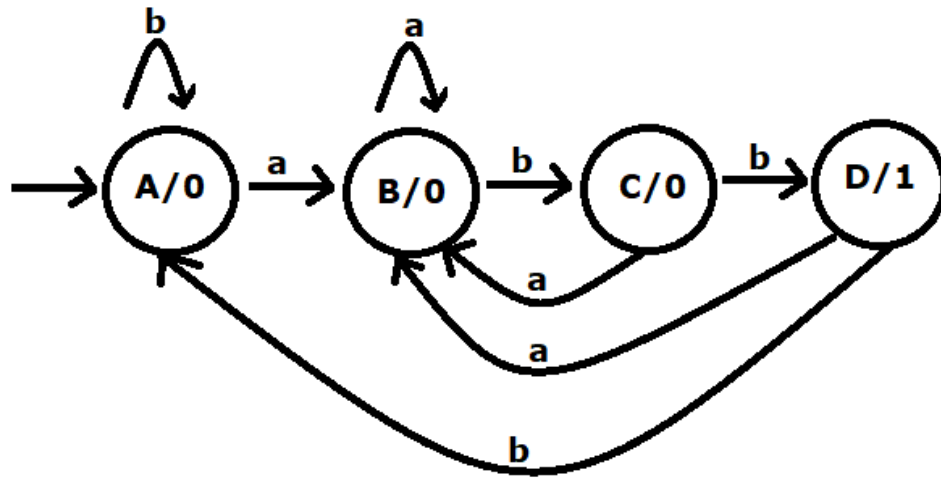
ilk başta DFA gibi tasarlıyoruz.

giriş	→	0	1	1	0	
durum	→	A	B	C	A	B
çıkı	→	b	b	a	b	b

giriş	→	0	1	0	1	
durum	→	A	B	C	B	C
çıkı	→	b	b	a	b	a

# Moore Makinesi

Örn :  $\Sigma=\{a,b\}$  giriş alfabesinde 'abb' sıralamasının kaç defa geçtiğini sayan Moore makinesini tasarlayın.  $\Delta=\{0,1\}$



ilk başta DFA gibi tasarlıyoruz.

# Moore Makinesi

Aşağıda geçiş tablosu ve çıktıları verilmiş Moore Makinesi için giriş alfabesi  $\Sigma=\{a,b\}$  ve çıkış alfabesi  $\Delta=\{0,1\}$  'dir. Verilen giriş sıralamalarına göre çıktıları yazınız.

durumlar	a	b	çıktılar
→q0	q1	q2	0
q1	q2	q3	0
q2	q3	q4	1
q3	q4	q4	0
q4	q0	q0	0

	a	a	b	a	b
q0	q1	q2	q4	q0	q2
0	0	1	0	0	1

---

	a	b	b	b
q0	q1	q3	q4	q0
0	0	0	0	0

---

	a	b	a	b	b
q0	q1	q3	q4	q0	q2
0	0	0	0	0	1

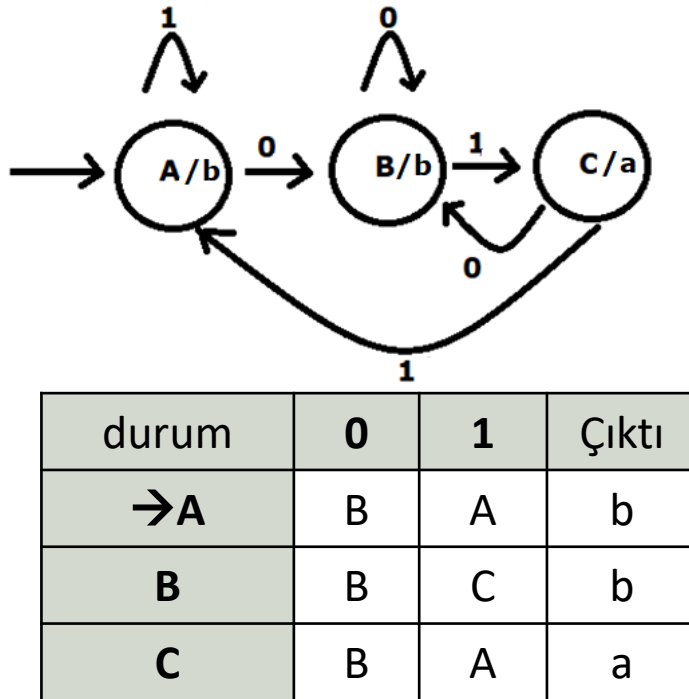
# Mealy $\leftrightarrow$ Moore Çevirimleri

---

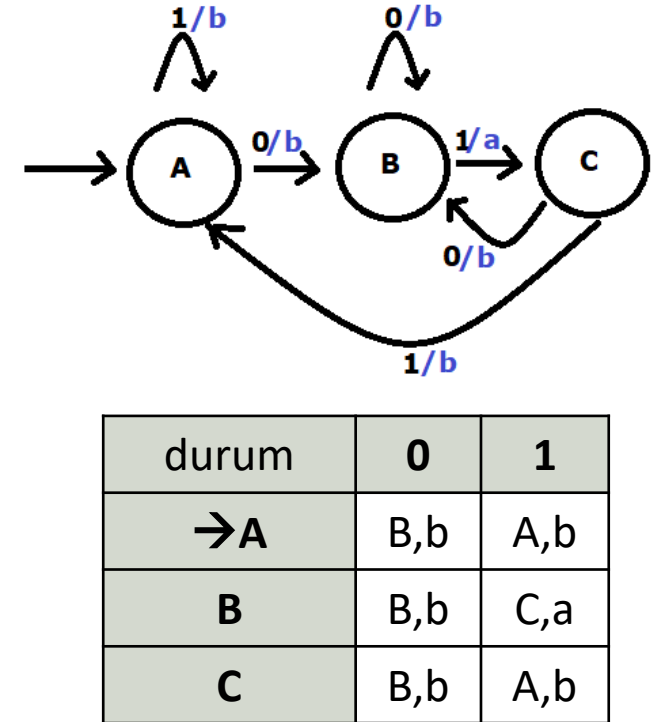
DR. ŞAFAK KAYIKÇI

# Moore → Mealy çevirimi

içerisinde '01' geçen herhangi bir binary girişte, 'a' çıktısı veren Moore makinesini eşdeğeri olan Mealy Makinesine çevirin.

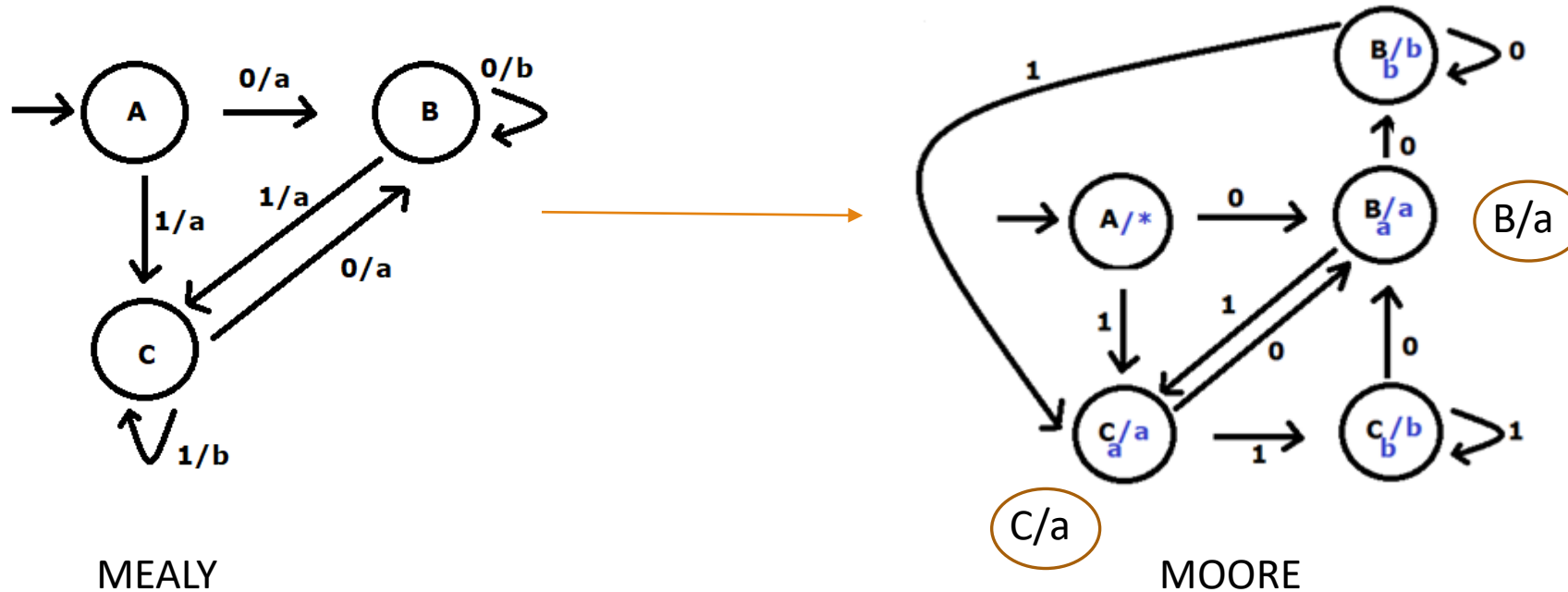


gittiği durumdaki  
output değerini  
geçiş yazıyoruz.



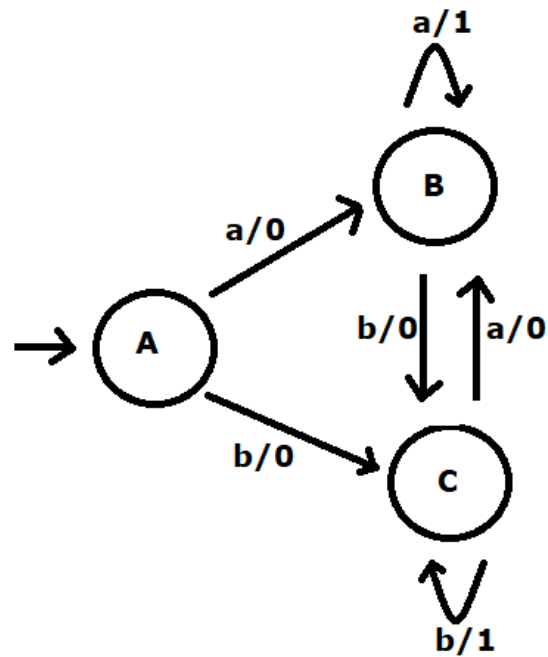
# Mealy $\rightarrow$ Moore Çevirimi

Aşağıdaki Mealy Makinesini Moore makinesine çeviriniz.  $\Sigma=\{0,1\}$   $\Delta=\{a,b\}$

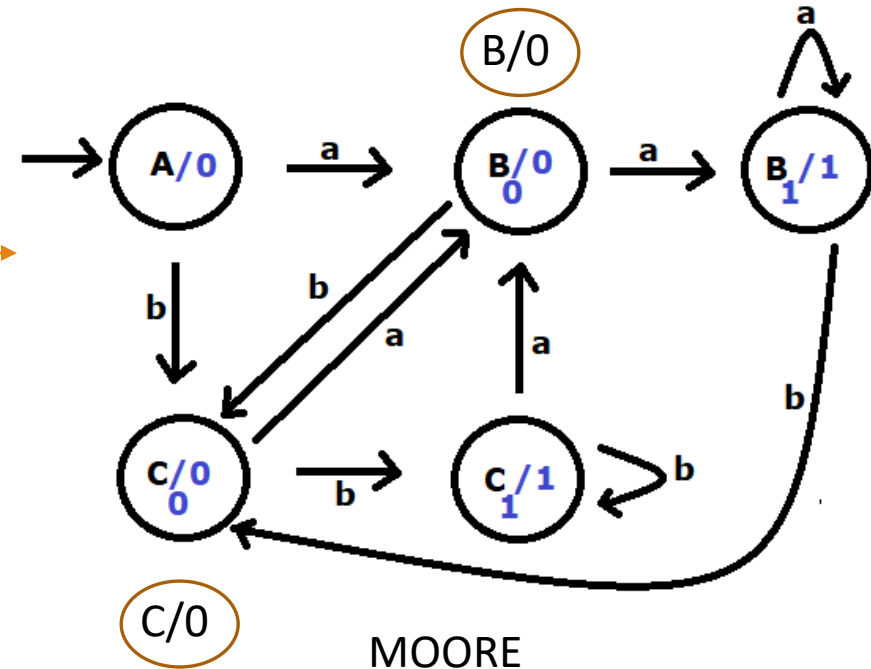


# Mealy $\rightarrow$ Moore Çevirimi

Örn:  $\Sigma = \{a,b\}$  alfabesinde 'aa' veya 'bb' içeren giriş geldiğinde 1 basan Mealy Makinesini Moore Makinesine çevirin.  $\Delta = \{0,1\}$



MEALY

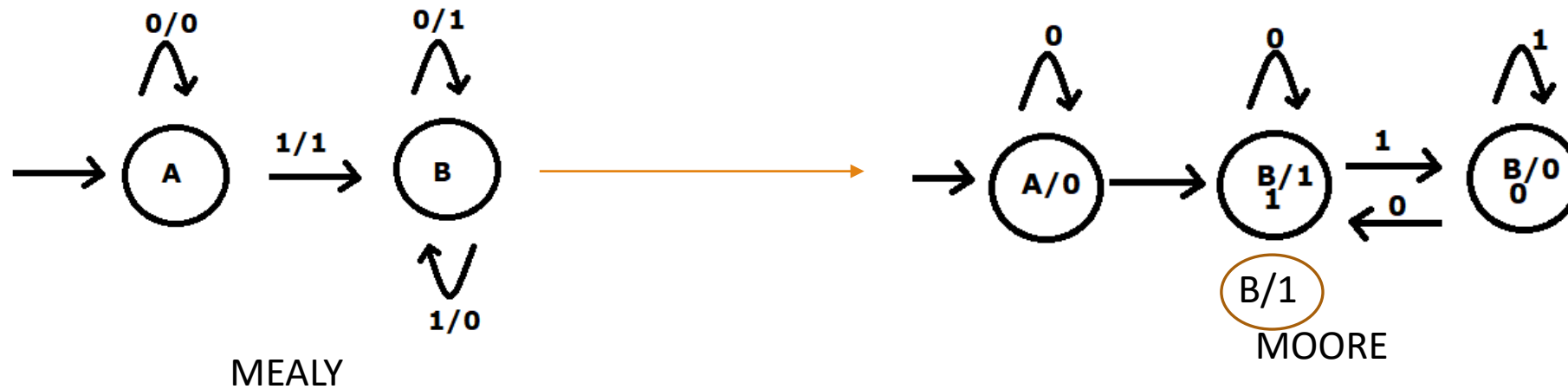


MOORE



# Mealy $\rightarrow$ Moore Çevirimi

Herhangi bir binary girişin 2'ye tümleyenini yapan Mealy makinesini Moore makinesine çeviriniz.



# Düzenli İfadeleri (Regex)

---

DR. ŞAFAK KAYIKÇI

# Düzenli İfadeler (Regular Expressions)

---

Düzenli ifadeler (regular expressions) programlama dillerinin temel bir parçası olup, bütün dillerde aynı söz dizimi (syntax) geçerlidir. Genel olarak, bir string'in bazı özel karakterler kullanılarak kısa yoldan ve esnek biçimde tanımlanmasını sağlar.

Düzenli ifadeler sayesinde doğrulama, arama yapma ve yer değiştirme gibi işlemler kolayca yapılabilir.

Düzenli ifadeler tarafından kabul edilen dillere düzenli diller (regular languages) denir.

# Düzenli İfadeler

**Örnek 1 :**  $\{0, 1, 2\}$  0 veya 1 veya 2

$$R = 0 + 1 + 2$$

$$R = 0 \mid 1 \mid 2$$


**Örnek 2:**  $\{abb, a, b, bba\}$

$$R = abb + a + b + bba$$

**Örnek 3:**  $\{\epsilon, 0, 00, 000, \dots\}$

$$R = 0^*$$

**Örnek 4:**  $\{1, 11, 111, 1111, \dots\}$

$$R = 1^+$$

$\Sigma = \{a, b\}$  kümesinde

**Örnek 5:** İki boyutlu stringleri kabul eden dil

$$L = \{aa, ab, ba, bb\}$$

$$R = aa + ab + ba + bb$$

$$= a(a+b) + b(a+b)$$

$$= (a+b)(a+b)$$

**Örnek 6:** en az iki boyutlu stringleri kabul eden dil

$$L = \{aa, ab, ba, bb, aaa, \dots\}$$

$$R = (a+b)(a+b)(a+b)^*$$

**Örnek 7:** en fazla iki boyutlu stringleri kabul eden dil

$$L = \{\epsilon, a, b, aa, ab, ba, bb\}$$

$$R = \epsilon + a + b + aa + ab + ba + bb$$

$$= (\epsilon + a + b)(\epsilon + a + b)$$

# Düzenli İfadelerde Özellikler

---

$$1) \emptyset + R = R$$

$$2) \epsilon R = R \epsilon = R$$

$$3) \epsilon^* = \epsilon \text{ ve } \emptyset^* = \epsilon$$

$$4) R + R = R$$

$$5) R^* R^* = R^*$$

$$6) R R^* = R^* R = R^+$$

$$7) (R^*)^* = R^*$$

$$8) \epsilon + R R^* = \epsilon + R^* R = R^*$$

$$9) (PQ)^* P = P(QP)^*$$

$$10) (P+Q)^* = (P^* Q^*)^* = (P^* + Q^*)^*$$

$$11) (P+Q)R = PR + QR$$

$$R(P+Q) = RP + RQ$$

$$12) R = Q+RP \rightarrow R = QP^* \text{ (Arden Teoremi)}$$

ispat: R yerine denklemde Q+RP koyalım

$$R = Q + RP$$

$$R = Q + (Q+RP) P$$

$$R = Q + QP + RPP$$

$$= Q + QP + (Q+RP)PP$$

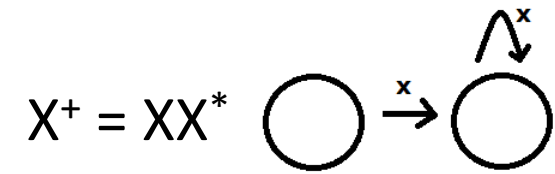
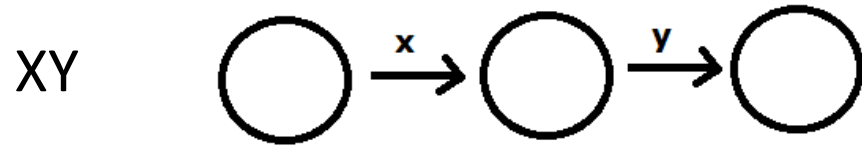
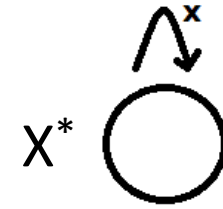
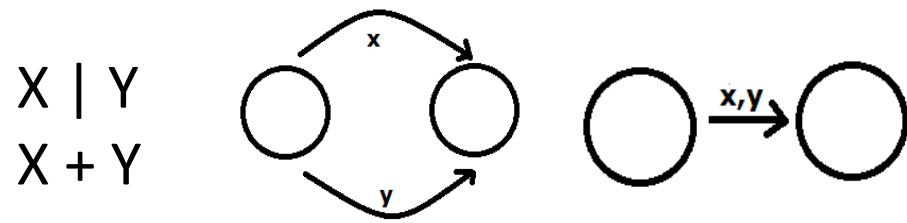
$$= Q + QP + QPP + RPPP$$

.....

$$= Q (\epsilon + P + PP + PPP + \dots)$$

$$= QP^*$$

# Düzenli İfadeden Sonlu Otomata Çevirim



# Örnekler

$ab^*a$

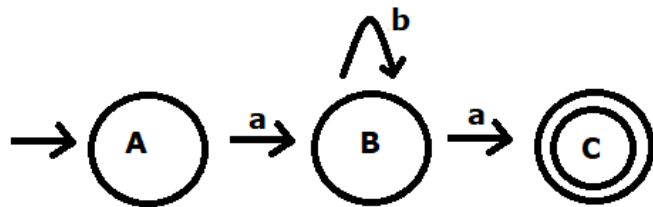
aa

aba

abba

abbba

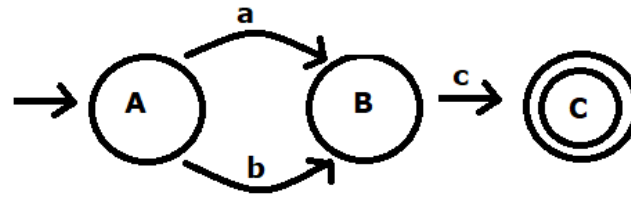
....



$(a \mid b)c$

ac

bc



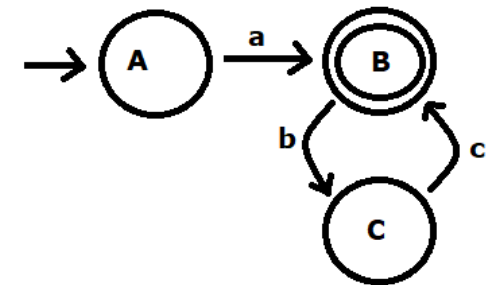
$a(bc)^*$

a

abc

abcbcb

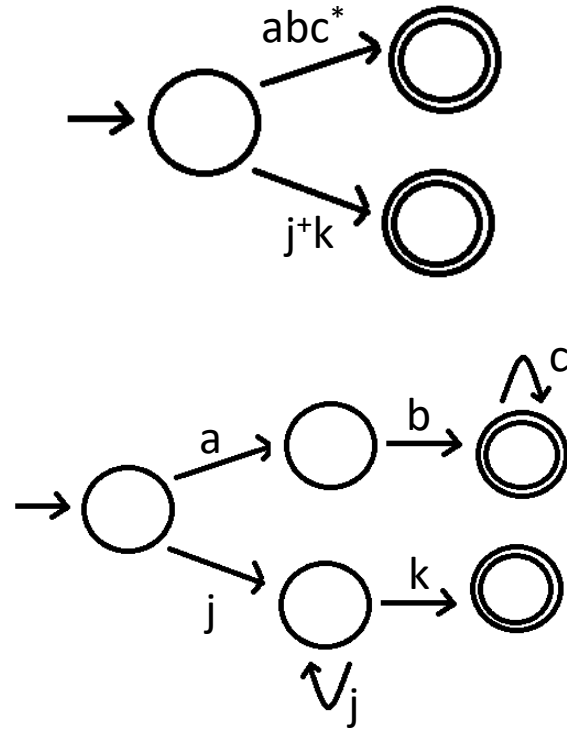
....



# Örnekler

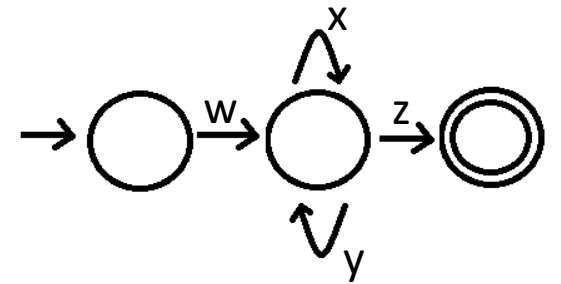
$abc^* \mid j^+k$

ab  
abc  
abcccc  
jk  
jjk  
jjjk



$w(x \mid y)^*z$

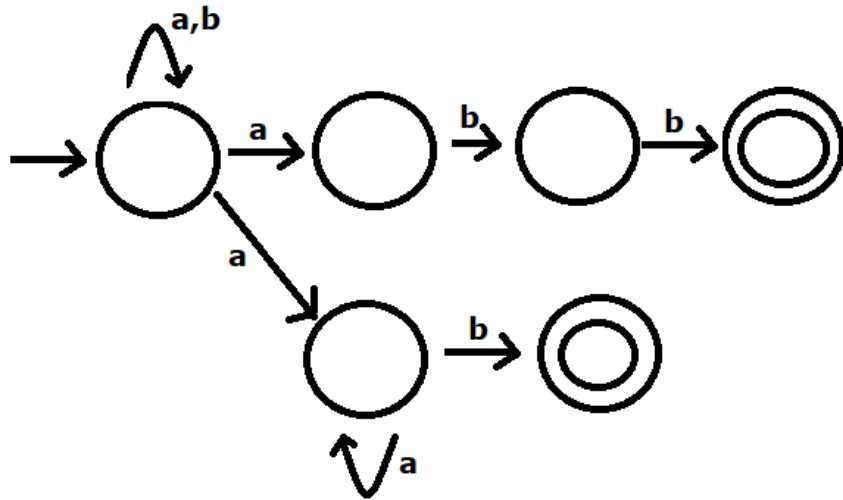
wz  
wxz  
wyz  
wxxz  
wxyz  
wxyxz



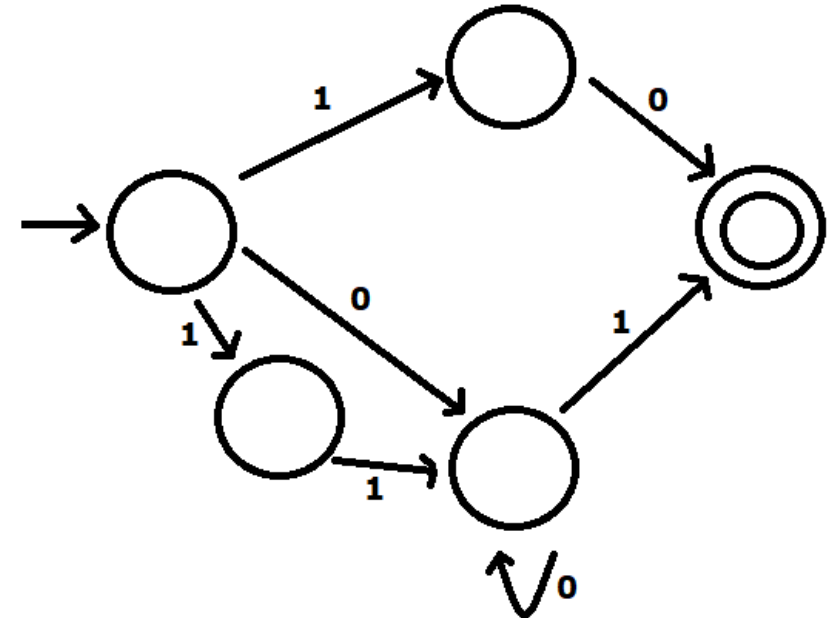


# Örnekler

$(a|b)^* (abb|a^+b)$



$10 + (0+11)0^*1$



# Düzenli Gramerler İçerik Bağımsız Diller (CFG)

---

DR. ŞAFAK KAYIKÇI

# Düzenli Gramerler

---

Gramer dört bileşen ile ifade edilir.  $G=(V, T, S, P)$

$V$  = değişkenler kümesi (non-terminaller)

$T$  = terminaller kümesi

$S$  = başlangıç sembolü

$P$  = terminaller ve non-terminaller için yapım (production) kuralları

$\alpha \rightarrow \beta$  şeklinde olup  $\alpha, \beta \in (V \cup T)$

# Düzenli Gramerler

---

## Right Linear Grammer

$$A \rightarrow x B$$

$$A \rightarrow x$$

$$A, B \in V \text{ ve } x \in T$$

(non-terminal sembol sağda)

$$\text{Örn : } S \rightarrow abS \mid b$$

## Left Linear Grammer

$$A \rightarrow B x$$

$$A \rightarrow x$$

$$A, B \in V \text{ ve } x \in T$$

(non-terminal sembol solda)

$$\text{Örn : } S \rightarrow Sab \mid b$$

# Düzenli Gramerler

---

$G = (\{S, A, B\}, \{a, b\}, S, \{S \rightarrow AB, A \rightarrow a, B \rightarrow b\})$

$V = \{S, A, B\}$

$T = \{a, b\}$

$S = S$

$P = S \rightarrow AB, A \rightarrow a, B \rightarrow b$

$S \rightarrow AB$

$\rightarrow aB$

$\rightarrow ab$

# Düzenli Diller

---

Bir gramerden türetilebilen bütün stringlere, o gramerin dili denir.

$G = (\{S, A\}, \{a, b\}, S, \{S \rightarrow aAb, aA \rightarrow aaAb, A \rightarrow \epsilon\})$

$S \rightarrow \underline{a}Ab$	( $S \rightarrow aAb$ kuralı)
$\rightarrow aa\underline{A}bb$	( $aA \rightarrow aaAB$ kuralı)
$\rightarrow aaaAbbb$	( $A \rightarrow \epsilon$ kuralı)
$\rightarrow aaabbb$	

$$L(G) = \{a^n b^n \mid n > 0\}$$

# Düzenli Diller

---

$G = (\{S, A, B\}, \{a, b\}, S, \{S \rightarrow AB, A \rightarrow aA \mid a, B \rightarrow bB \mid b\})$

$S \rightarrow AB$   
 $\rightarrow ab$

$S \rightarrow AB$   
 $\rightarrow aAbB$   
 $\rightarrow aabb$

$S \rightarrow AB$   
 $\rightarrow aAb$   
 $\rightarrow aab$

$S \rightarrow AB$   
 $\rightarrow abB$   
 $\rightarrow abb$

$L(G) = \{ab, a^2b^2, a^2b, ab^2, \dots\}$   
 $= \{a^m b^n \mid m > 0, n > 0\}$

# İçerik Bağımsız Diller

## Context Free Languages(CFG)

---

Dört bileşen ile ifade edilir.  $G=(V, \Sigma, S, P)$

$V$  = değişkenler kümesi (non-terminaller)

$\Sigma$  = semboller kümesi (terminaller)

$S$  = başlangıç sembolü

$P$  = terminaller ve non-terminaller için yapım (production) kuralları

$A \rightarrow a$  ,  $a = \{V \cup \Sigma\}^*$  ve  $A \in V$



# İçerik Bağımsız Diller

---

Bir string ifadenin gramere dahil olup olmadığını bulmak için

1. start sembolü ile başlayıp, verilen string ifadeye en yakın işlem seçilir
2. Değişkenler uygun işlem kuralları ile değiştirilir. String ifade oluşturulana kadar veya hiçbir işlem kalmayınca kadar devam edilir.

# İçerik Bağımsız Diller

---

$S \rightarrow 0B \mid 1A$ ,  $A \rightarrow 0 \mid 0S \mid 1AA \mid \epsilon$ ,  $B \rightarrow 1 \mid 1S \mid 0BB$  gramerinin 00110101 stringini ürettiğini gösteriniz.

$S \rightarrow 0B$	(string 00110101 olduğu için, 0 ile başlayan kural seçildi) ( $S \rightarrow 0B$ )
$0B \rightarrow 00BB$	(stringin ikinci karakteri 0 olduğu için) ( $B \rightarrow 0BB$ )
$00BB \rightarrow 001B$	( $B \rightarrow 1$ )
$001B \rightarrow 0011S$	( $B \rightarrow 1S$ )
$0011S \rightarrow 00110B$	( $S \rightarrow 0B$ )
$00110B \rightarrow 001101S$	( $B \rightarrow 1S$ )
$001101S \rightarrow 0011010B$	( $S \rightarrow 0B$ )
$0011010B \rightarrow 00110101$	( $B \rightarrow 1$ )

# Derivation (parse) Tree

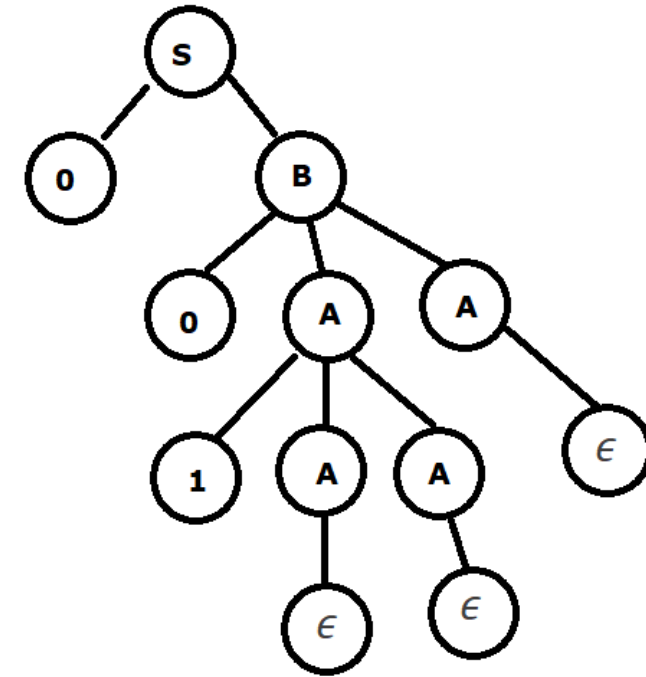
---

$G=(V, T, S, P)$   $S \rightarrow 0B$ ,  $A \rightarrow 1AA | \epsilon$ ,  $B \rightarrow 0AA$

kök düğüm (root vertex) : başlangıç sembolü

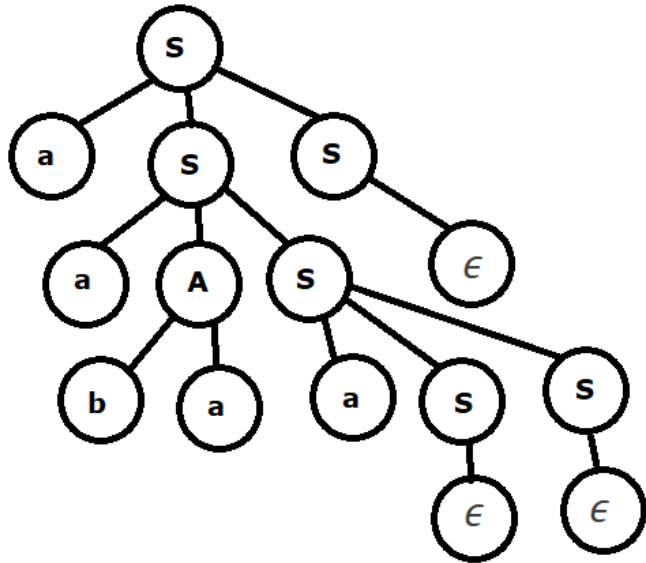
düğüm (vertex) : non-terminal semboller

yaprak (leaves) : terminal semboller yada  $\epsilon$

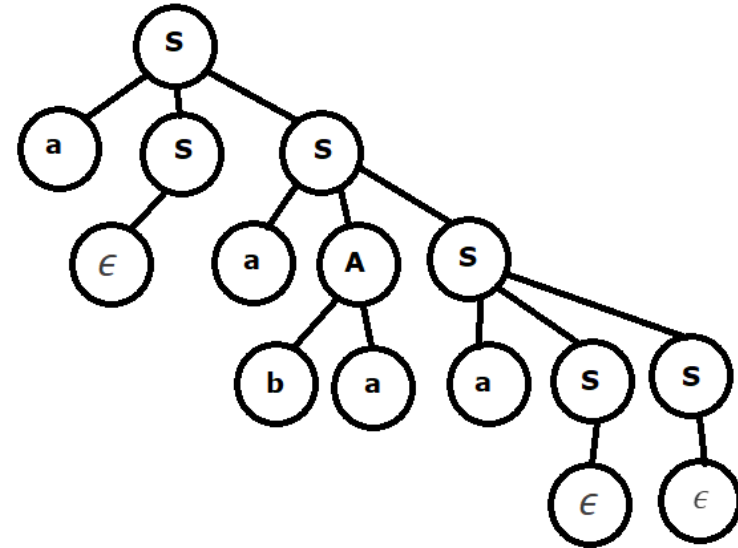


# Left Derivation – Right Derivation

$S \rightarrow aAS \mid aSS \mid \epsilon$ ,  $A \rightarrow SbA \mid ba$  kurallarında aabaa için parse tree



left derivation tree



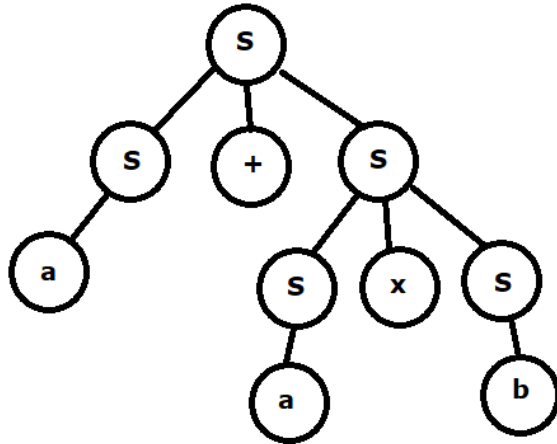
right derivation tree

# Belirsiz (ambiguous) Gramer

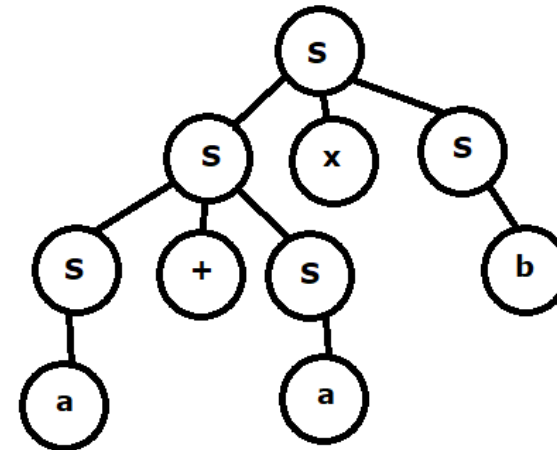
$G = (\{S\}, \{a, b, +, x\}, P, S)$      $S \rightarrow S+S \mid SxS \mid a \mid b$

string : a+axb

$S \rightarrow S+S$   
 $\rightarrow a+S$   
 $\rightarrow a+SxS$   
 $\rightarrow a+axS$   
 $\rightarrow a+axb$



$S \rightarrow SxS$   
 $\rightarrow S+SxS$   
 $\rightarrow a+SxS$   
 $\rightarrow a+axS$   
 $\rightarrow a+axb$



# Pushdown Automata (PDA)

---

DR. ŞAFAK KAYIKÇI

# Pushdown Automata

---

PDA, yığını (stack) olan bir NFA gibidir

$$M = Q, \Sigma, \Gamma, \delta, q_0, Z, F$$

$Q$  = durumlar kümesi

$\Sigma$  = giriş alfabesi

$\Gamma$  = yığın alfabesi

$\delta$  = geçiş fonksiyonu

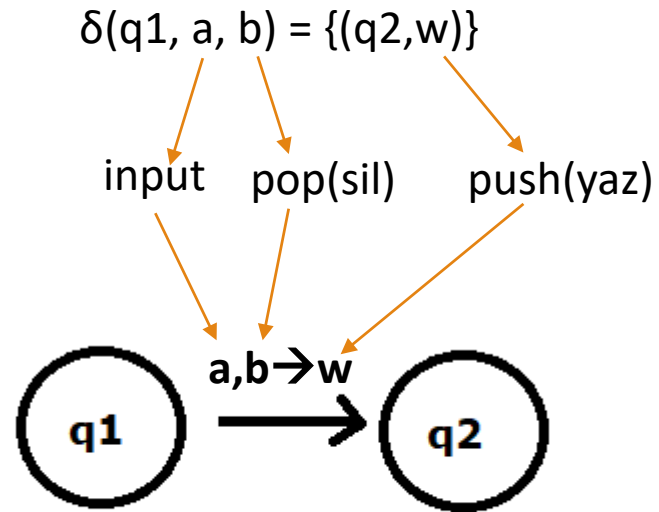
$q_0$  = başlangıç durumu

$Z$  = yığın sembolü (\$)

$F$  = son durum

# Pushdown Automata

---



Bitiş durumu son durumda ve stack boş ise string kabul edilir.



# Örnek

---

$$\begin{aligned} M &= Q, \Sigma, \Gamma, \delta, q_0, Z, F \\ &= (\{q_0, q_1, q_2, q_3\}, \{a, b, c\}, \{A\}, \delta, q_0, \$, \{q_2, q_3\}) \end{aligned}$$

$$\delta(q_0, b, \lambda) = \{(q_1, \lambda)\}$$

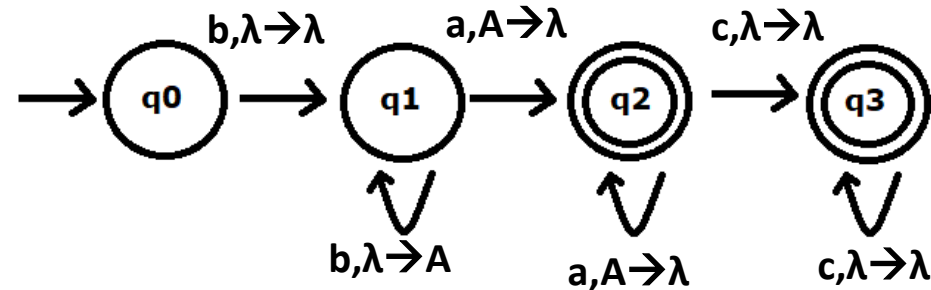
$$\delta(q_1, b, \lambda) = \{(q_1, A)\}$$

$$\delta(q_1, a, A) = \{(q_2, \lambda)\}$$

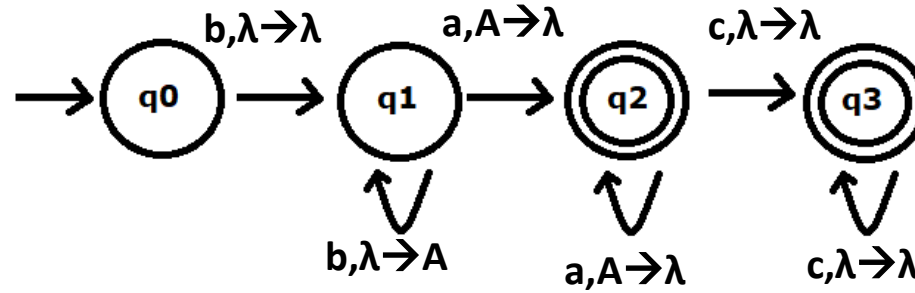
$$\delta(q_2, a, A) = \{(q_2, \lambda)\}$$

$$\delta(q_2, c, \lambda) = \{(q_3, \lambda)\}$$

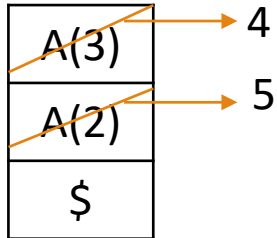
$$\delta(q_3, c, \lambda) = \{(q_3, \lambda)\}$$



# Örnek - devam



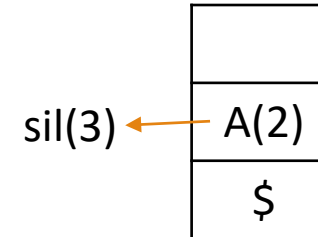
bbbaacc



- b - 1) boş ( $\lambda$  sil,  $\lambda$  yaz)
- b - 2) A yaz
- b - 3) A yaz
- a - 4) A sil (en üstteki)
- a - 5) A sil (ortadaki)
- c - 6) boş ( $\lambda$  sil,  $\lambda$  yaz)
- c - 7) boş ( $\lambda$  sil,  $\lambda$  yaz)

Bitiş durumu son durum ve stack boş.  
Kabul edilir.

bbaac

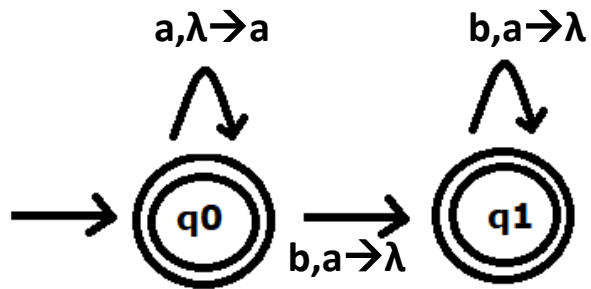


- b - 1) boş ( $\lambda$  sil,  $\lambda$  yaz)
- b - 2) A yaz
- a - 3) A sil
- a - 4) A sil (yığında yok)

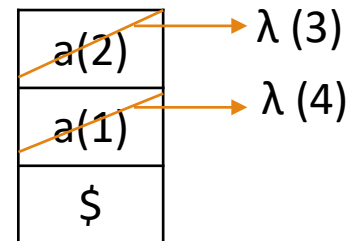
Kabul edilmez

# Örnek

$$M = \{a^n b^n : n \geq 0\}$$



aabb



Bitiş durumu son durum ve stack boş.  
Kabul edilir.

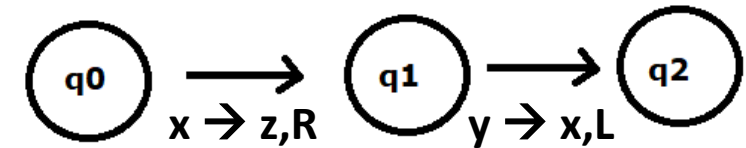
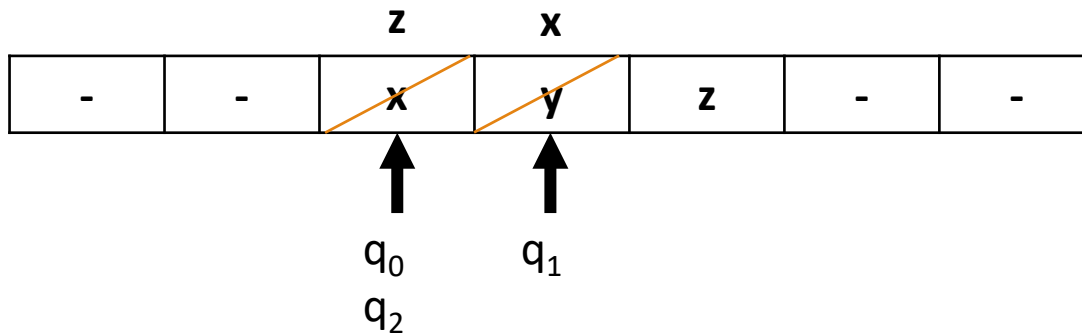
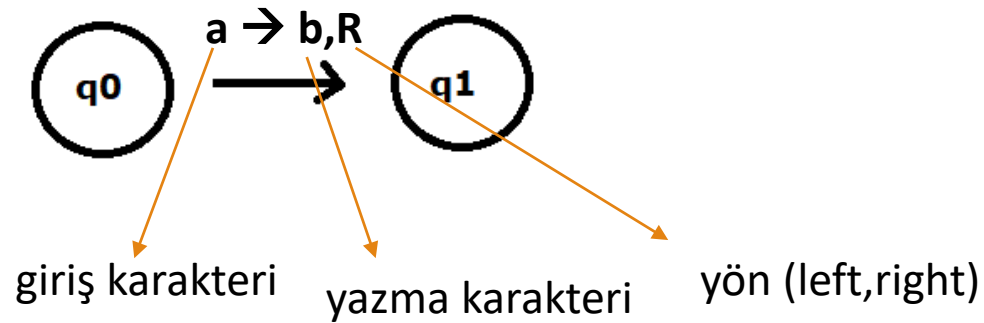
# Turing Makinesi

---

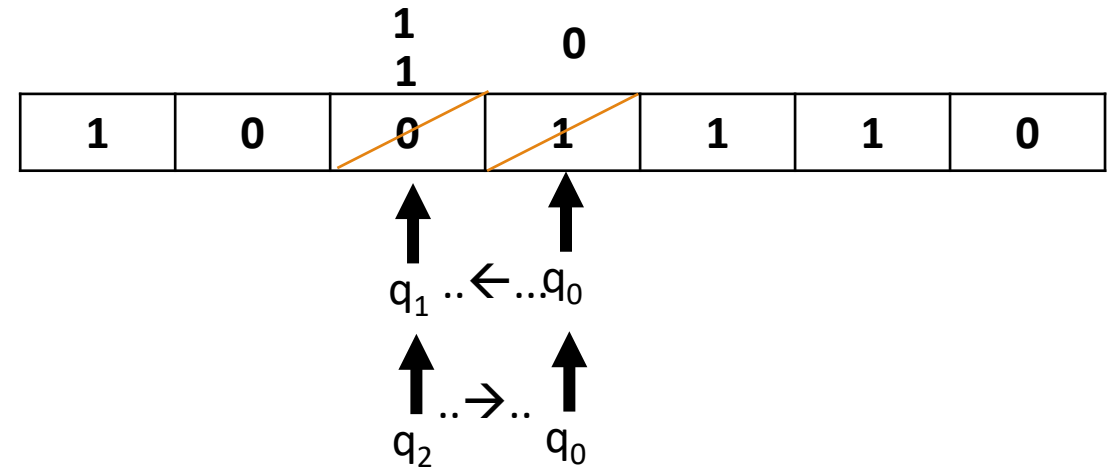
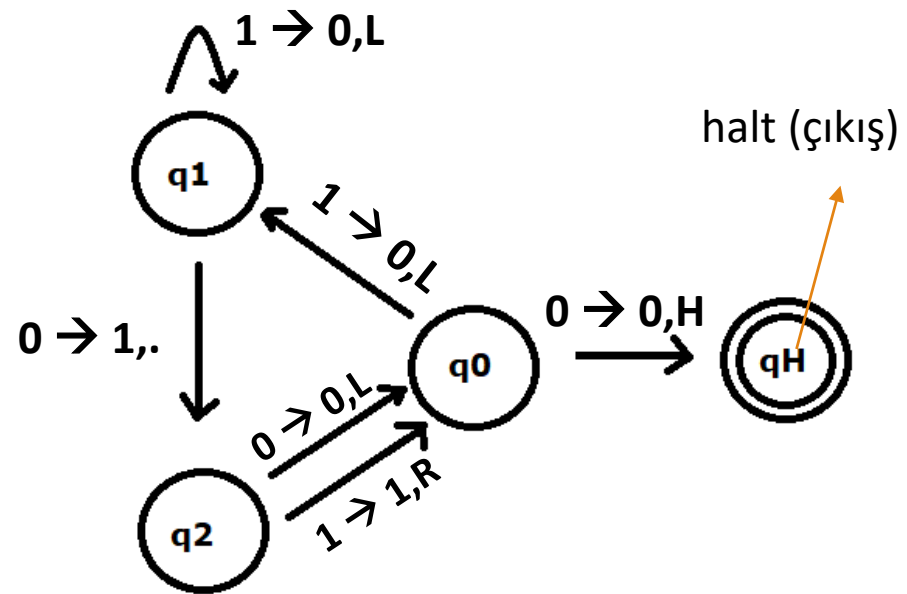
DR. ŞAFAK KAYIKÇI

# Turing Makinesi

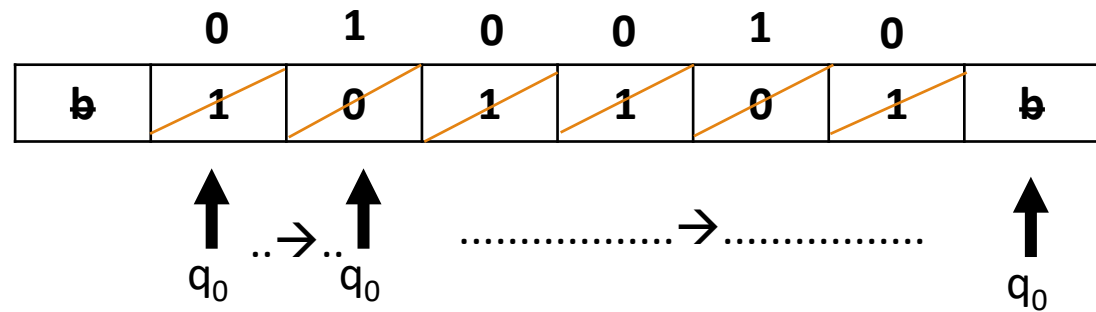
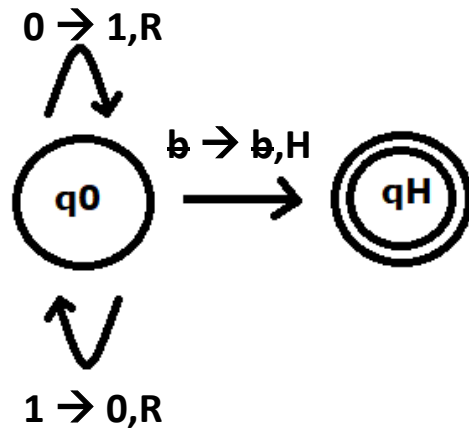
Turing Makinesi, random access memory kullanılır.



# Örnek

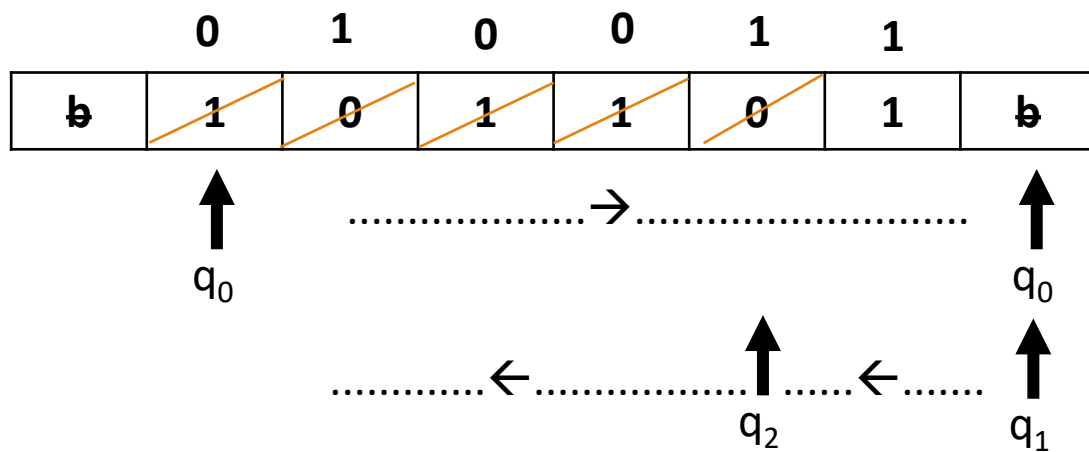


# Örn: 1'e tümleyen

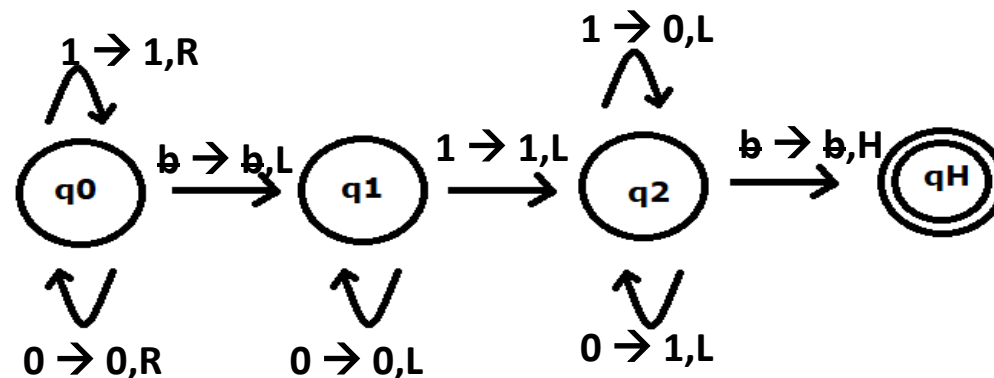


$$\begin{aligned}\delta(q_0, 0) &= (q_0, 1, R) \\ \delta(q_0, 1) &= (q_0, 0, R) \\ \delta(q_0, \text{b}) &= (q_0, \text{b}, \text{halt})\end{aligned}$$

# Örn: 2'e tümleyen



$\delta(q_0, 1) = (q_0, 1, R)$   
 $\delta(q_0, 0) = (q_0, 0, R)$   
 $\delta(q_0, \text{b}) = (q_1, \text{b}, L)$   
 $\delta(q_1, 0) = (q_0, 0, L)$   
 $\delta(q_1, 1) = (q_2, 1, L)$   
 $\delta(q_2, 0) = (q_2, 1, L)$   
 $\delta(q_2, 1) = (q_2, 0, L)$   
 $\delta(q_2, \text{b}) = (q_2, \text{b}, \text{halt})$





# Chomsky Hiyerarşisi

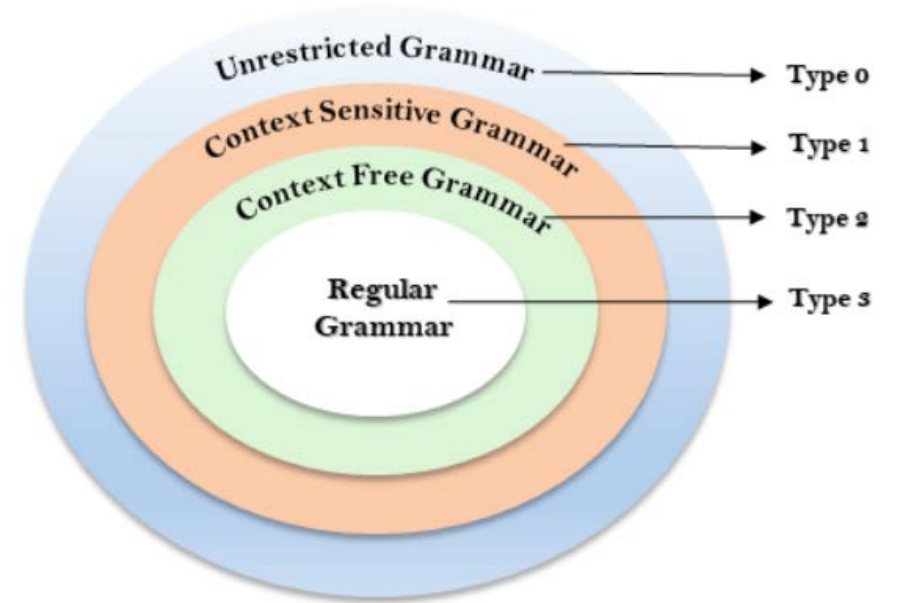
---

DR. ŞAFAK KAYIKÇI

# Chomsky Hiyerarşisi

Chomsky Hiyerarşisi, farklı makine tarafından kabul edilen dil sınıfını temsil eder.

- Type 0 – Sınırsız Dilbilgisi
- Type 1 – İçerik Duyarlı Dilbilgisi
- Type 2 – İçerik Bağımsız Dilbilgisi
- Type 3 – Düzenli Dilbilgisi



# Type 0 Dilbilgisi

---

Tip 0 dilbilgisi kısıtlanmamış dilbilgisi olarak bilinir. Bu tür dillerin dilbilgisi kurallarında herhangi bir kısıtlama yoktur. Bu diller Turing makineleri tarafından verimli bir şekilde modellenenir.

Örneğin:

$bAa \rightarrow aa$

$S \rightarrow s$

# Tip 1 Dilbilgisi

---

- Tip 1 dilbilgisi, içerik duyarlı dilbilgisi olarak tanımlanır.
- İçerik duyarlı dilbilgisi, üretim kurallarının sol tarafında birden fazla simgeye sahip olabilir.
- Sol taraftaki sembol sayısı, sağ taraftaki sembol sayısını geçmemelidir.
- A bir başlangıç sembolü olmadıkça  $A \rightarrow \varepsilon$  formunun kuralına izin verilmez. Herhangi bir kuralın sağ tarafında meydana gelmez.
- Tip 1 dilbilgisi Tip 0 olmalıdır. Tip 1'de Üretim  $V \rightarrow T$  şeklindedir. V'deki sembol sayısı T'den küçük veya ona eşittir.

Örneğin:

$S \rightarrow AT$

$T \rightarrow xy$

$A \rightarrow a$

# Tip 2 Dilbilgisi

---

Tip 2 Dilbilgisi, İçerik Bağımsız Dilbilgisi olarak bilinir. İçerik bağımsız diller, İçerik bağımsız dilbilgisi (CFG) ile temsil edilebilecek dillerdir. Tip 2 tip 1 olmalıdır. Üretim kuralı şu şekildedir

$$A \rightarrow \alpha$$

Burada A, herhangi bir terminal olmayan ve terminaller ile terminal olmayanların herhangi bir kombinasyonudur.

Örneğin:

$$A \rightarrow aBb$$

$$A \rightarrow b$$

$$B \rightarrow a$$

# Tip 3 Dilbilgisi

---

Tip 3 Dil Bilgisi düzenli bilgisi olarak bilinir. Düzenli diller, düzenli ifadeler kullanılarak tanımlanabilen dillerdir. Bu diller NFA veya DFA tarafından modellenenebilir.

Tip 3 en kısıtlanmış dilbilgisidir. Tip 3 dilbilgisi Tip 2 ve Tip 1 olmalıdır.

Tip 3 formu şu şekildedir :

$$V \rightarrow T^* V / T^*$$

Örneğin:

$$A \rightarrow xy$$

# Genel Gösterim

Grammar	Languages	Automaton	Production rules (constraints)*	Examples <sup>[3]</sup>
Type-0	Recursively enumerable	Turing machine	$\alpha A \beta \rightarrow \delta$ (no constraints)	$L = \{w   w \text{ describes a terminating Turing machine}\}$
Type-1	Context-sensitive	Linear-bounded non-deterministic Turing machine	$\alpha A \beta \rightarrow \alpha \gamma \beta$	$L = \{a^n b^n c^n   n > 0\}$
Type-2	Context-free	Non-deterministic pushdown automaton	$A \rightarrow \alpha$	$L = \{a^n b^n   n > 0\}$
Type-3	Regular	Finite state automaton	$A \rightarrow a$ and $A \rightarrow aB$	$L = \{a^n   n \geq 0\}$
<p>* Meaning of symbols:</p> <ul style="list-style-type: none"> <li>• <math>a</math> = terminal</li> <li>• <math>A, B</math> = non-terminal</li> <li>• <math>\alpha, \beta, \gamma, \delta</math> = string of terminals and/or non-terminals <ul style="list-style-type: none"> <li>• <math>\alpha, \beta, \delta</math> = maybe empty</li> <li>• <math>\gamma</math> = never empty</li> </ul> </li> </ul>				

# JLAP Kullanımı

---

DR. ŞAFAK KAYIKÇI



# JFLAP

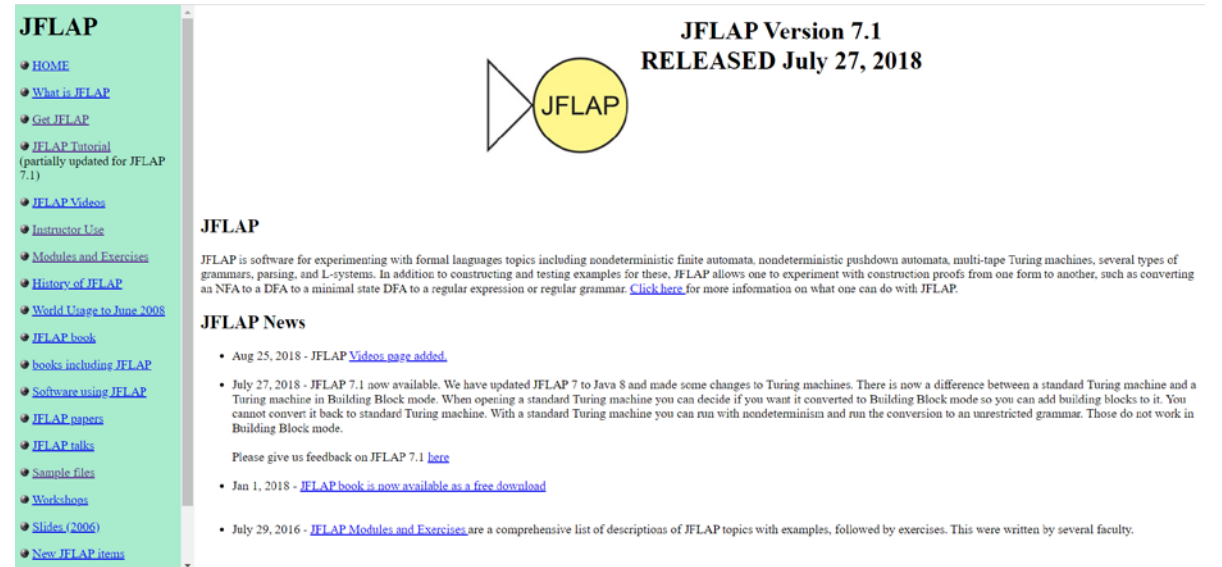
<http://www.jflap.org/>

download

<http://www.jflap.org/jflaptmp/>

tutorial

<http://www.jflap.org/tutorial/>



The screenshot shows the JFLAP website homepage. On the left is a green sidebar with a list of links: HOME, What is JFLAP, Get JFLAP, JFLAP Tutorial (partially updated for JFLAP 7.1), JFLAP Videos, Instructor Use, Modules and Exercises, History of JFLAP, World Usage to June 2008, JFLAP book, books including JFLAP, Software using JFLAP, JFLAP papers, JFLAP talks, Sample files, Workshops, Slides (2006), and New JFLAP items. The main content area has a white background. At the top right, it says 'JFLAP Version 7.1 RELEASED July 27, 2018' next to a logo consisting of a yellow circle with 'JFLAP' inside a black triangle. Below this, the 'JFLAP' section describes the software's capabilities. The 'JFLAP News' section contains several bullet points with dates and links to updates and feedback.

**JFLAP**

• HOME  
• What is JFLAP  
• Get JFLAP  
• JFLAP Tutorial (partially updated for JFLAP 7.1)  
• JFLAP Videos  
• Instructor Use  
• Modules and Exercises  
• History of JFLAP  
• World Usage to June 2008  
• JFLAP book  
• books including JFLAP  
• Software using JFLAP  
• JFLAP papers  
• JFLAP talks  
• Sample files  
• Workshops  
• Slides (2006)  
• New JFLAP items

**JFLAP Version 7.1  
RELEASED July 27, 2018**

**JFLAP**

JFLAP is software for experimenting with formal languages topics including nondeterministic finite automata, nondeterministic pushdown automata, multi-tape Turing machines, several types of grammars, parsing, and L-systems. In addition to constructing and testing examples for these, JFLAP allows one to experiment with construction proofs from one form to another, such as converting an NFA to a DFA to a minimal state DFA to a regular expression or regular grammar. [Click here](#) for more information on what one can do with JFLAP.

**JFLAP News**

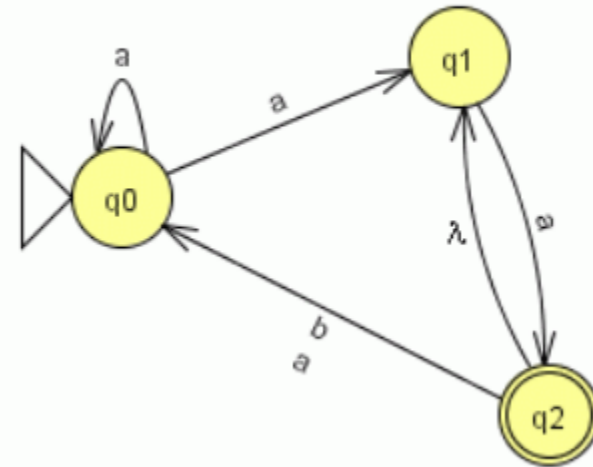
- Aug 25, 2018 - JFLAP [Videos page added](#).
- July 27, 2018 - JFLAP 7.1 now available. We have updated JFLAP 7 to Java 8 and made some changes to Turing machines. There is now a difference between a standard Turing machine and a Turing machine in Building Block mode. When opening a standard Turing machine you can decide if you want it converted to Building Block mode so you can add building blocks to it. You cannot convert it back to standard Turing machine. With a standard Turing machine you can run with nondeterminism and run the conversion to an unrestricted grammar. Those do not work in Building Block mode.  
Please give us feedback on JFLAP 7.1 [here](#)
- Jan 1, 2018 - [JFLAP book is now available as a free download](#)
- July 29, 2016 - [JFLAP Modules and Exercises](#) are a comprehensive list of descriptions of JFLAP topics with examples, followed by exercises. These were written by several faculty.

JFLAP, belirli - belirsiz sonlu otomata, push down otomata, Turing makineleri, düzenli gramerler, içerik bağımsız diller konularını denemek için kullanabileceğiniz bir yazılımdır.

# Düzenli Dillerin Oluşturulması

**Regular languages - create**

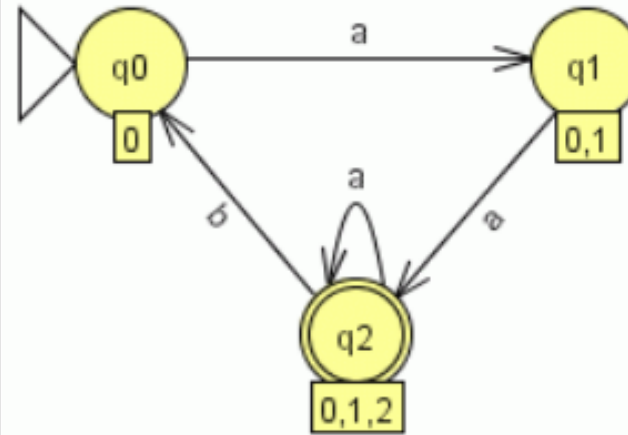
- **DFA**
- **NFA**
- **regular grammar**
- **regular expression**



# Düzenli Diller - Dönüşümler

## Regular languages - conversions

- **NFA  $\rightarrow$  DFA  $\rightarrow$  Minimal DFA**
- **NFA  $\leftrightarrow$  regular expression**
- **NFA  $\leftrightarrow$  regular grammar**

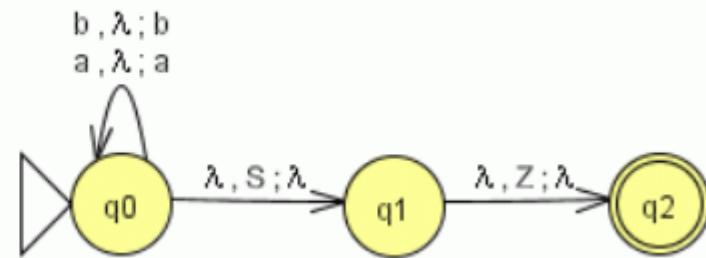


# İçerik Bağımsız Dillerin Oluşturulması

---

**Context-free languages - create**

- **push-down automaton**
- **context-free grammar**

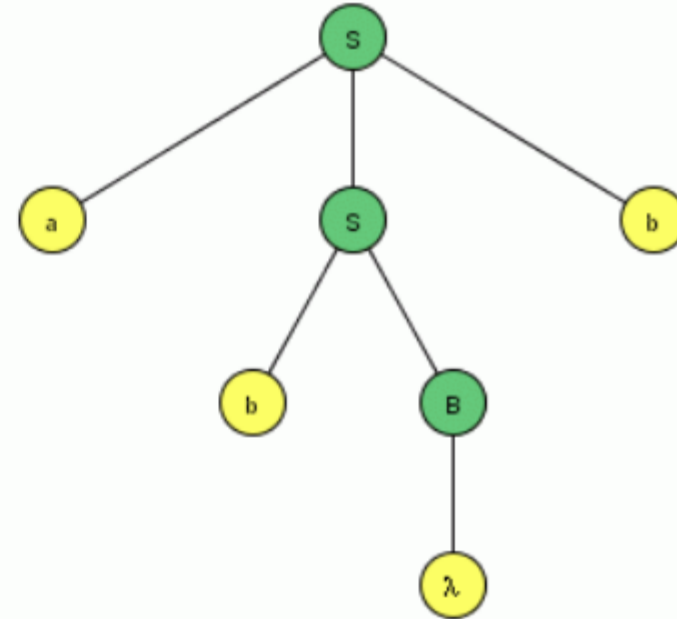


# İçerik Bağımsız Diller - Dönüşümler

---

## Context-free languages - transform

- **PDA  $\rightarrow$  CFG**
- **CFG  $\rightarrow$  PDA (LL parser)**
- **CFG  $\rightarrow$  PDA (SLR parser)**
- **CFG  $\rightarrow$  CNF**
- **CFG  $\rightarrow$  LL parse table and parser**
- **CFG  $\rightarrow$  SLR parse table and parser**
- **CFG  $\rightarrow$  Brute force parser**

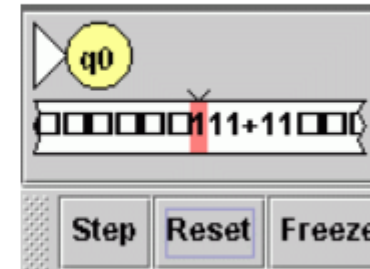


# Turing Makineleri

---

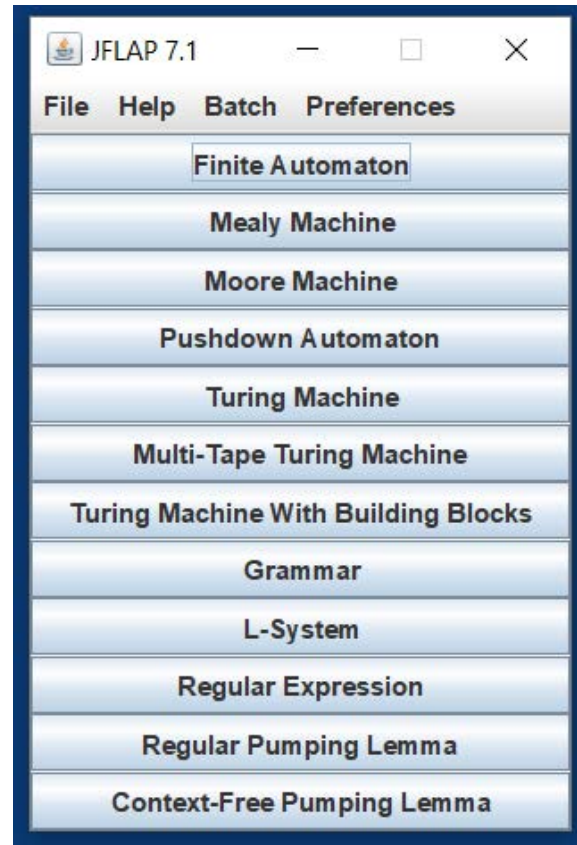
## Recursively Enumerable languages

- Turing machine (1-tape)
- Turing machine (multi-tape)
- Turing machine (building blocks)
- unrestricted grammar
- unrestricted grammar  $\rightarrow$  brute force parser



# Arayüz

---



# Arayüz

