FINITE STATE AUTOMATA DETERMINISTIC FINITE STATE (DFA)

DR. ŞAFAK KAYIKÇI

Sonlu Durumlar Makinesi (FSA)

Sembol = a, b, c, 0, 1, ,2, ...
Alfabe = Σ (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

Σ = {0,1} → 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} → sonlu küme

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

Σ kuvvetleri

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

Σ^0 = Boyutu 0 olan bütün stringler = $\Sigma^0 = \{\in\}$ (bazı yerlerde λ ile gösterilir)

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

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

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

……

Σ^n = Boyutu n olan bütün stringler = Σ^n

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

Sigma star (Kleene Yıldızı)

$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots\ldots.$

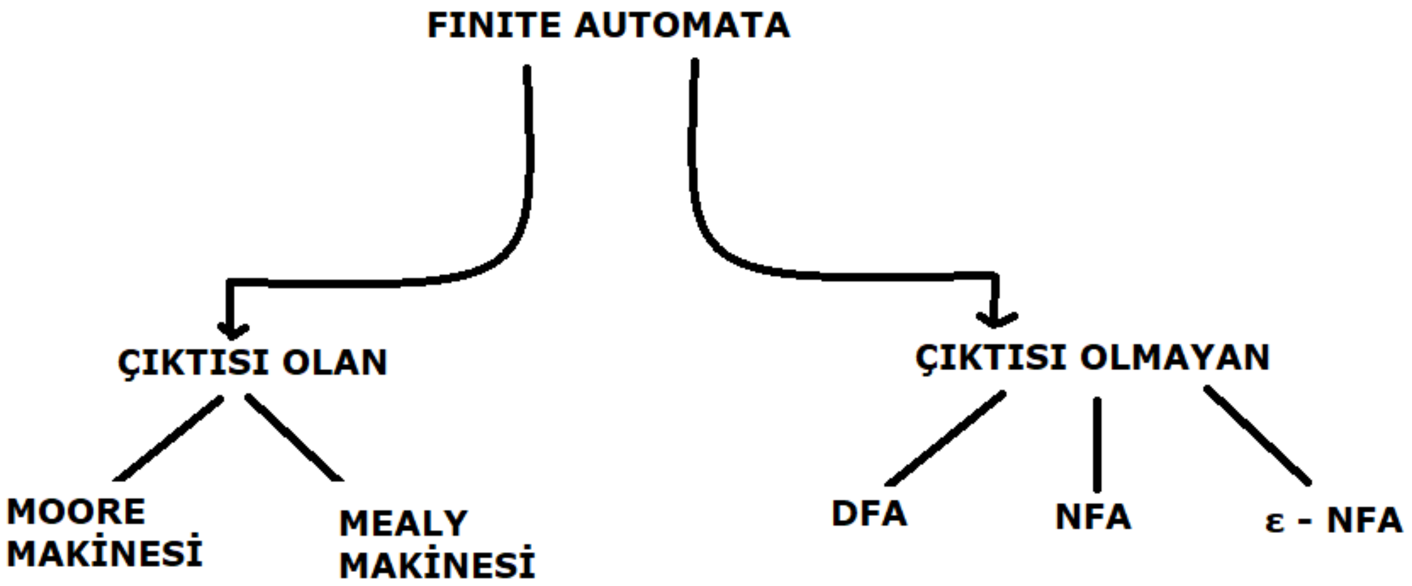
$= \{\in\} \cup \{0,1\} \cup \{00, 01, 10, 11\} \cup \ldots\ldots.$

= bütün olasılıklar

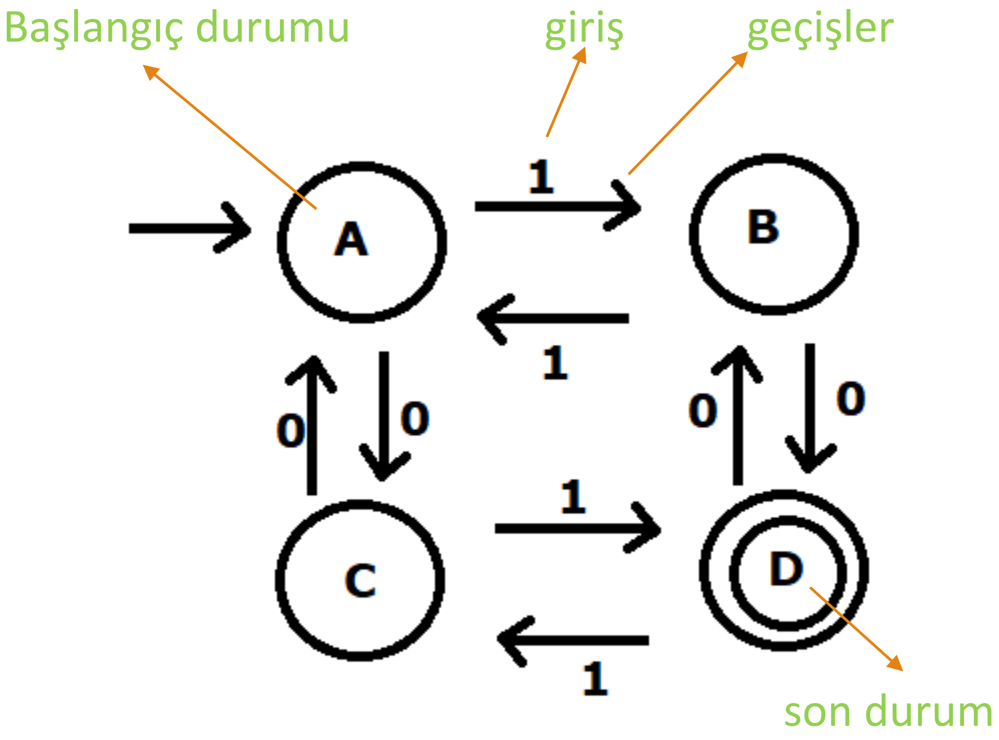
Kleene Kapaması (Artı)

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

Sonlu Durumlar Makinesi



DETERMINISTIC FINITE AUTOMATA (DFA)



(Q, Σ, q_0, F, δ)

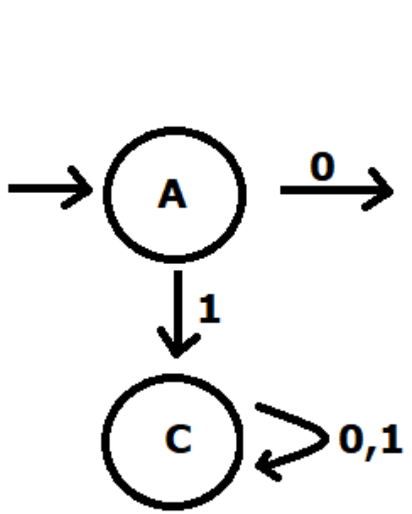
Q = tüm durumların kümesi
Σ = giriş alfabesi
q_0 = başlangıç durumu
F = son durumlar kümesi
δ = geçiş fonksiyonları Qx Σ →Q

Q = {A, B, C, D}
Σ = {0, 1}
q_0 = A
δ = geçiş fonksiyonları Qx Σ →Q
F = {D}
δ tablosu

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

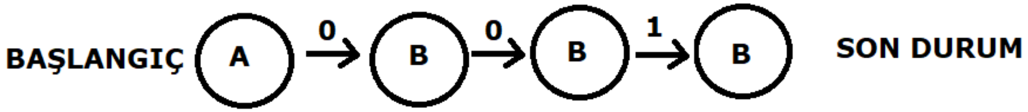
DFA – örnek 1

L1 = 0 ile başlayan bütün stringler = {0, 00, 01, 000, 010, 011, ….}

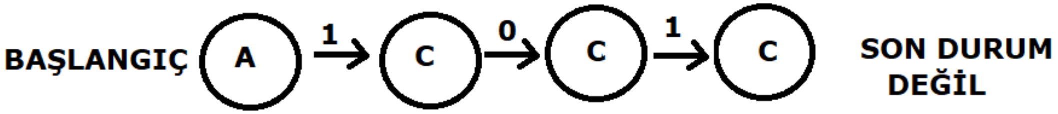


X ölü durum

0 0 1 - KABUL

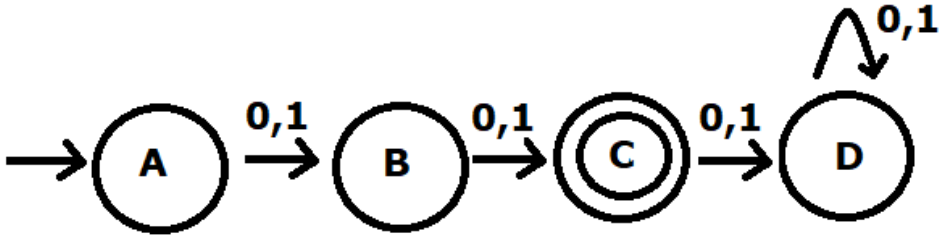


1 0 1 - RED

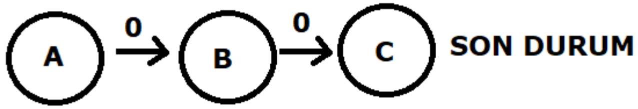


DFA – örnek 2

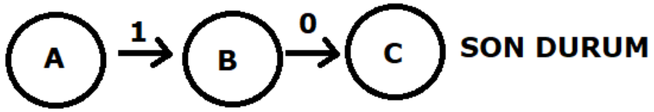
L2 = boyutu iki olan bütün stringler = {00,01,10,11}



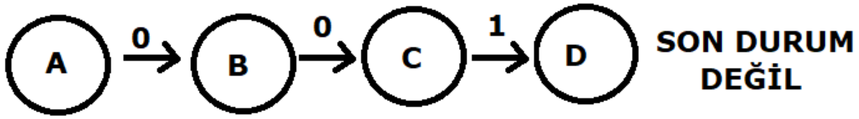
0 0 - KABUL



1 0 - KABUL



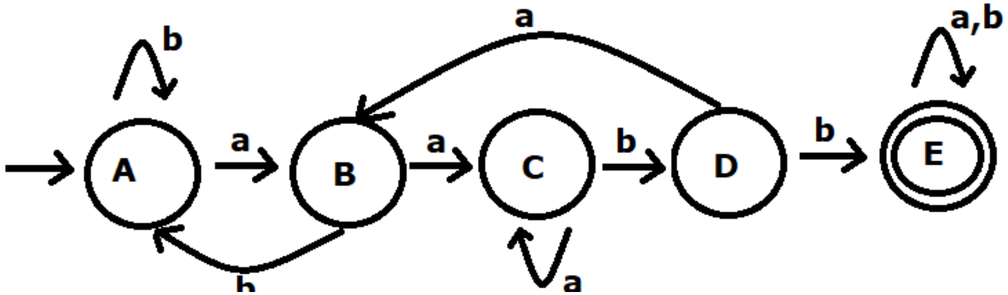
0 0 1 – RED



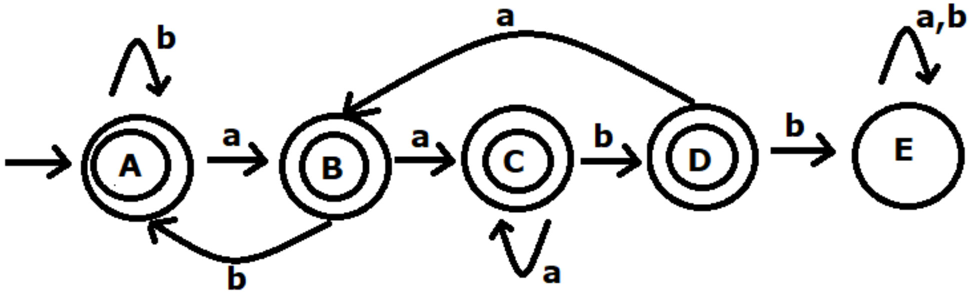
DFA – örnek 3

Σ = {a, b} üzerinde **<u>aabb içermeyen</u>** 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.

$$\delta(A,x) \to F \ ve \ \ \delta(B,x) \to F$$

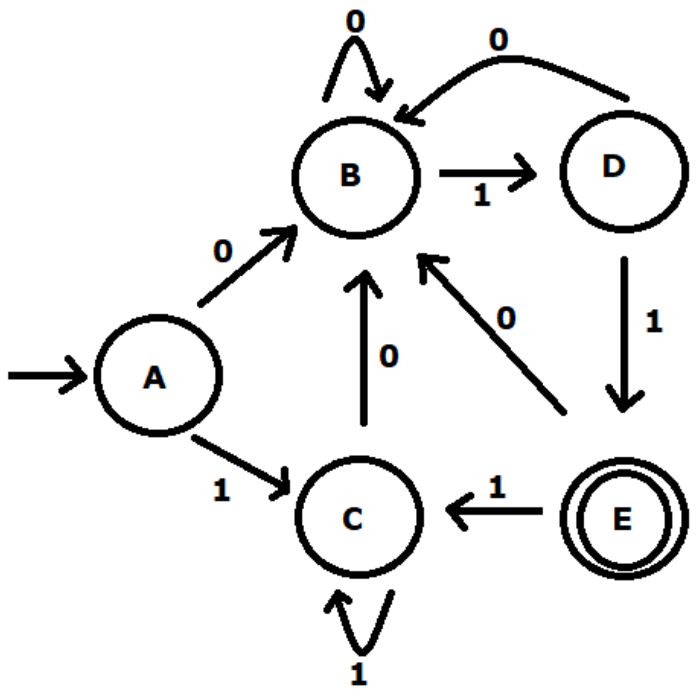
veya

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

|x| = 0 → A ve B ‘0’ denklik
|x| = 1 → A ve B ‘1’ denklik
…..
|n| = n → A ve B ‘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 X

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

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

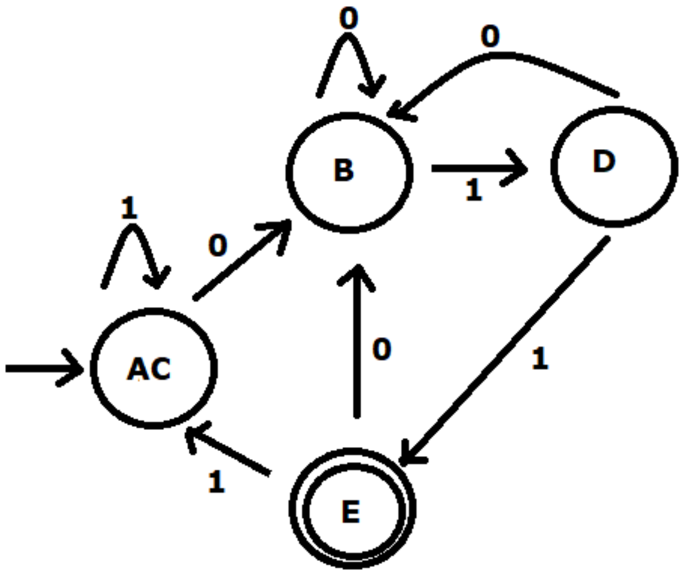
3 denklik = {A,C} {B} {D} {E}

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}

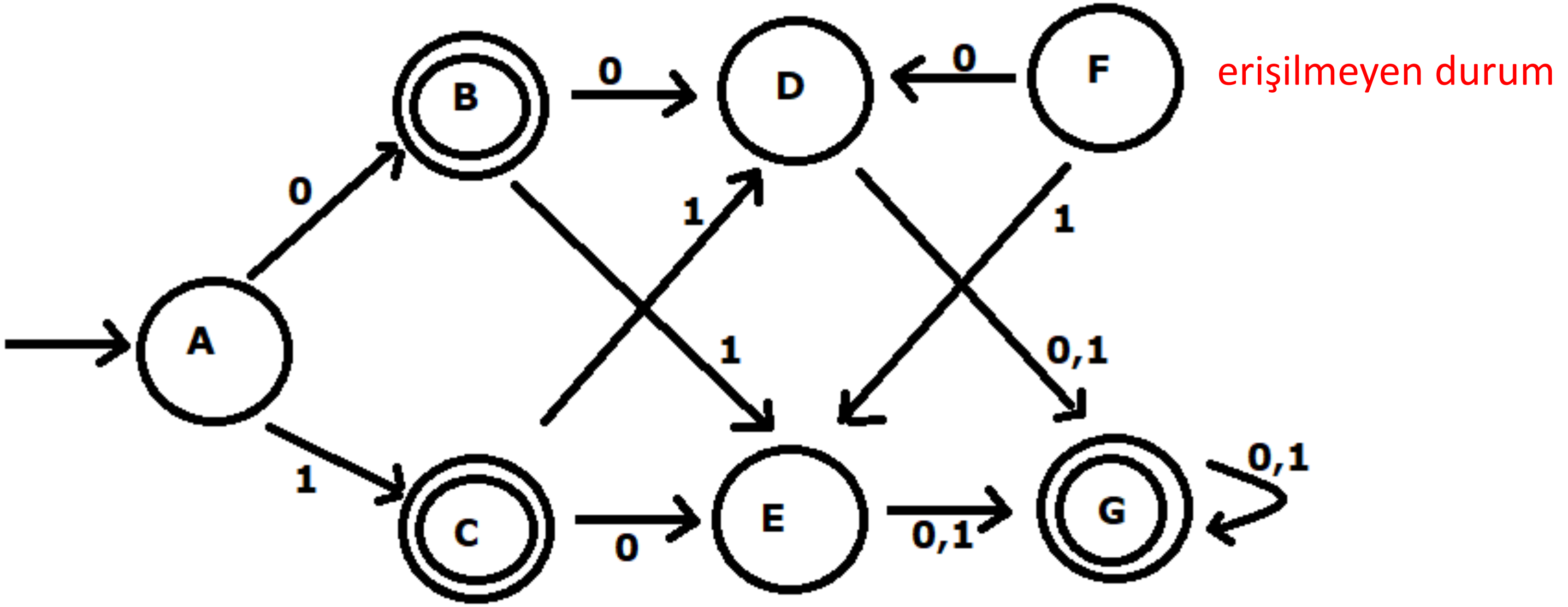
2 denklik = {q0, q4}   {q6}   {q1,q7}   {q2}   {q3,q5}

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.

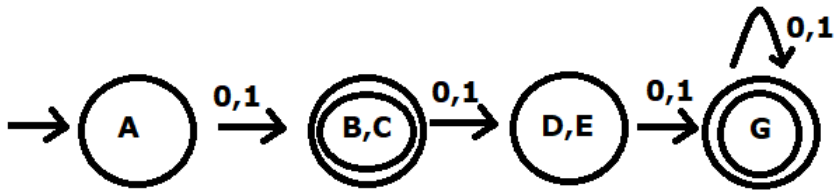
	0	1
→A	B	C
B	D	E
C	E	D
D	G	G
E	G	G
G	G	G

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

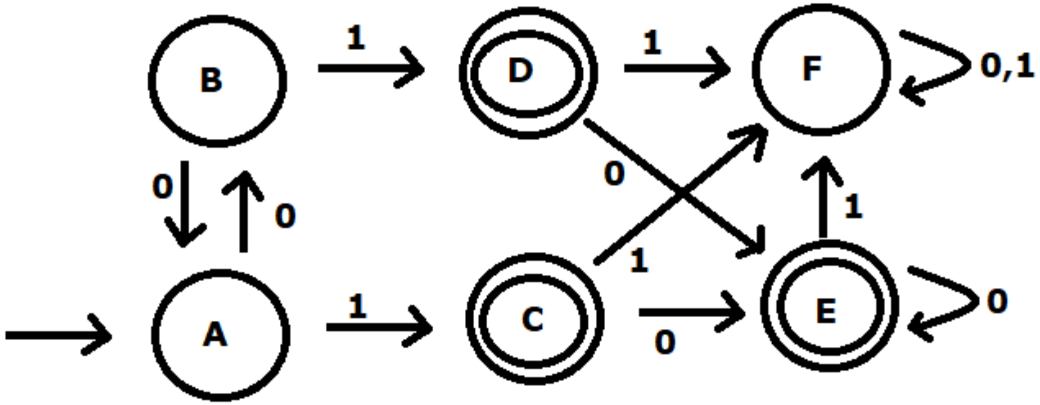
DR. ŞAFAK KAYIKÇI

Myhill – Nerode Teoremi (Doldurma Metodu)

Adımlar :

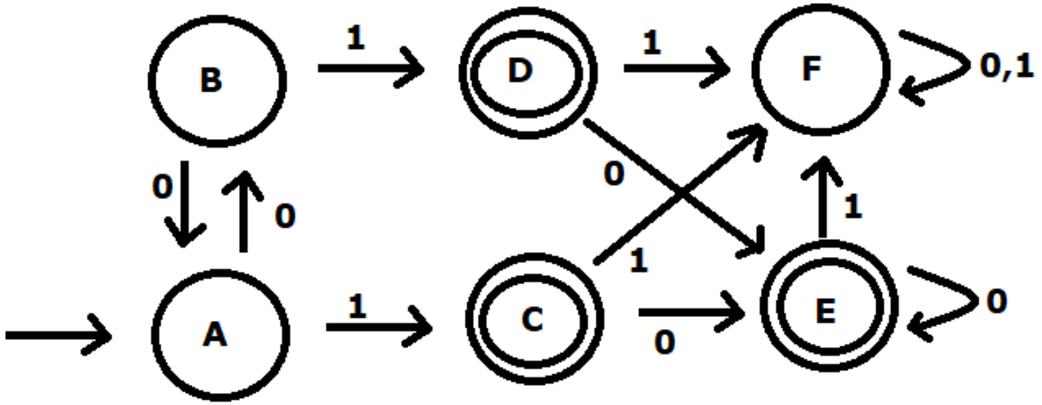
1. Her (P,Q) çifti için durumlar tablosu oluşturulur.
2. $P \in F$ ve $Q \notin F$ olan çiftler işaretlenir.
3. İşaretlenmemiş *(P,Q)* çiftleri arasında $[\delta(P,x), \delta(Q,x)]$ işaretlenmiş ise bunlarda işaretlenir.
4. Daha fazla işlem yapılamayacak duruma kadar döngü devam eder.
5. İşaretlenmemiş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

Örnek -1



	A	B	C	D	E	F
A						
B						
C	$\surd_1$	$\surd_1$				
D	$\surd_1$	$\surd_1$				
E	$\surd_1$	$\surd_1$				
F			$\surd_1$	$\surd_1$	$\surd_1$	

Çiftlerden biri son durum olup, diğeri olmayanlar işaretlenir.



	A	B	C	D	E	F
A						
B						
C	$\surd_1$	$\surd_1$				
D	$\surd_1$	$\surd_1$				
E	$\surd_1$	$\surd_1$				
F			$\surd_1$	$\surd_1$	$\surd_1$	

(B,A) için

δ(B,0):A
δ(A,0):B
AB işaretsiz, bir şey yapma

δ(B,1):D
δ(A,1):C
DC işaretsiz, bir şey yapma

(D,C) için

δ(D,0):E
δ(C,0):E
EE tabloda yok, bir şey yapma

δ(D,1):F
δ(C,1):F
FF tabloda yok, bir şey yapma

(E,C) için

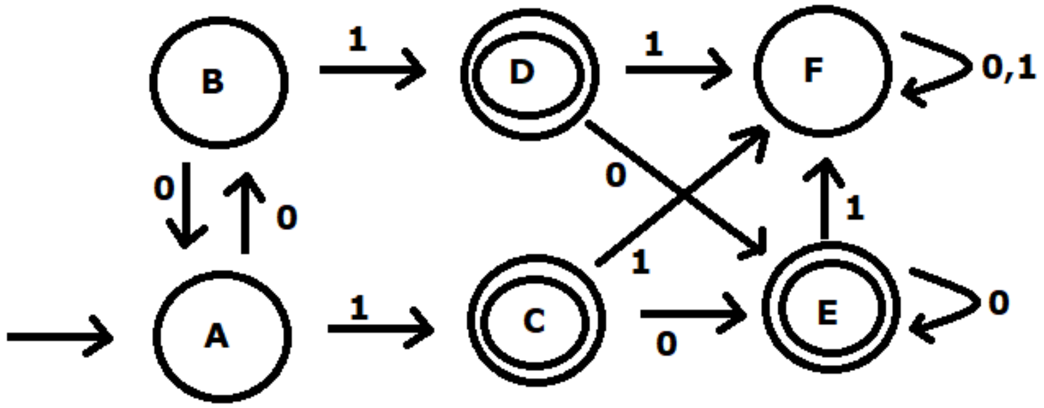
δ(E,0):E
δ(C,0):E
EE tabloda yok, bir şey yapma

δ(E,1):F
δ(C,1):F
FF tabloda yok, bir şey yapma

(E,D) için

δ(E,0):E
δ(D,0):E
EE tabloda yok, bir şey yapma

δ(E,1):F
δ(D,1):F
FF tabloda yok, bir şey yapma



	A	B	C	D	E	F
A						
B						
C	$\surd_1$	$\surd_1$				
D	$\surd_1$	$\surd_1$				
E	$\surd_1$	$\surd_1$				
F			$\surd_1$	$\surd_1$	$\surd_1$	

(F,A) için

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

FB işaretsiz, bir şey yapma

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

FC işaretli, dolayısıyla FA işaretliyoruz

(F,B) için

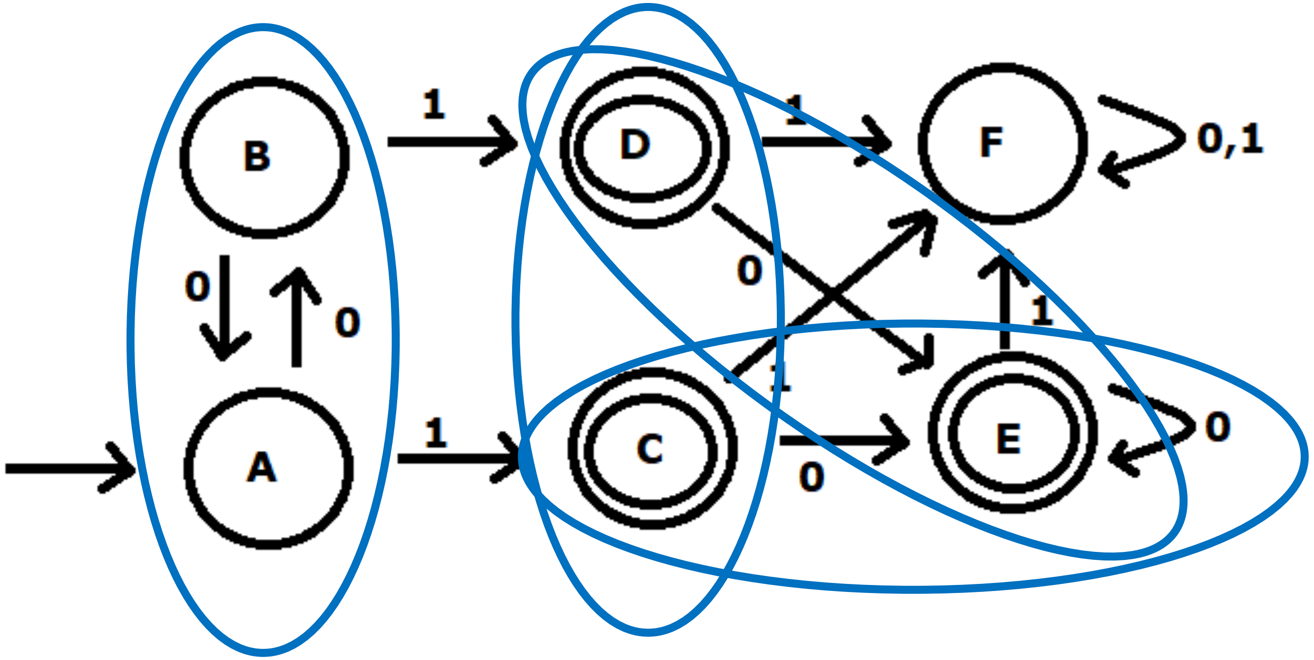
$\delta(F,0)$:F
$\delta(B,0)$:A

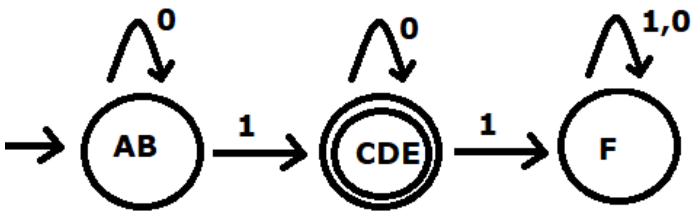
FA tabloda işaretli, dolayısıyla FB işaretliyoruz

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

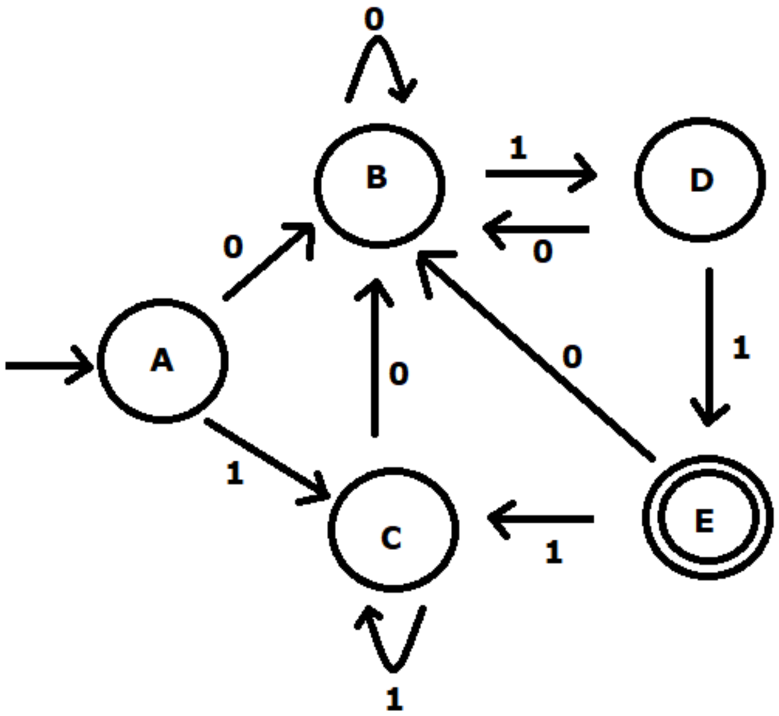
	A	B	C	D	E	F
A						
B						
C	$\surd_1$	$\surd_1$				
D	$\surd_1$	$\surd_1$				
E	$\surd_1$	$\surd_1$				
F	$\surd_2$	$\surd_2$	$\surd_1$	$\surd_1$	$\surd_1$	

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



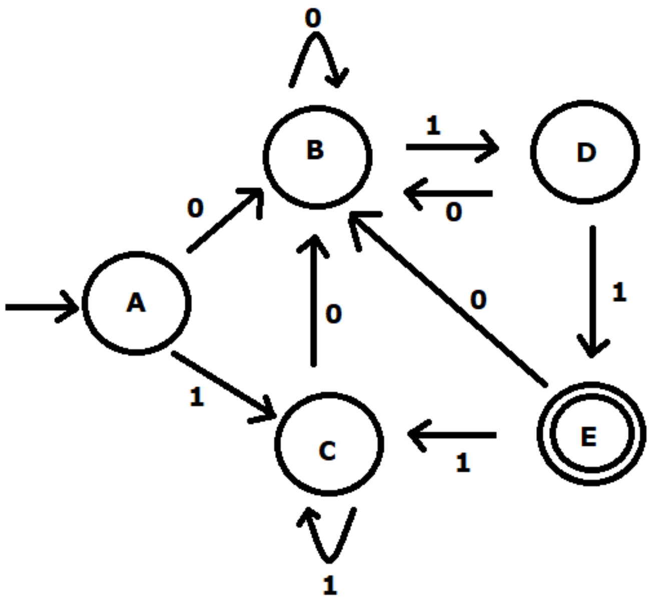


Örnek 2



	A	B	C	D	E
A					
B					
C					
D					
E	$\surd_1$	$\surd_1$	$\surd_1$	$\surd_1$	

Çiftlerden biri son durum olup, diğeri olmayanlar işaretlenir.



	A	B	C	D	E
A					
B					
C					
D	$\surd_2$				
E	$\surd_1$	$\surd_1$	$\surd_1$	$\surd_1$	

(B,A) için

δ(B,0):B
δ(A,0):B
BB tabloda yok, bir şey yapma

δ(B,1):D
δ(A,1):C
DC işaretsiz, bir şey yapma

(C,A) için

δ(C,0):B
δ(A,0):B
BB tabloda yok, bir şey yapma

δ(C,1):C
δ(A,1):C
CC tabloda yok, bir şey yapma

(C,B) için

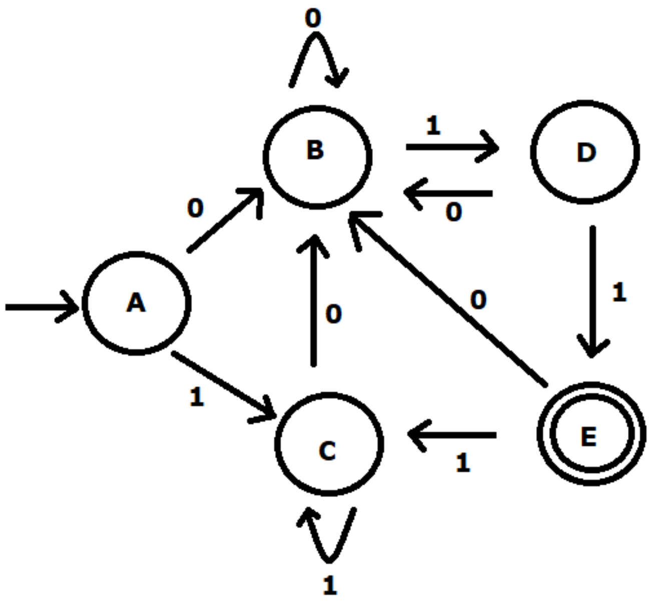
δ(C,0):B
δ(B,0):B
BB tabloda yok, bir şey yapma

δ(C,1):C
δ(B,1):D
DC işaretsiz, bir şey yapma

(A,D) için

δ(A,0):B
δ(D,0):B
BB tabloda yok, bir şey yapma

δ(A,1):C
δ(D,1):E
CE işaretli, dolayısıyla AD işaretliyoruz



	A	B	C	D	E
A					
B					
C					
D	$\surd_2$	$\surd_2$	$\surd_2$		
E	$\surd_1$	$\surd_1$	$\surd_1$	$\surd_1$	

(D,B) için

δ(D,0):B
δ(B,0):B
BB tabloda yok, bir şey yapma

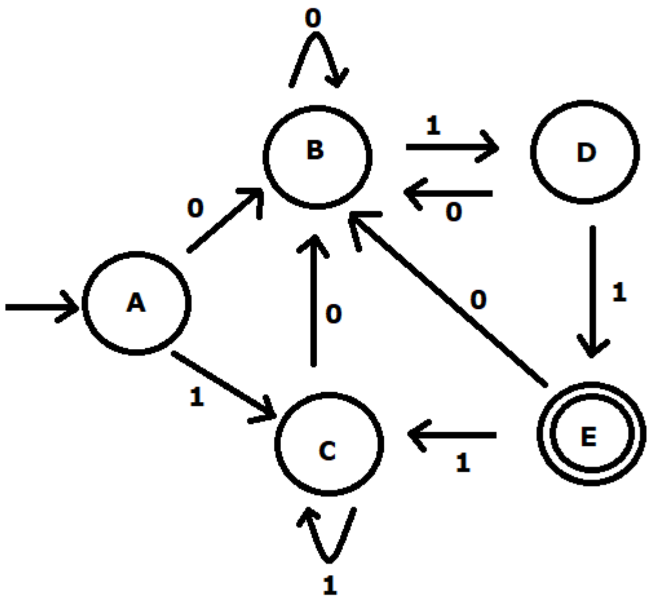
δ(D,1):E
δ(B,1):D
ED işaretli, dolayısıyla DB işaretliyoruz

(D,C) için

δ(D,0):B
δ(C,0):B
BB tabloda yok, bir şey yapma

δ(D,1):E
δ(C,1):C
EC işaretli, dolayısıyla DC işaretliyoruz

1. TUR BİTTİ, 2. TURA BAŞLAYACAĞIZ



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

(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

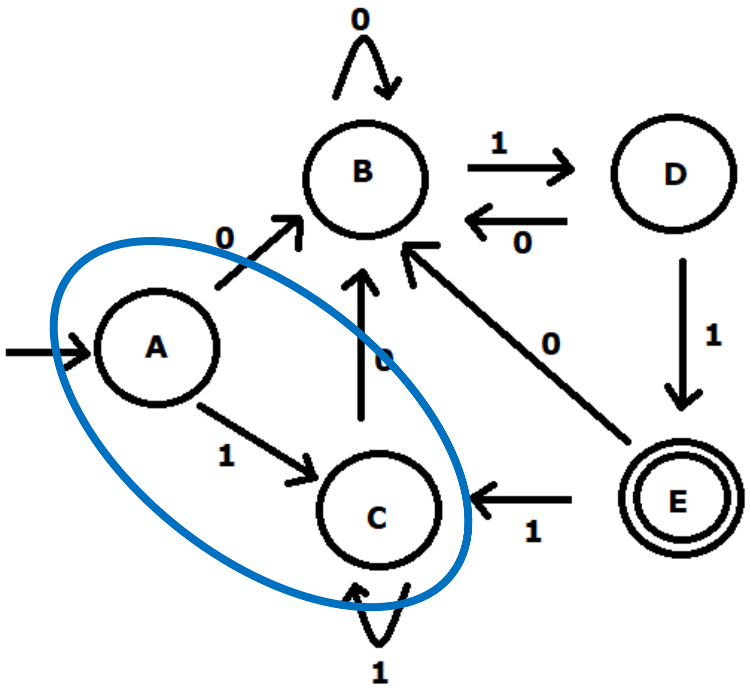
$\delta(A,1)$:C
$\delta(C,1)$:C
CC tabloda yok, bir şey yapma

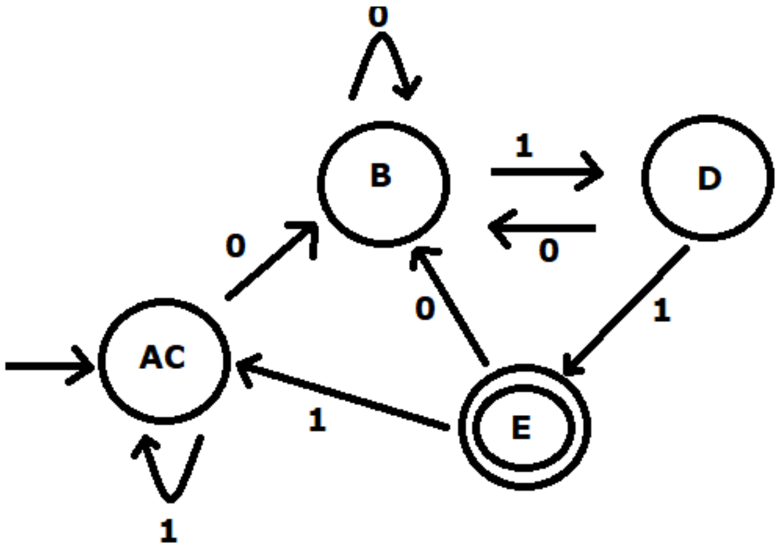
(B,C) için

$\delta(B,0)$: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

ε - NFA

ε → boş sembol ifadeleri gösterir (λ ile de gösterilebilir)

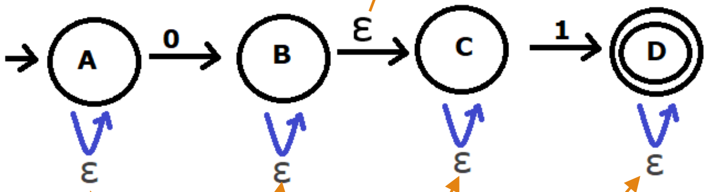
NFA = $(Q, \Sigma, q_0, F, \delta)$

Q = Tüm durumlar kümesi = {A,B}
Σ = giriş (input) alfabesi = {0,1}
q_0= başlangıç durumu = {A}
F = son durumlar kümesi ={B}
$\delta = Q \times \Sigma \rightarrow 2^Q$

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

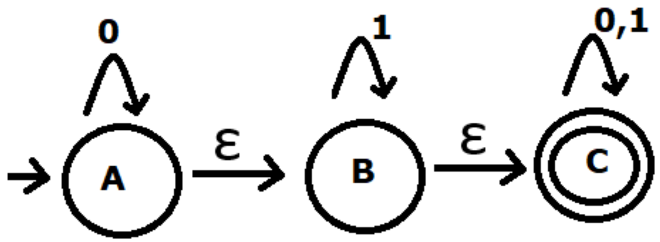
birleşim ε

B hiçbir input gelmediği zaman C'ye gider

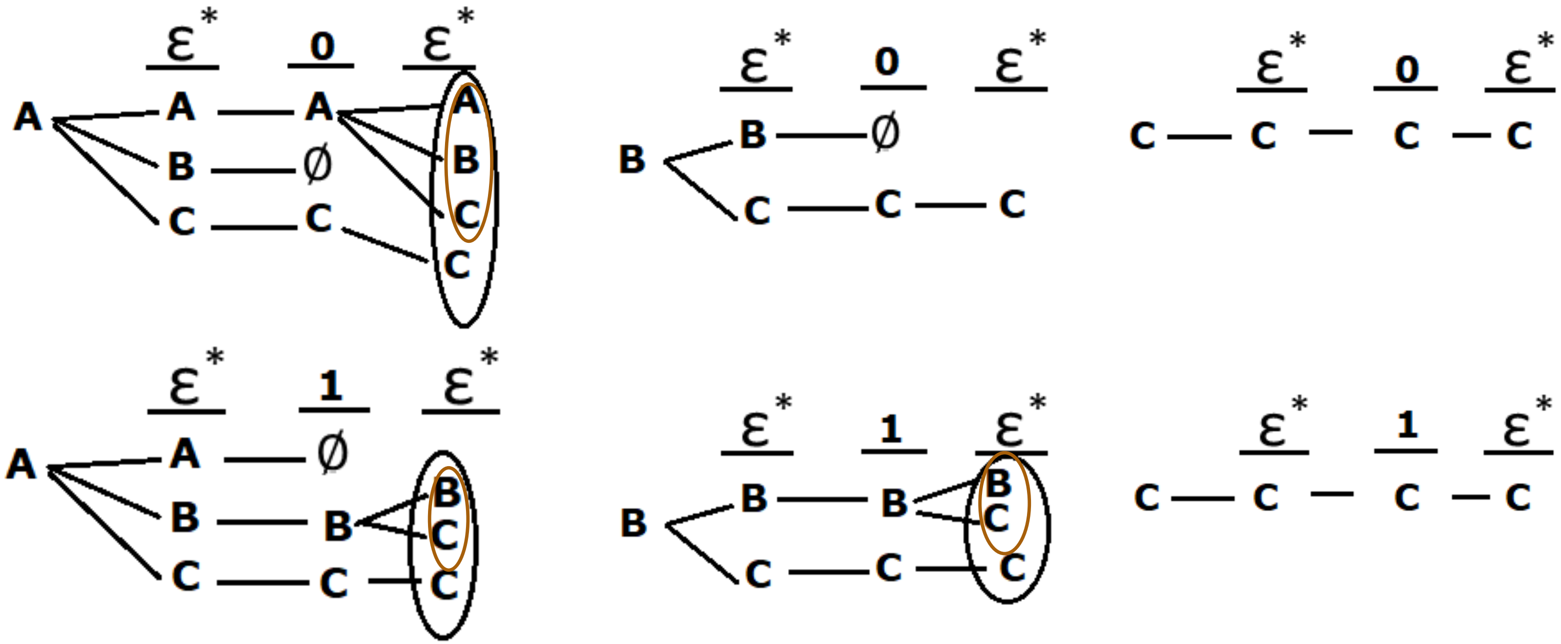


ε üzerindeki her durum kendisine gider

Örnek 1



ε^* : Bir durum üzerinden sadece ε 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 ε üzerinden erişebilenlerde son durum olur (A,B)

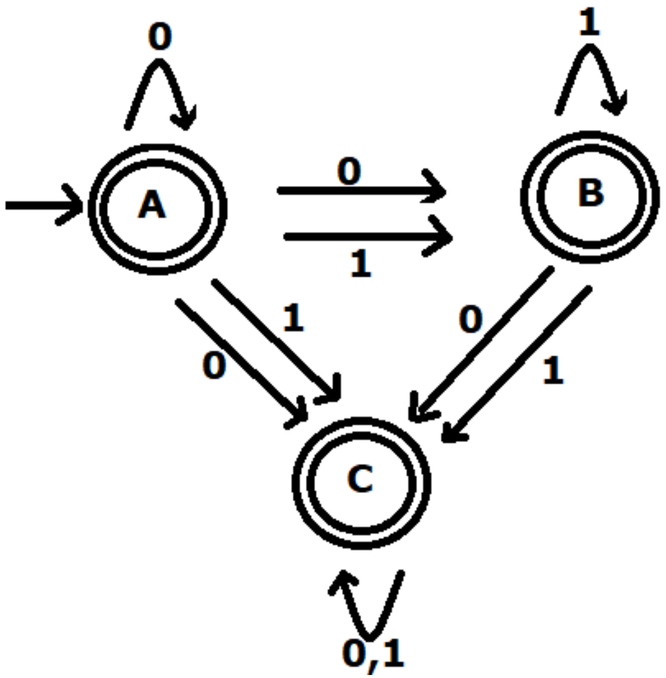
Örnek 1 çözüm

Ɛ'dan – NFA çevirimi

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

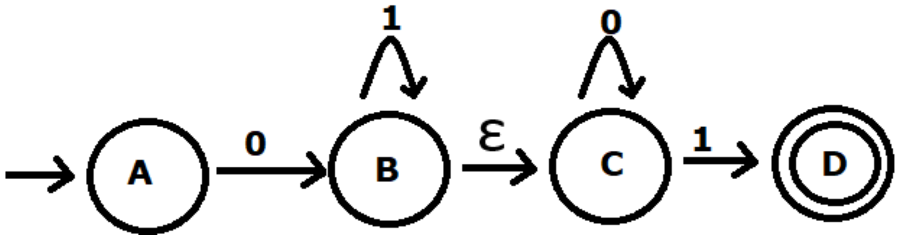
Son durumlar hesaplanırken,

C zaten son durum. C'ye Ɛ üzerinden erişebilenlerde son durum olur (A,B)



NFA

Örnek 2

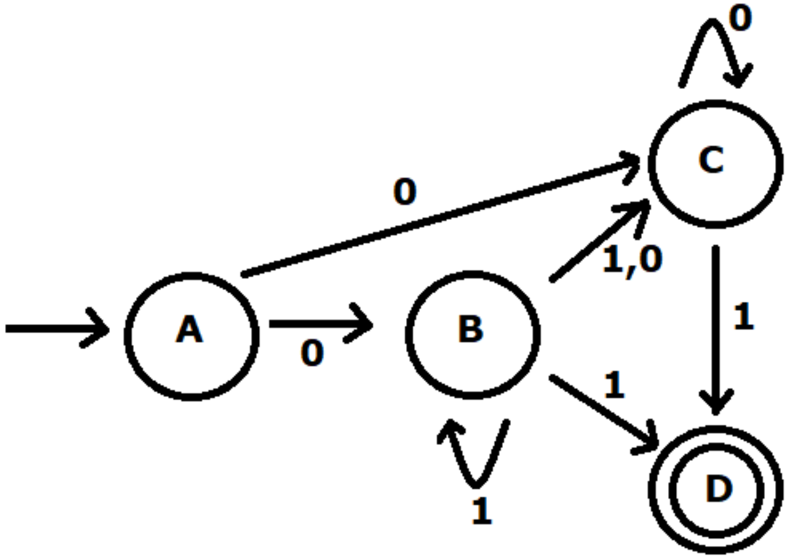


ε - NFA

<table>
<tr><th></th><th>ε*</th><th>0</th><th>ε*</th></tr>
<tr><td rowspan="2">A</td><td rowspan="2">A</td><td rowspan="2">B</td><td>B</td></tr>
<tr><td>C</td></tr>
<tr><td rowspan="2">B</td><td>B</td><td>∅</td><td>∅</td></tr>
<tr><td>C</td><td>C</td><td>C</td></tr>
<tr><td>C</td><td>C</td><td>C</td><td>C</td></tr>
<tr><td>D</td><td>D</td><td>∅</td><td>∅</td></tr>
</table>

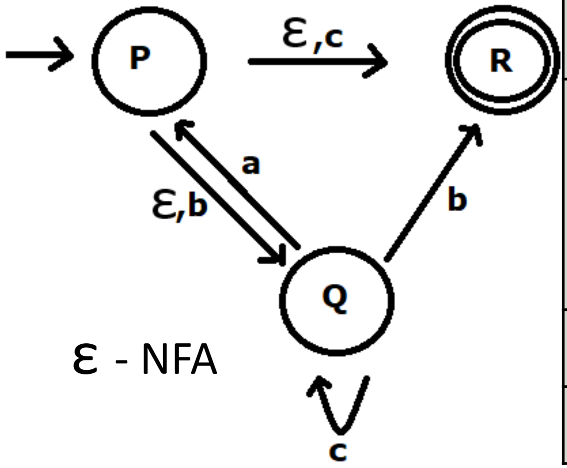
<table>
<tr><th></th><th>ε*</th><th>1</th><th>ε*</th></tr>
<tr><td>A</td><td>A</td><td>∅</td><td>∅</td></tr>
<tr><td rowspan="3">B</td><td rowspan="2">B</td><td rowspan="2">B</td><td>B</td></tr>
<tr><td>C</td></tr>
<tr><td>C</td><td>D</td><td>D</td></tr>
<tr><td>C</td><td>C</td><td>D</td><td>D</td></tr>
<tr><td>D</td><td>D</td><td>∅</td><td>∅</td></tr>
</table>

	0	**1**
A	B,C	∅
B	C	B,C,D
C	C	D
D	∅	∅



NFA

Örnek 3



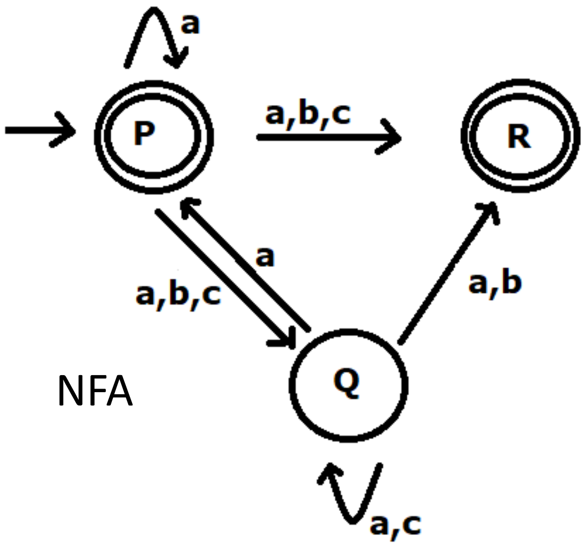
ε - NFA

	ε^*	a	ε^*
	P	Ø	Ø
P	Q	P	P,Q,R
	R	Ø	Ø
Q	Q	P	P,Q,R
R	R	Ø	Ø

	ε^*	b	ε^*
	P	Q	Q
P	Q	R	R
	R	Ø	Ø
Q	Q	R	R
R	R	Ø	Ø

	ε^*	c	ε^*
	P	R	R
P	Q	Q	Q
	R	Ø	Ø
Q	Q	Q	Q
R	R	Ø	Ø

	a	b	c
P	P,Q,R	Q,R	R,Q
Q	P,Q,R	R	Q
R	Ø	Ø	Ø

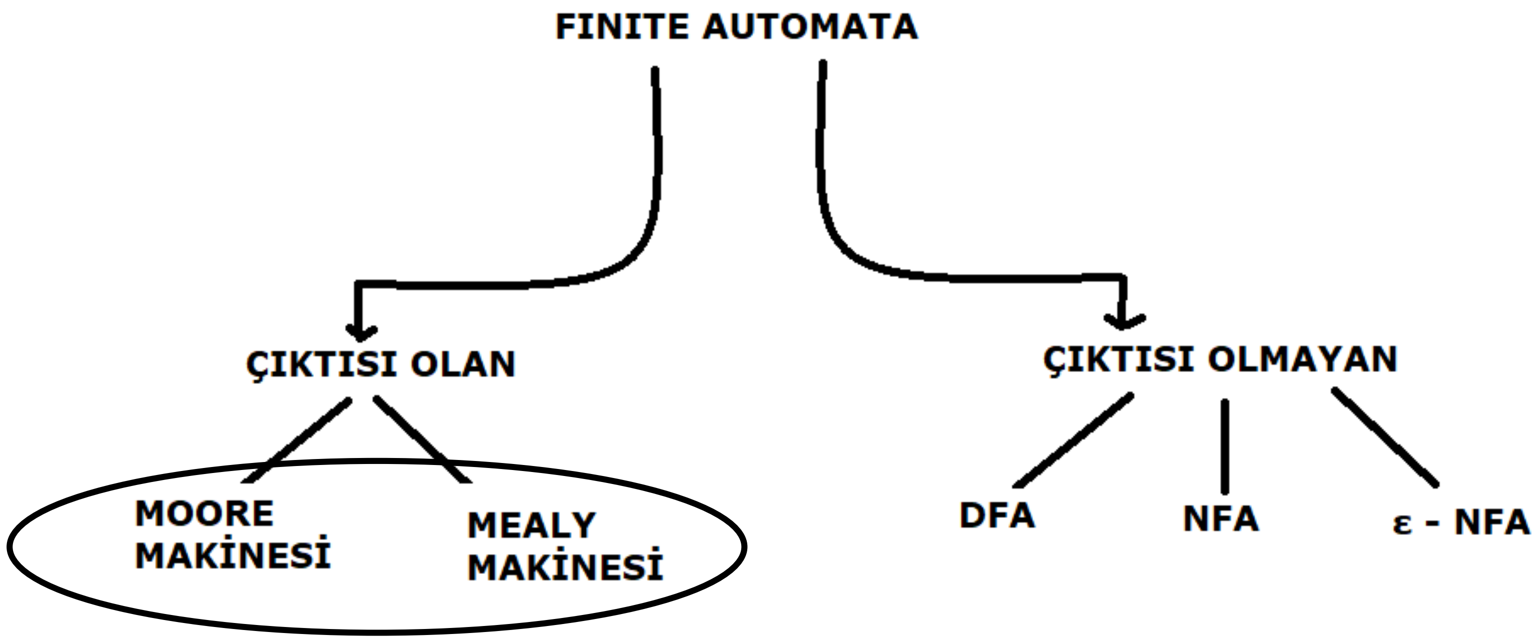


NFA

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

Σ = sonlu boş olmayan giriş alfabe kümesi

Δ = çıkış alfabe kümesi

δ = geçiş fonksiyonu $Q \times \Sigma \rightarrow Q$

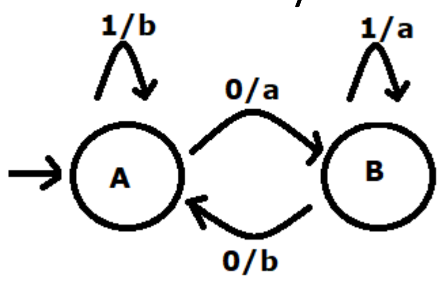
λ = çıkış fonksiyonu

q_0= başlangıç durumu

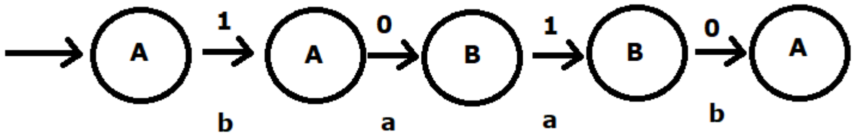
$Q \rightarrow \Delta$

Mealy – Moore Makineleri

Mealy



örn : 1 0 1 0



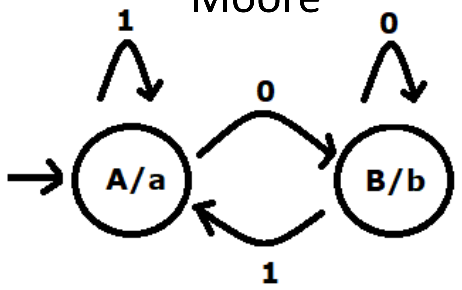
baab

n→n

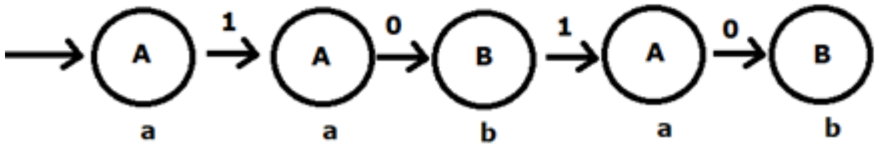
giriş uzunluğu

cıkış uzunluğu

Moore



örn : 1 0 1 0



aabab

n→n+1

giriş uzunluğu

cıkış uzunluğu

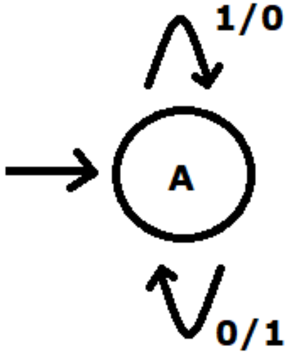
(başlangıç durumundan dolayı)

Mealy Makinesi

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

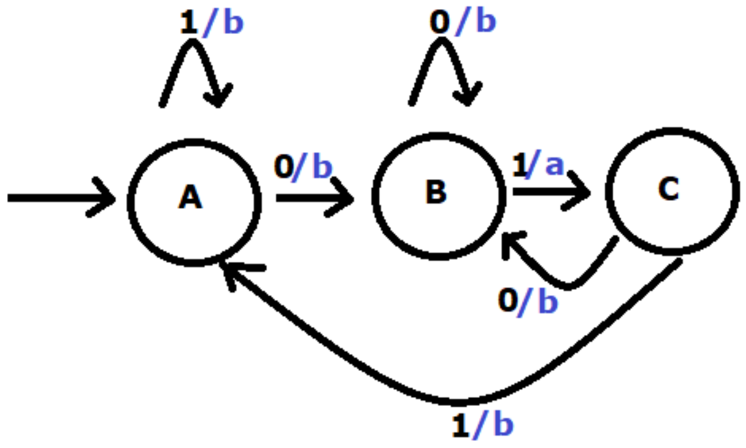
1 0 1 0 0
0 1 0 1 1

1'e tümleme



Mealy Makinesi

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

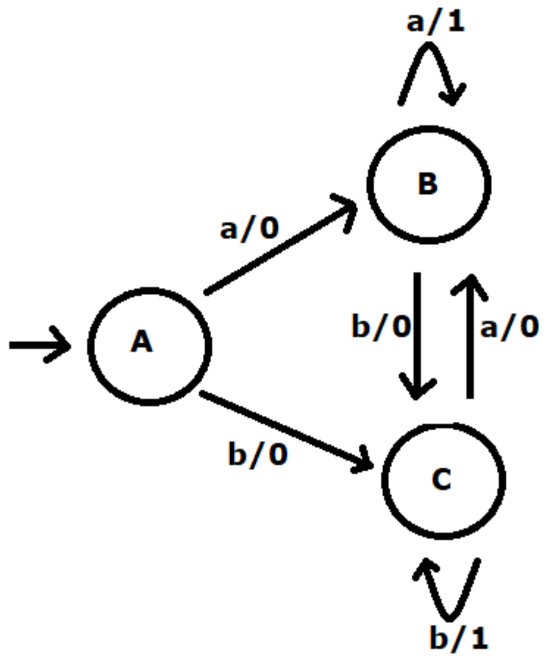


ilk başta DFA gibi tasarlıyoruz.

0110 → babb
1000 → bbbb

Mealy Makinesi

Örn: Σ = {a,b} alfabesinde ‘aa’ veya ‘bb’ içeren giriş geldiğinde 1 basan makineyi tasarlayın.



aaa →011
aab →010

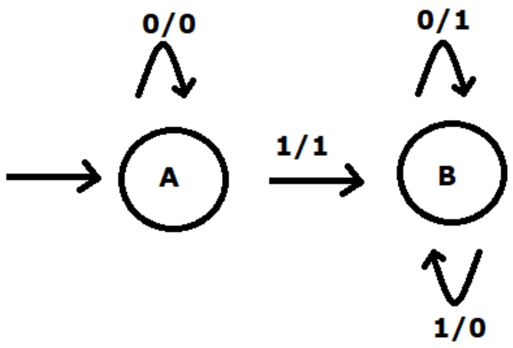
Mealy Makinesi

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

2’ye tümleyen = 1’e tümleyen + 1

* sondan başlayarak ilk 1 görene kadar aynı, 1’i gördükten sonra tersini alınır

1 0 1 0 0 → 0 1 1 0 0   1 1 1 0 0 → 0 0 1 0 0   1 1 1 1 1 → 0 0 0 0 1

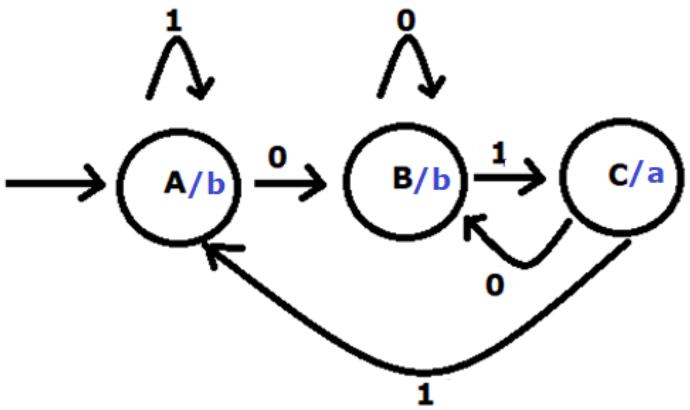


1	0	1	0	0
B	B	A	A	A
0	1	1	0	0

* sondan başlayarak düşünerek

Moore Makinesi

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



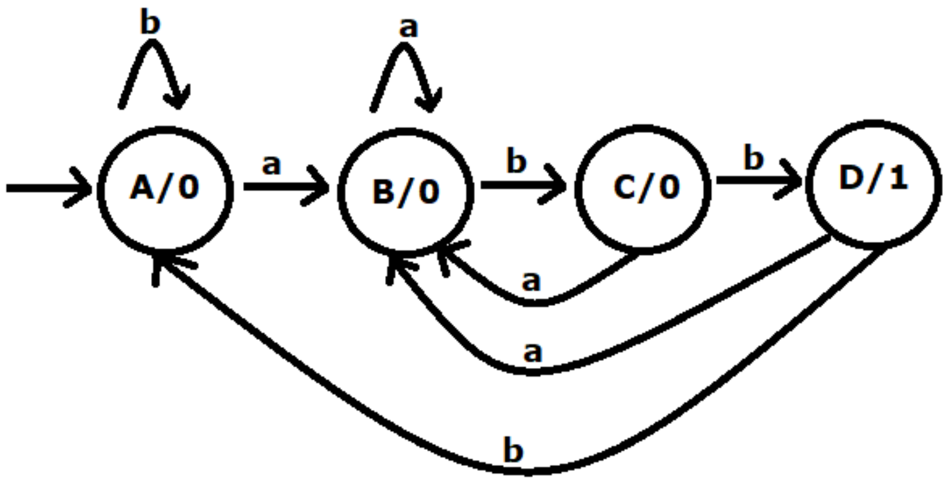
ilk başta DFA gibi tasarlıyoruz.

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

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

Moore Makinesi

Örn : Σ={a,b} giriş alfabesinde ‘abb’ sıralamasının kaç defa geçtiğini sayan Moore makinesini tasarlayın. Δ={0,1}



ilk başta DFA gibi tasarlıyoruz.

Moore Makinesi

Aşağıda geçiş tablosu ve çıktıları verilmiş Moore Makinesi için giriş alfabesi Σ={a,b} ve çıkış alfabesi Δ={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**

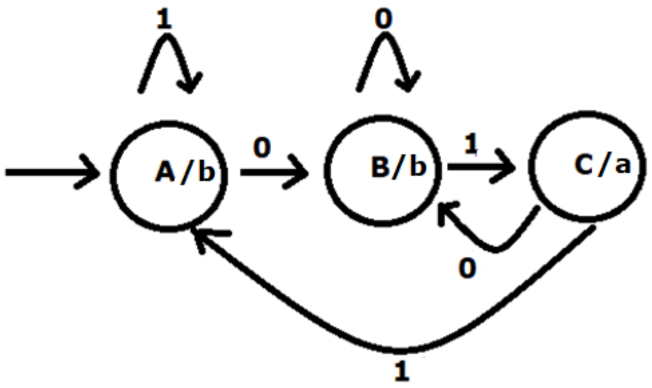
	a	**b**	**a**	**b**	**b**
q0	q1	q3	q4	q0	q2
0	**0**	**0**	**0**	**0**	**1**

Mealy $\leftrightarrow$ Moore Çevirimleri

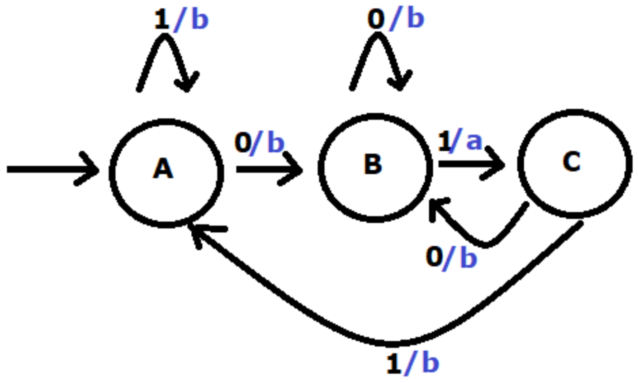
DR. ŞAFAK KAYIKÇI

Moore → Mealy çevirimi

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



gittiği durumdaki output değerini geçişe yazıyoruz.

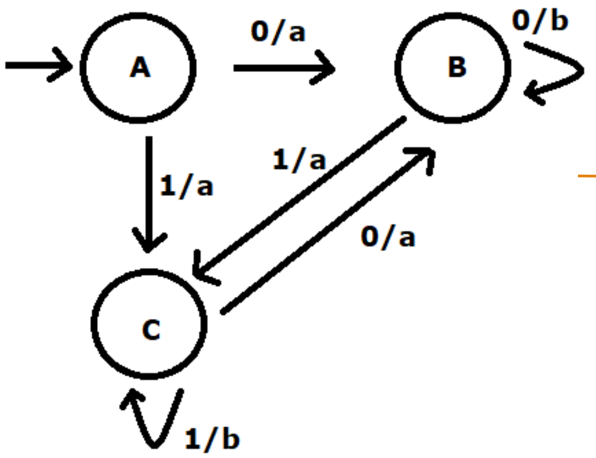


durum	0	1	Çıktı
→A	B	A	b
B	B	C	b
C	B	A	a

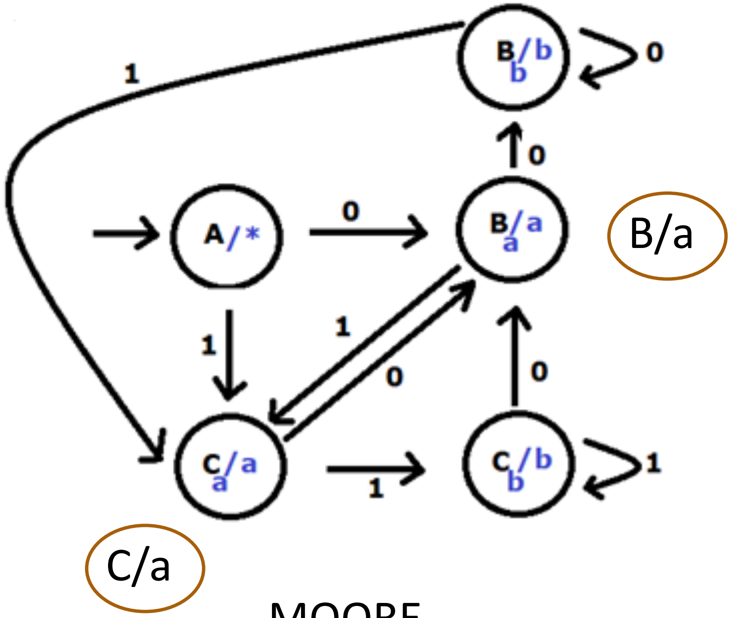
durum	0	1
→A	B,b	A,b
B	B,b	C,a
C	B,b	A,b

Mealy → Moore Çevirimi

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



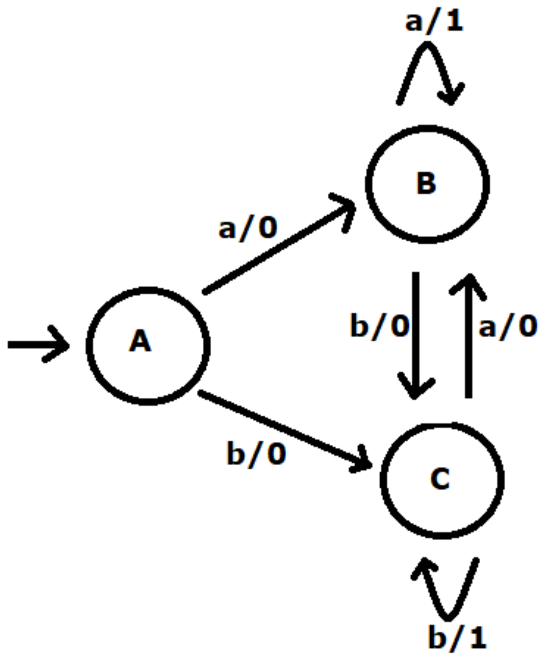
MEALY



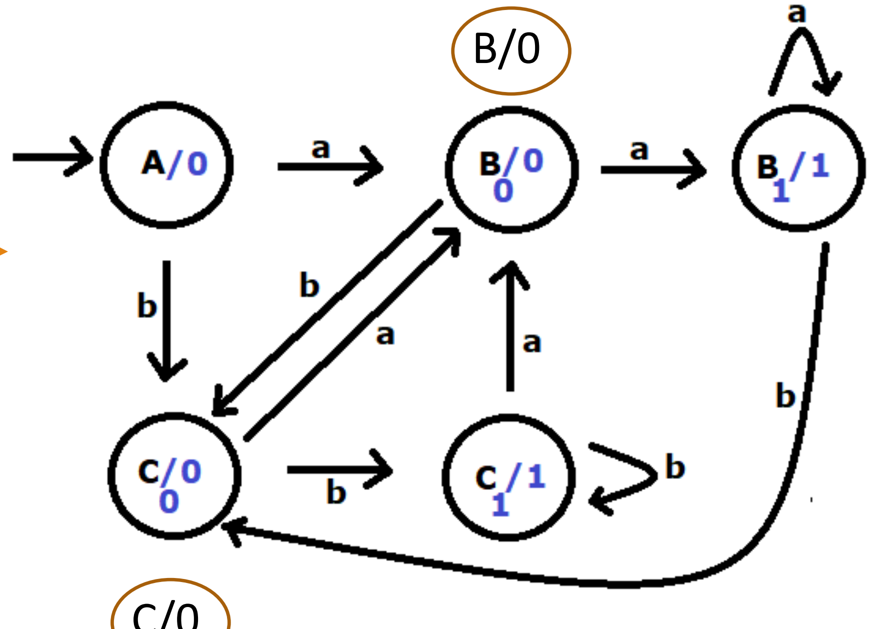
MOORE

Mealy → Moore Çevirimi

Örn: Σ = {a,b} alfabesinde ‘aa’ veya ‘bb’ içeren giriş geldiğinde 1 basan Mealy Makinesini Moore Makinesine çevirin. Δ={0,1}



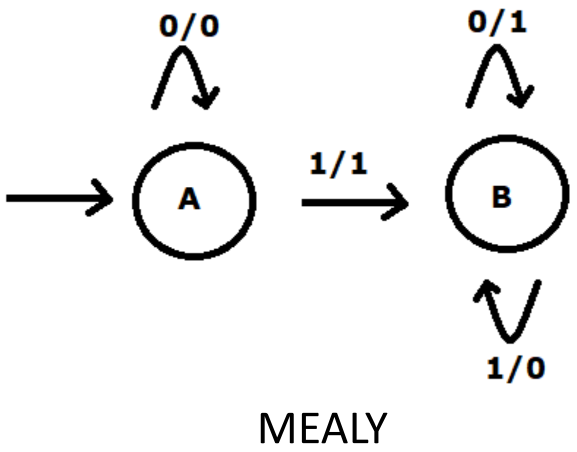
MEALY

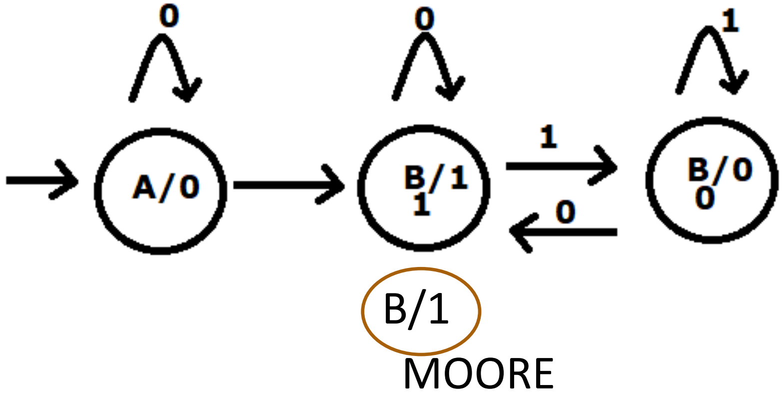


MOORE

Mealy → 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 ifedeler (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 | 1 | 2

veya

Örnek 2: {abb, a, b, bba}

R= abb + a + b + bba

Örnek 3: {ε, 0, 00, 000, ...}

R = 0^*

Örnek 4: {1, 11, 111, 1111, ...}

R = 1^+

Σ={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, ...}

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

Örnek 7: en fazla iki boyutlu stringleri kabul eden dil

L = {ε, a, b, aa, ab, ba, bb}

R = ε + a + b + aa + ab + ba + bb

= (ε+a+b) (ε+a+b)

Düzenli İfadelerde Özellikler

1) $\emptyset + R = R$

2) $\varepsilon R = R\varepsilon = R$

3) $\varepsilon^* = \varepsilon$ ve $\emptyset^* = \varepsilon$

4) $R + R = R$

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

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

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

8) $\varepsilon + RR^* = \varepsilon + R^*R = R^*$

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

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

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

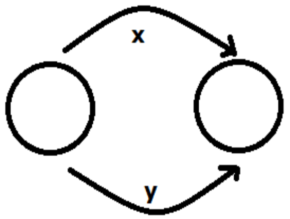
12) $R = Q+RP \rightarrow R = QP^*$ (Arden Teoremi)

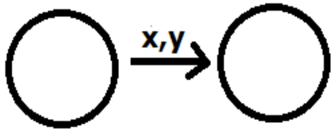
ispat: R yerine denklemde Q+RP koyalım

$R = Q + RP$
$R = Q + (Q+RP)\, P$
$R = Q + QP + RPP$
$\quad = Q + QP + (Q+RP)PP$
$\quad = Q + QP + QPP + RPPP$
....
$\quad = Q\, (\varepsilon + P + PP + PPP + \ldots)$
$\quad = QP^*$

Düzenli İfadeden Sonlu Otomata Çevirim

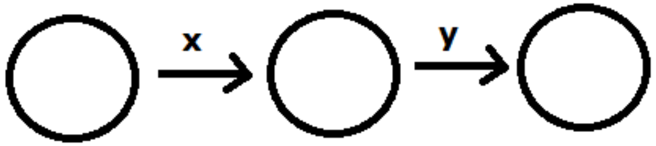
X | Y
X + Y



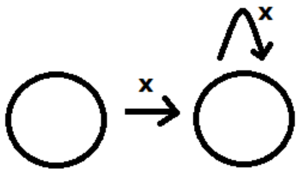


X^*

XY



$X^+ = XX^*$



Örnekler

ab^*a

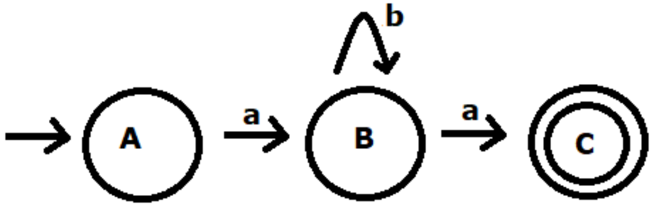
aa

aba

abba

abbba

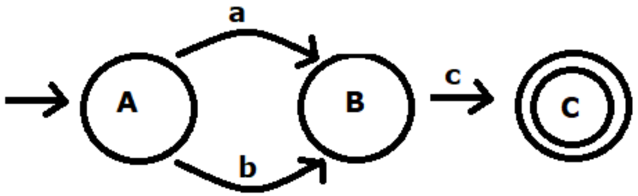
....



$(a|b)c$

ac

bc



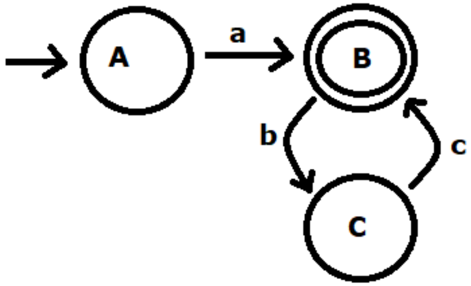
$a(bc)^*$

a

abc

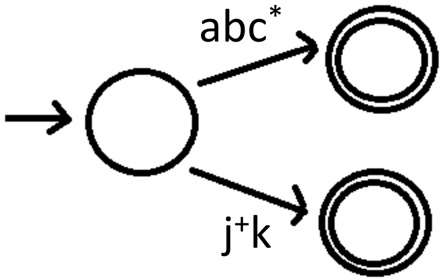
abcbc

....

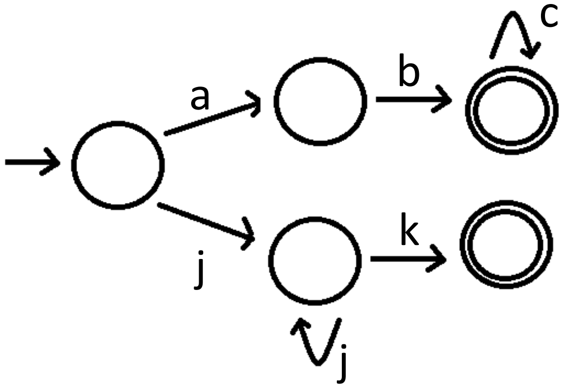


Örnekler

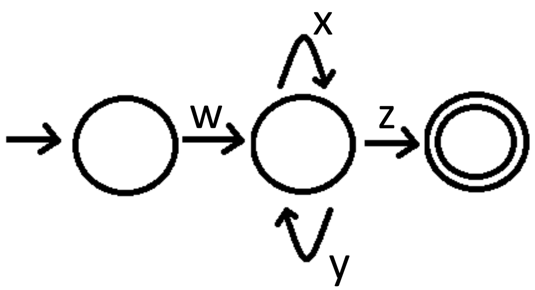
$abc^* \mid j^+k$



ab
abc
abcccc
jk
jjk
jjjk



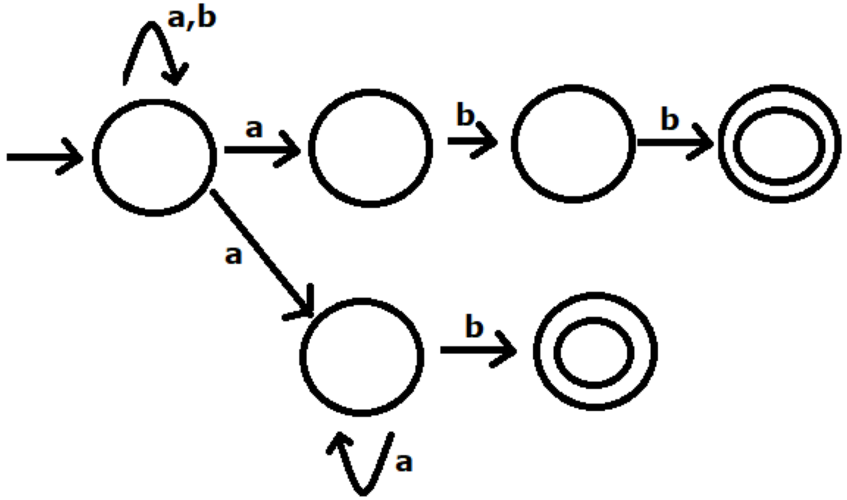
$w(x|y)^*z$



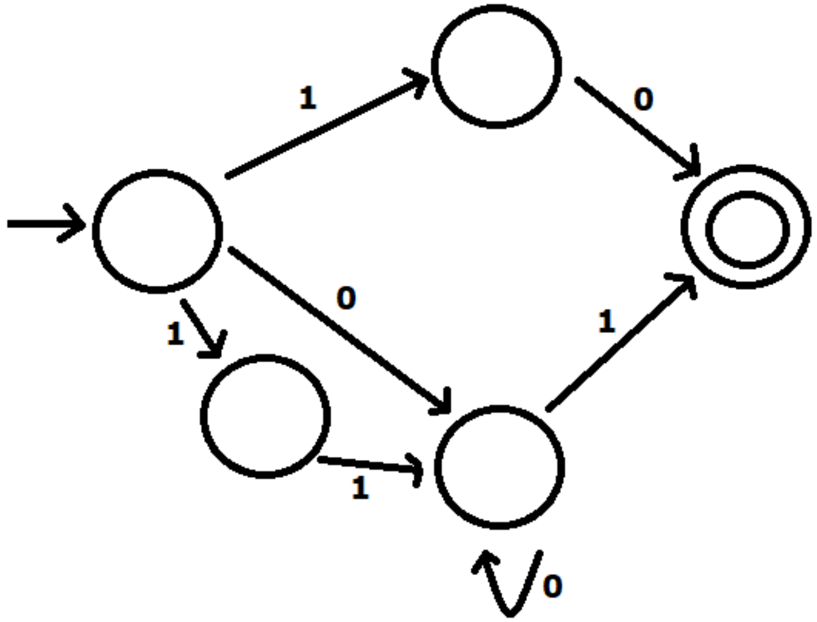
wz
wxz
wyz
wxxz
wxyz
wxyxz

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

$A \rightarrow x\,B$

$A \rightarrow x$

$A,B \in V$ ve $x \in T$

(non-terminal sembol sağda)

Örn : $S \rightarrow abS \mid b$

Left Linear Grammer

$A \rightarrow B\,x$

$A \rightarrow x$

$A,B \in V$ ve $x \in T$

(non-terminal sembol solda)

Örn : $S \rightarrow Sab \mid b$

Düzenli Gramerler

G = ({S, A, B}, {a, b}, S, {S→AB, A→a, B→b})

V = {S, A, B}

T = {a, b}

S = S

P = S→AB, A→a, B→b

S→AB
→aB
→ab

Düzenli Diller

Bir gramerden türetilebilen <u>bütün</u> stringlere, o gramerin <u>dili</u> denir.

G = ({S, A}, {a, b}, S, {S→aAb, aA→aaAb, A→ϵ})

S → <u>aA</u>b (S→aAb kuralı)

→ a<u>aA</u>bb (aA→aaAB kuralı)

→aaaAbbb (A→ ϵ kuralı)

→aaabbb

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

Düzenli Diller

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

$S \rightarrow AB$
$\rightarrow ab$

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

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

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

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

İçerik Bağımsız Diller
Context Free Languages(CFG)

Dört bileşen ile ifade edilir. G=(V, Σ, S, P)

V = değişkenler kümesi (non-terminaller)
Σ = semboller kümesi (terminaller)
S = başlangıç sembolü
P = terminaller ve non-terminaller için yapım (production) kuralları

A→ a , a = $\{V \cup \Sigma\}^*$ ve A∈ 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 kalmayıncaya kadar devam edilir.

İçerik Bağımsız Diller

S→0B|1A, A→0|0S|1AA|ϵ, B→1|1S|0BB gramerinin 00110101 stringini ürettiğini gösteriniz.

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

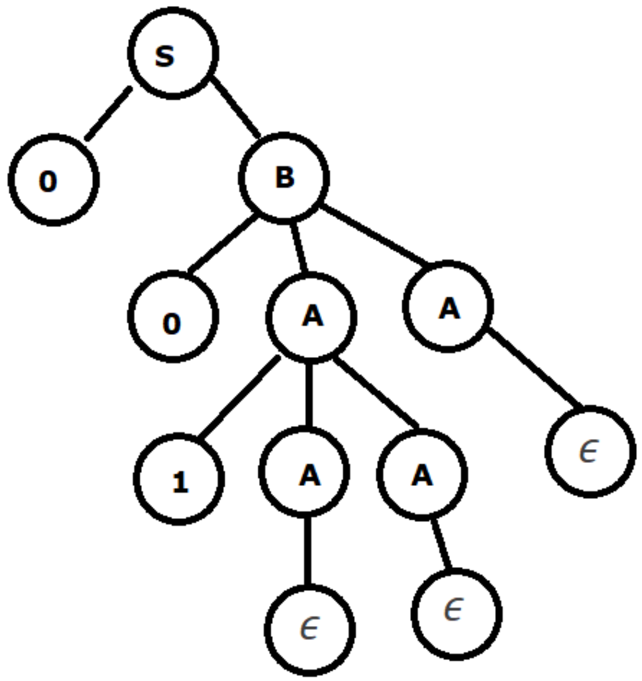
Derivation (parse) Tree

G=(V, T,S, P) S→0B, A→1AA|ϵ, B→0AA

kök düğüm (root vertex) : başlangıç sembolü

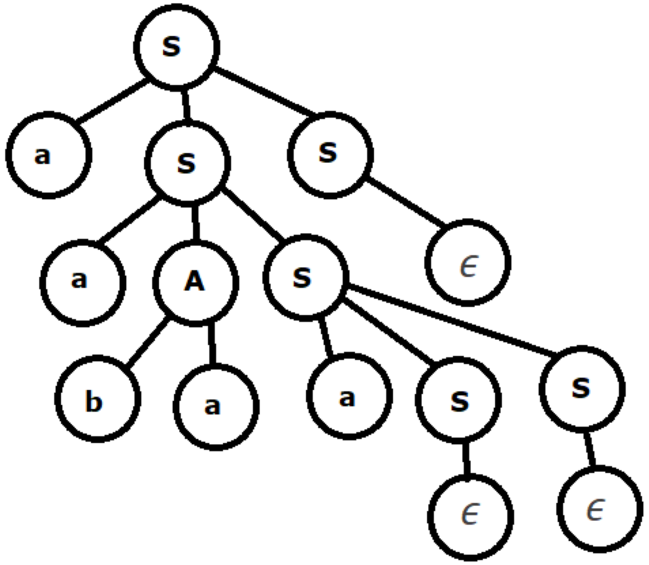
düğüm (vertex) : non-terminal semboller

yaprak (leaves) : terminal semboller yada ϵ

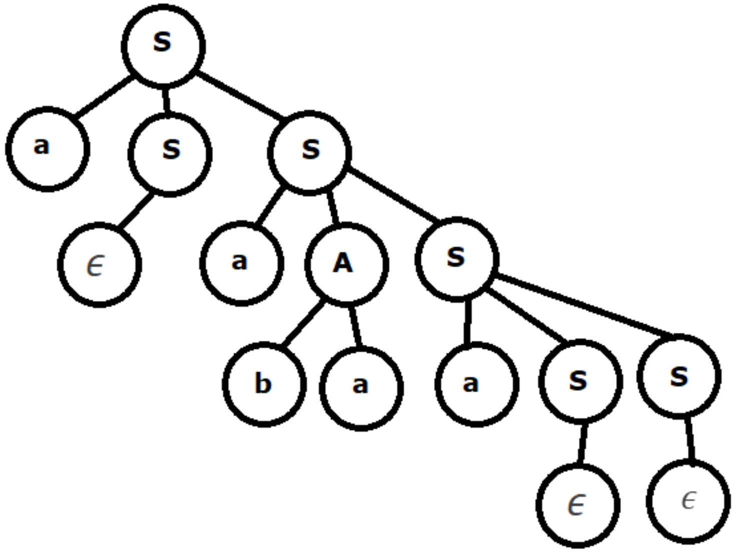


Left Derivation – Right Derivation

S→aAS|aSS|ϵ, A→SbA|ba kurallarında <u>aabaa</u> için parse tree



left derivation tree

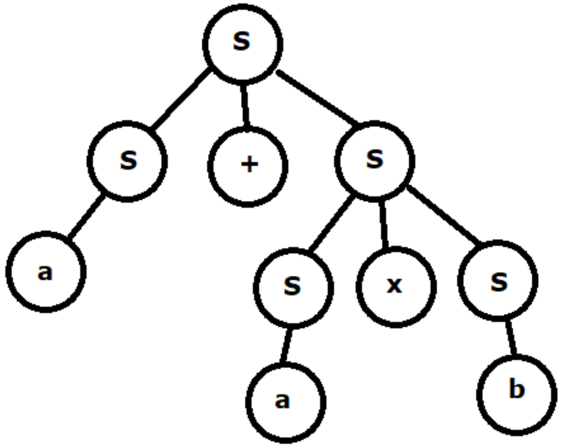


right derivation tree

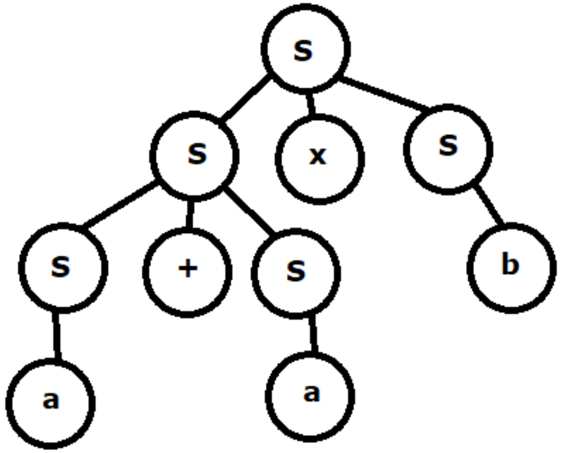
Belirsiz (ambiguous) Gramer

G=({S}, {a, b, +, x}, P, S)   S→S+S|SxS|a|b   string : a+axb

S→S+S
→a+S
→a+SxS
→a+axS
→a+axb



S→SxS
→S+SxS
→a+SxS
→a+axS
→a+axb



Pushdown Automata (PDA)

DR. ŞAFAK KAYIKÇI

Pushdown Automata

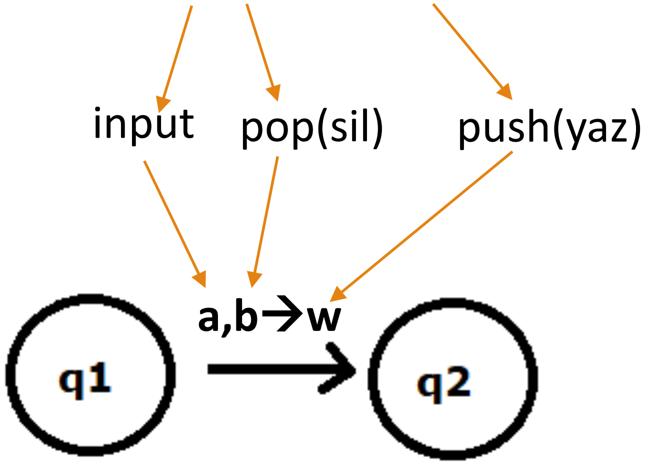
PDA, yığını (stack) olan bir NFA gibidir

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

Q = durumlar kümesi
Σ = giriş alfabesi
Γ = yığın alfabesi
δ = geçiş fonksiyonu
q_0 = başlangıç durumu
Z = yığın sembolü ($)
F = son durum

Pushdown Automata

$$\delta(q1, a, b) = \{(q2,w)\}$$



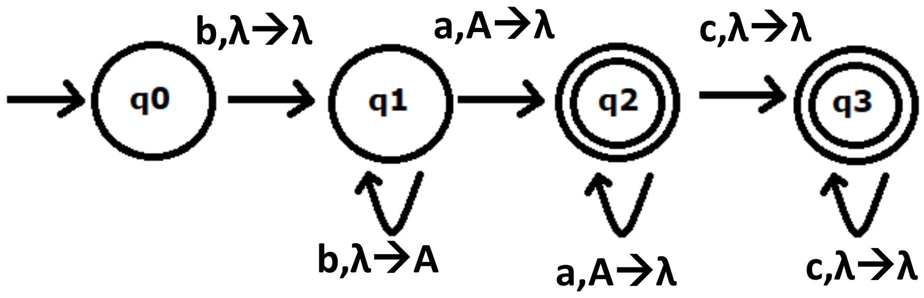
Bitiş durumu son durumda ve stack boş ise string kabul edilir.

Örnek

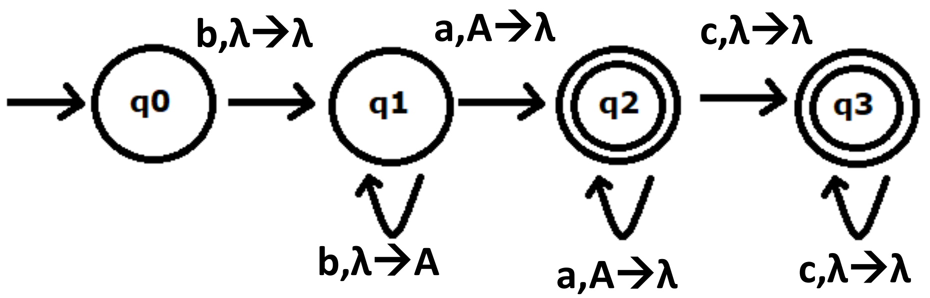
$M = Q, \Sigma, \Gamma, \delta, q_0, Z, F$

$= (\{q_0, q_1, q_2, q_3\}, \{a, b, c\}, \{A\}, \delta, q_0, \$, \{q_2, q_3\})$

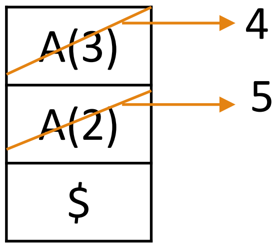
$\delta(q_0, b, \lambda) = \{(q_1, \lambda)\}$
$\delta(q_1, b, \lambda) = \{(q_1, A)\}$
$\delta(q_1, a, A) = \{(q_2, \lambda)\}$
$\delta(q_2, a, A) = \{(q_2, \lambda)\}$
$\delta(q_2, c, \lambda) = \{(q_3, \lambda)\}$
$\delta(q_3, c, \lambda) = \{(q_3, \lambda)\}$



Örnek - devam



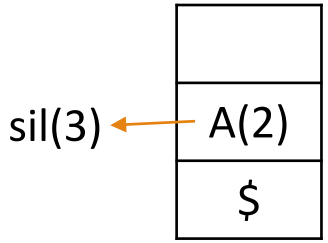
bbbaacc



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

Bitiş durumu son durum ve stack boş.
Kabul edilir.

bbaac

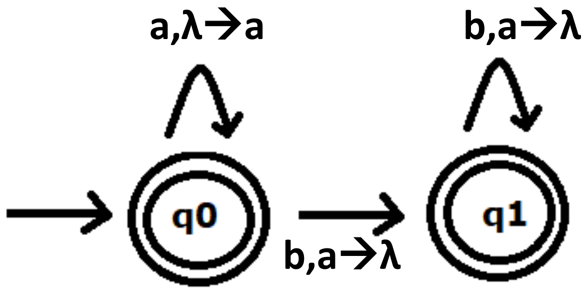


b - 1) boş (λ sil, λ yaz)
b - 2) A yaz
a - 3) A sil
a - 4) A sil (yığında yok)

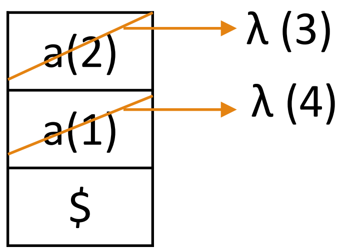
Kabul edilmez

Örnek

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



aabb



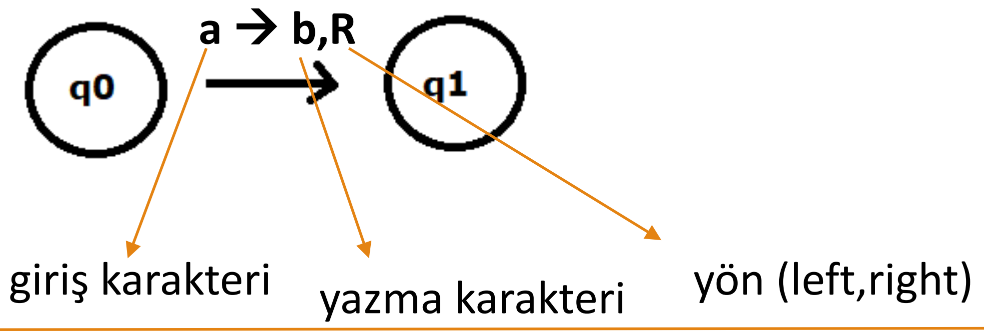
Bitiş durumu son durum ve stack boş.
Kabul edilir.

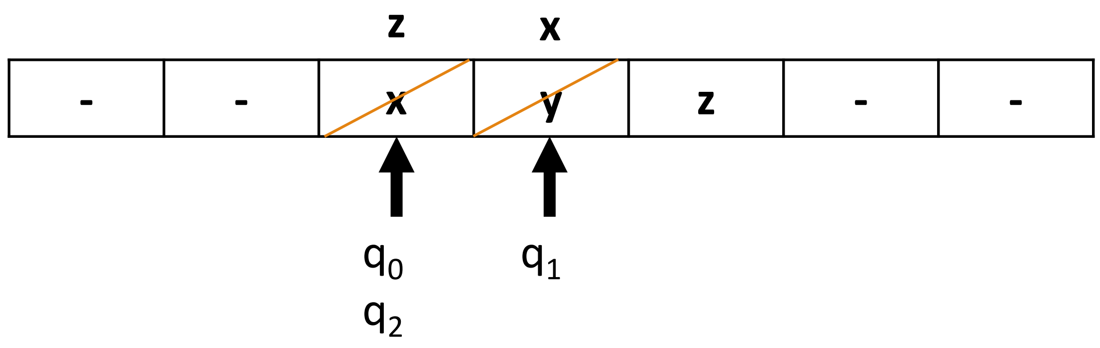
Turing Makinesi

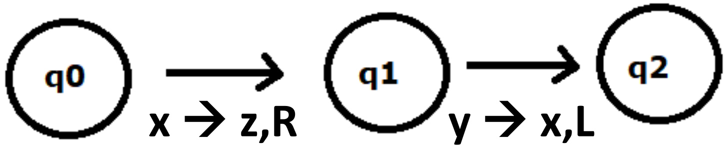
DR. ŞAFAK KAYIKÇI

Turing Makinesi

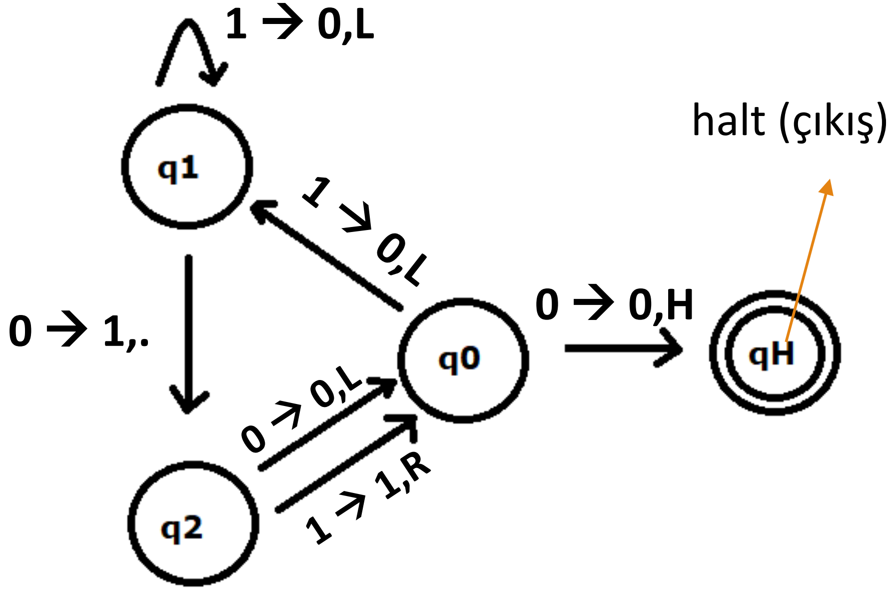
Turing Makinesi, random access memory kullanılır.

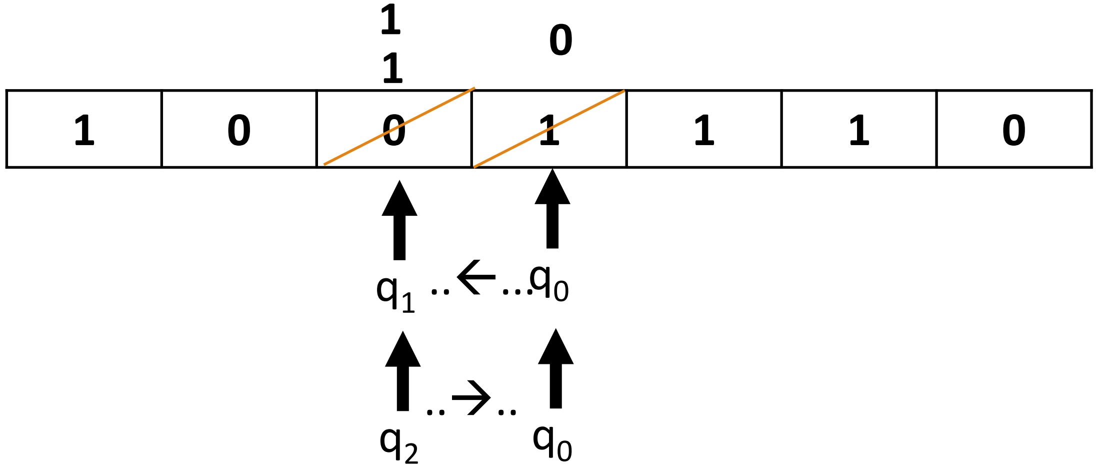




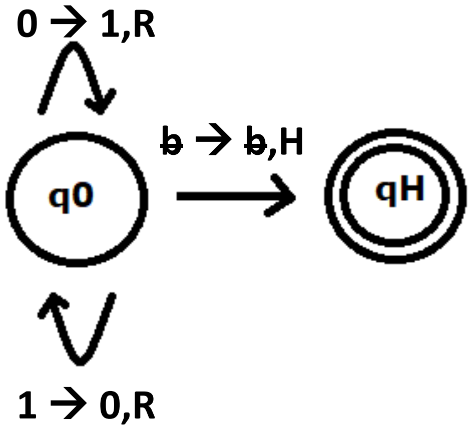


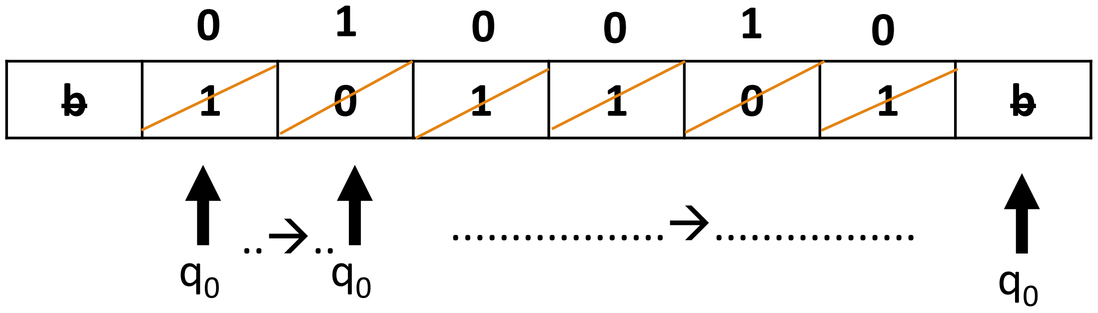
Örnek


1 → 0,L
q1
1 → 0,L
0 → 1,.
q0
0 → 0,H
qH
halt (çıkış)
0 → 0,L
1 → 1,R
q2


1
1
0
1
0
0
1
1
1
0
q1 ..←...q0
..→..
q2
q0

Örn: 1’e tümleyen



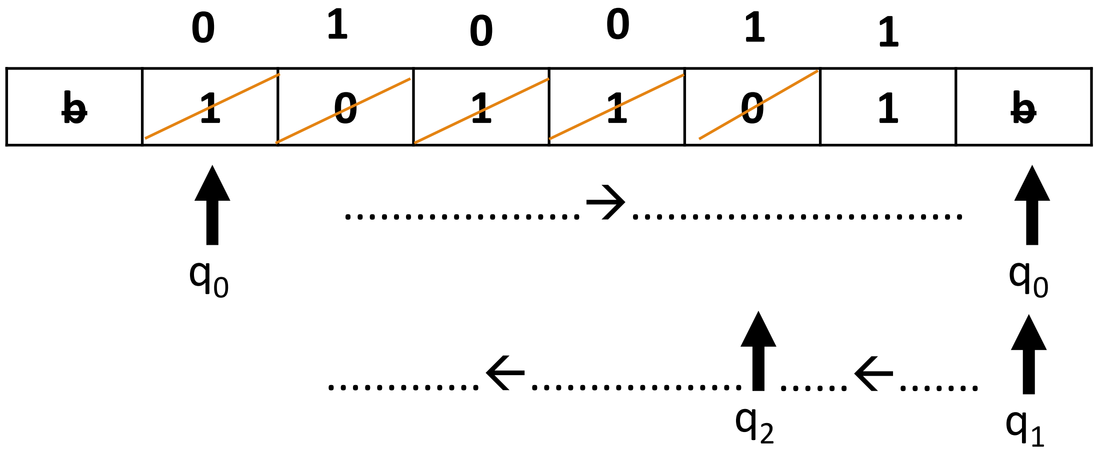


$\delta(q_0, 0) = (q_0, 1, R)$

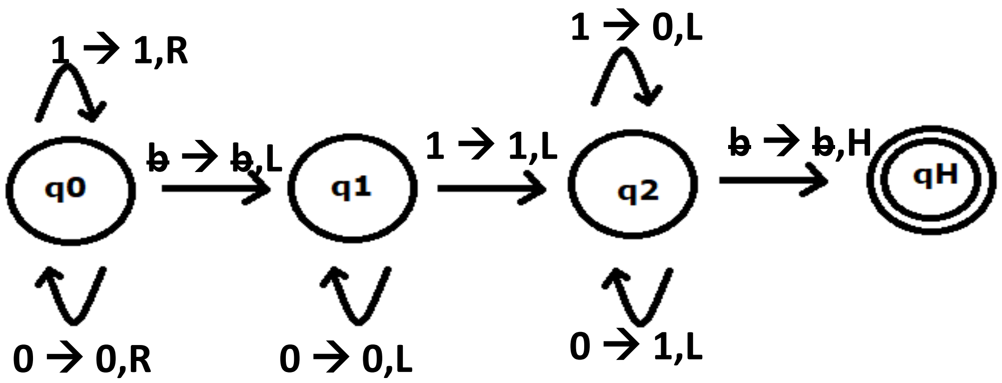
$\delta(q_0, 1) = (q_0, 0, R)$

$\delta(q_0, \text{ƀ}) = (q_0, \text{ƀ}, \text{halt})$

Örn: 2’e tümleyen



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



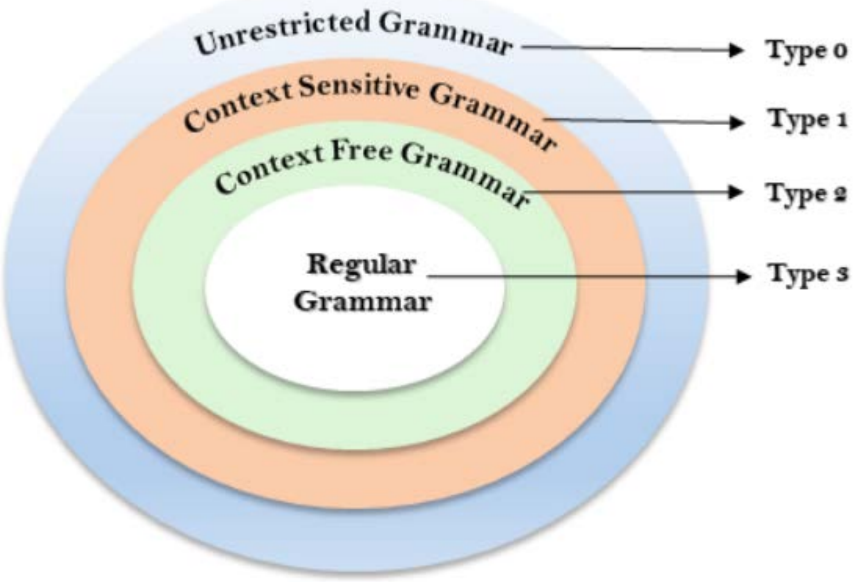
Chomsky Hiyerarşisi

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 modellenebilir.

Örneğin:

bAa → aa

S → 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 → ε formunun kuralına izin verilmez. Herhangi bir kuralın sağ tarafında meydana gelmez.
- Tip 1 dilbilgisi Tip 0 olmalıdır. Tip 1'de Üretim V → T şeklindedir. V'deki sembol sayısı T'den küçük veya ona eşittir.

Örneğin:

S → AT

T → xy

A → 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 modellenebilir.

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

Tip 3 formu şu şekildedir :

V → T * V / T *

Örneğin:

A → xy

Genel Gösterim

Grammar	Languages	Automaton	Production rules (constraints)*	Examples[3]
Type-0	Recursively enumerable	Turing machine	$\alpha A\beta \rightarrow \delta$ (no constraints)	$L = \{w \mid w$ describes a terminating Turing machine$\}$
Type-1	Context-sensitive	Linear-bounded non-deterministic Turing machine	$\alpha A\beta \rightarrow \alpha\gamma\beta$	$L = \{a^n b^n c^n \mid n > 0\}$
Type-2	Context-free	Non-deterministic pushdown automaton	$A \rightarrow \alpha$	$L = \{a^n b^n \mid n > 0\}$
Type-3	Regular	Finite state automaton	$A \rightarrow \mathrm{a}$ and $A \rightarrow \mathrm{a}B$	$L = \{a^n \mid n \geq 0\}$

* Meaning of symbols:

- a = terminal
- A, B = non-terminal
- α, β, γ, δ = string of terminals and/or non-terminals
    - α, β, δ = maybe empty
    - γ = never empty

JLAP Kullanımı

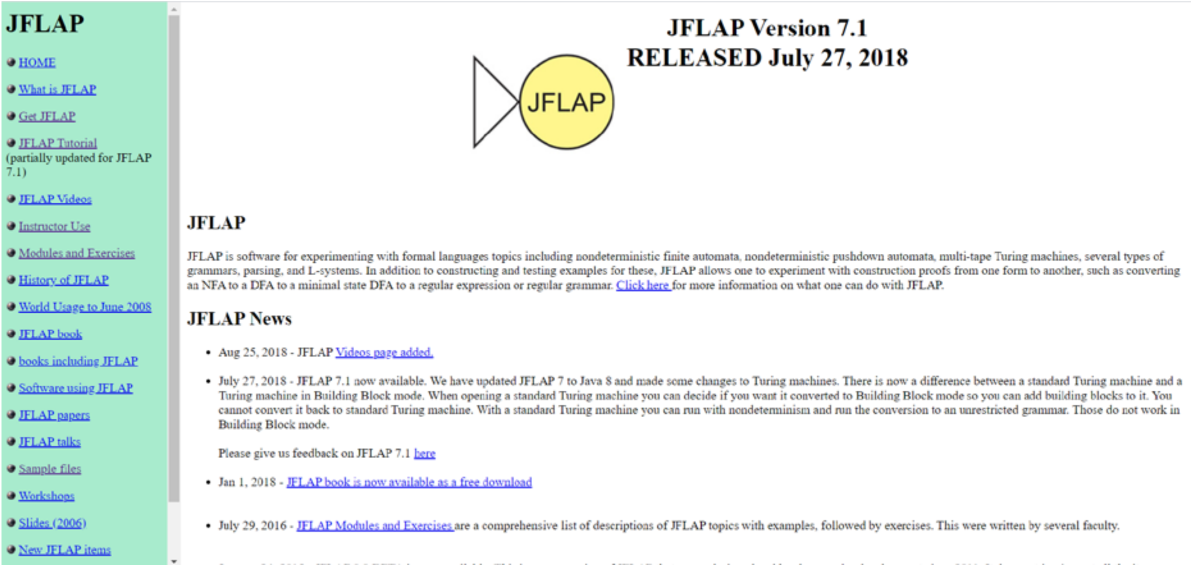
DR. ŞAFAK KAYIKÇI

JLAP

http://www.jflap.org/

download
http://www.jflap.org/jflaptmp/

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

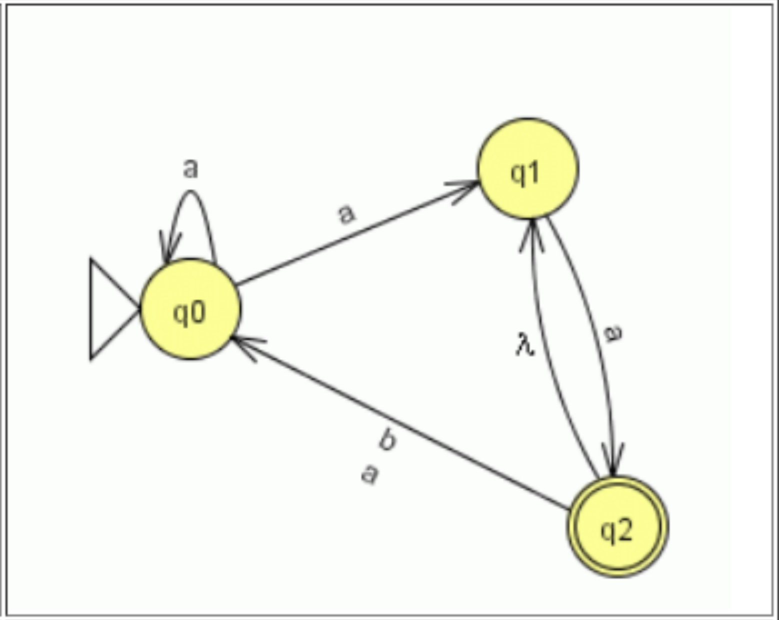


JFLAP, belirli - belirsiz sonlu otomata, push down otomata, Turing makineleri, düzenli gramerler, içerik bağımsız diller konularını denemek için kullanabileceğiniz bir yazılımdır.

Düzenli Dillerin Oluşturulması

Regular languages - create

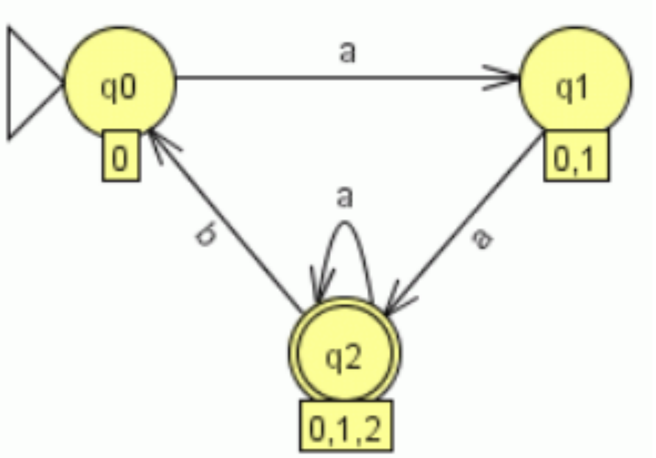
- **DFA**
- **NFA**
- **regular grammar**
- **regular expression**



Düzenli Diller - Dönüşümler

Regular languages - conversions

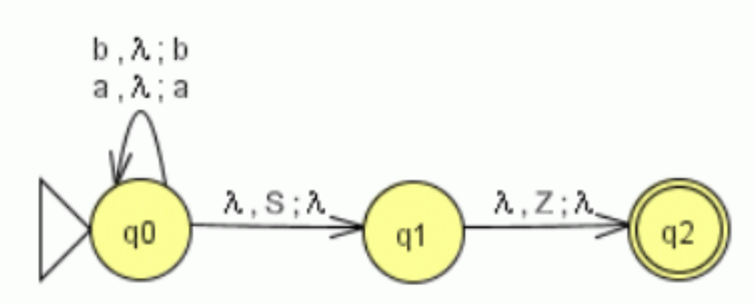
- **NFA -> DFA -> Minimal DFA**
- **NFA <-> regular expression**
- **NFA <-> regular grammar**



İçerik Bağımsız Dillerin Oluşturulması

Context-free languages - create

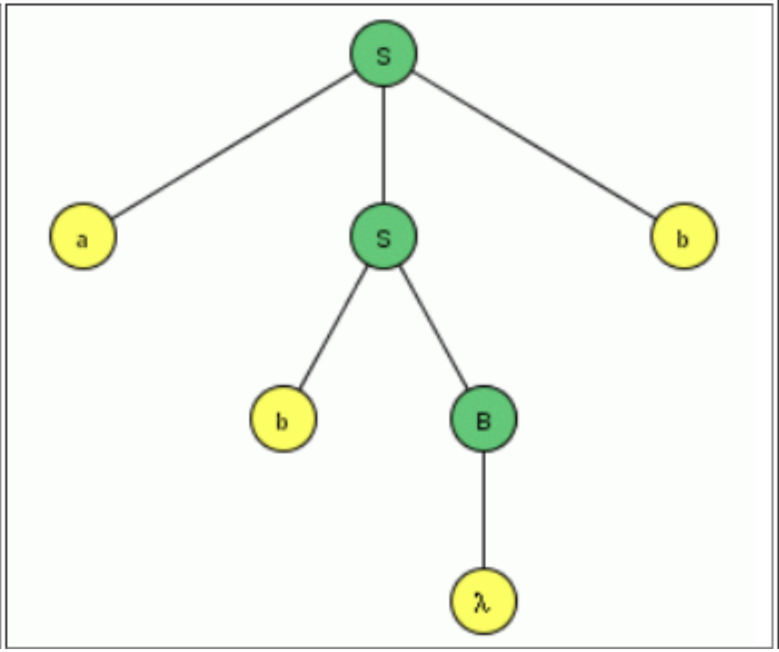
- **push-down automaton**
- **context-free grammar**



İçerik Bağımsız Diller - Dönüşümler

Context-free languages - transform

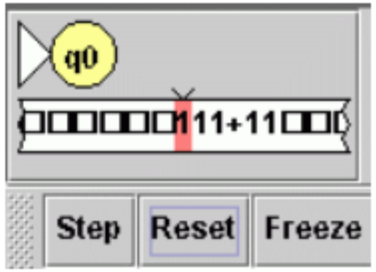
- **PDA -> CFG**
- **CFG -> PDA (LL parser)**
- **CFG -> PDA (SLR parser)**
- **CFG -> CNF**
- **CFG -> LL parse table and parser**
- **CFG -> SLR parse table and parser**
- **CFG -> Brute force parser**



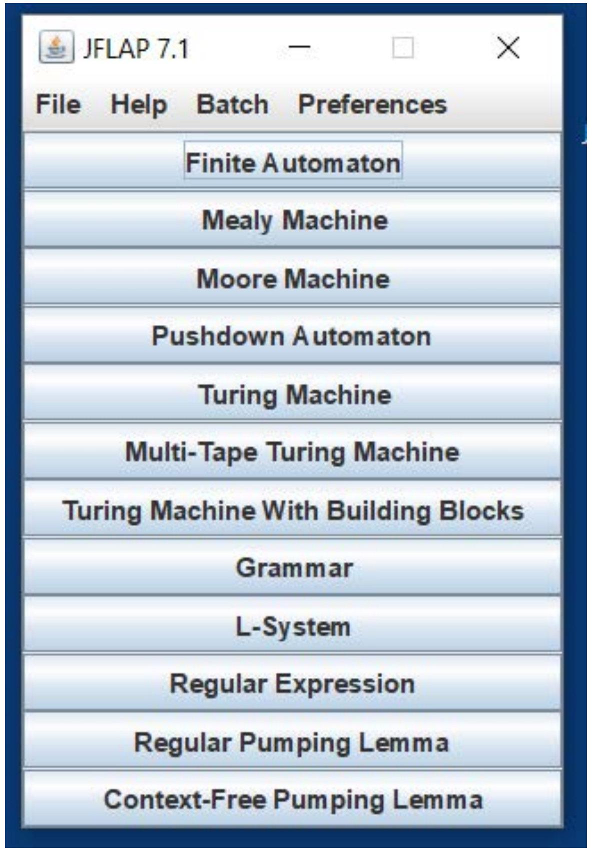
Turing Makineleri

Recursively Enumerable languages

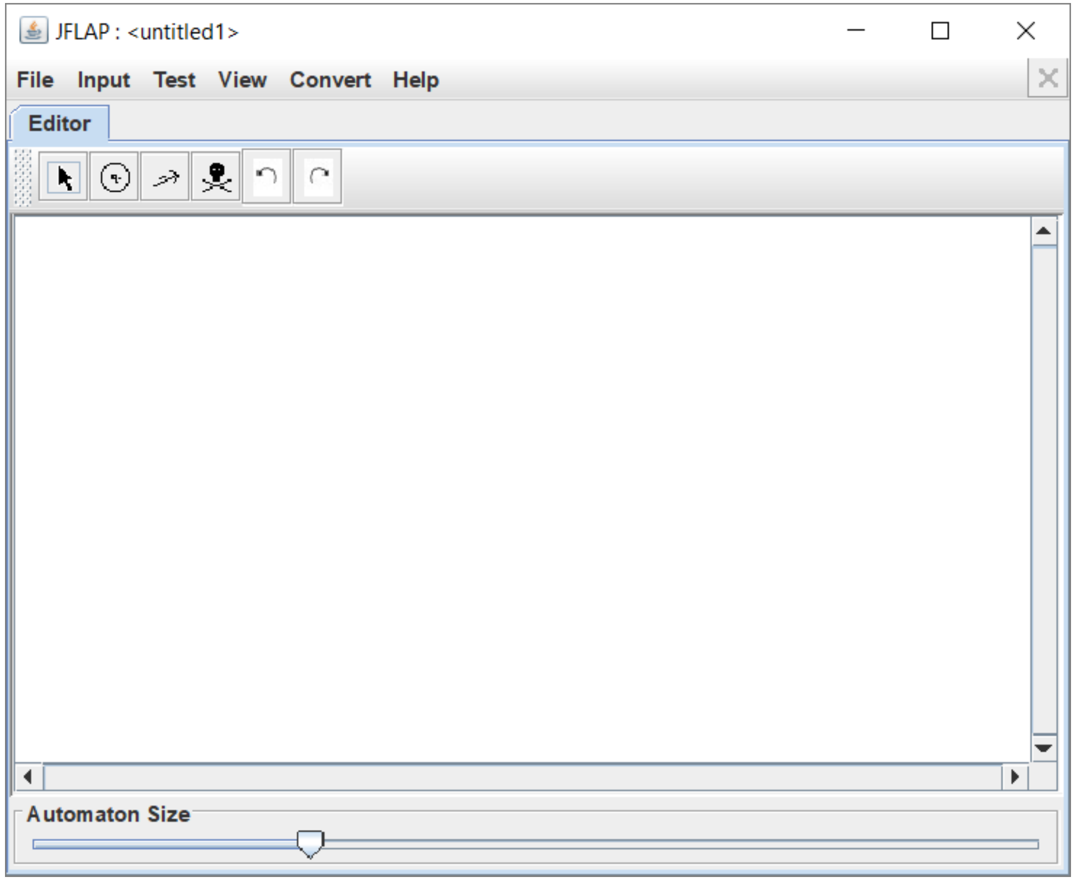
- **Turing machine (1-tape)**
- **Turing machine (multi-tape)**
- **Turing machine (building blocks)**
- **unrestricted grammar**
- **unrestricted gramamr -> brute force parser**



Arayüz


JFLAP 7.1
File
Help
Batch
Preferences
Finite Automaton
Mealy Machine
Moore Machine
Pushdown Automaton
Turing Machine
Multi-Tape Turing Machine
Turing Machine With Building Blocks
Grammar
L-System
Regular Expression
Regular Pumping Lemma
Context-Free Pumping Lemma

Arayüz


JFLAP : <untitled1>
File Input Test View Convert Help
Editor
Automaton Size


Deletor tool
Transition Creator tool
State Creator tool
Attribute Editor tool