

# BIMU4032

## DERLEYİCİ TASARIMI

# Bu Haftaki Konu Başlıkları

- **Sonlu Otomat (Finite Automata-FA)**
  - ▣ **Sonlu Otomat Gösterimleri**
  - ▣ **Non-Deterministik Finite Automata (NFA)**
  - ▣ **Bir Kurallı İfadenin NFA'ya Dönüştürülmesi**
  - ▣ **İyileştirmeler**
  - ▣ **DFA'nın Kurallı İfadeye Dönüştürülmesi**

# Sonlu Otomat (Finite Automata - FA)

- Sonlu otomat, durumları (states) içeren bir küme ve dışsal girdilere göre bu durumlar arasında gerçekleşen geçişlerden oluşmaktadır.
- Sonlu otomata modelleri, belli bir kelimenin karakterlerini sırayla girdi olarak alarak, belirli kelimeleri kabul veya reddetmekte, böylece belli bir kelimenin belli bir dile ait olup olmadığı ortaya çıkmaktadır.
- Bir sonlu otomatı sonuç durumlarından birisinde bırakan kelimeler, söz konusu sonlu otomat modelinin tanımladığı dilin elemanıdır.

# Sonlu Otomat (Finite Automata - FA)

Bir sonlu otomat üç elemandan oluşmaktadır:

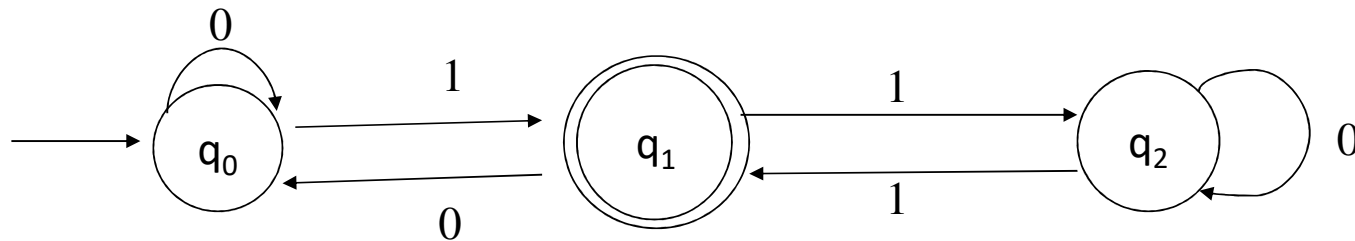
- Bir tanesi başlangıç, bir veya birden fazlası sonuç durumu olmak üzere, **Q sonlu durumlar kümesi**.
- Girdi sembollerine ilişkin  **$\Sigma$  alfabesi**.
- Belli bir durumdayken, girdi alfabesinin belirli bir sembolü için, o durumdan hangi duruma geçileceğini belirleyen **geçişler kümesi**.

# Sonlu Otomat Gösterimleri

- Sonlu otomatlar, 2 şekilde gösterilebilmektedir:
  - ▣ Geçiş Diyagramı (Transition Diagram)
  - ▣ Geçiş Tablosu (Transition Table)

# Sonlu Otomat Gösterimleri

□ **ÖRNEK:**  $M = ( \{q_0, q_1, q_2\}, \{0,1\}, \delta, q_0, \{q_1\} )$



	0	1
$q_0$	$q_0$	$q_1$
$*q_1$	$q_0$	$q_2$
$q_2$	$q_2$	$q_1$

Bu DFA tarafından kabul edilen bazı dizgiler:

01, 101, 0111, 11001

Kabul edilmeyen bazı dizgiler:

11, 00, 100, 1100

# Sonlu Otomat Gösterimleri

Belli bir girdiye ilişkin olarak, belli bir durumdan sadece tek bir çıkış varsa söz konusu sonlu otomat **deterministik(belirli)** olarak adlandırılır. DFA modeline göre, FA bir durumdan başlar ve uygulanan her giriş simgesi ile yeni bir duruma geçer. Her an FA'nın bulunduğu durum kesin olarak bellidir. Bu kurala uymayan otomatlar, deterministik değildir (**non-deterministic**).

# Non-Deterministik Finite Automata (NFA)

Kurallı ifadeleri verimli programlara dönüştürmek için bir basamak kullanılır. NFA non-deterministik olması nedeniyle gerçek makinelere yakın değildir. Bu nedenle NFA'nın Deterministik Finite Automata (DFA)'ya çevrilmesi gerekmektedir.

Sonlu bir otomat, sonlu bir durum sayısına sahip ve bunlar arasında sonlu geçiş sayısına sahip bir makinedir. Durumlar arasındaki bir geçiş, giriş alfabesindeki bir karakter yoluyla etiketlenir. Fakat burada aynı zamanda  $\epsilon$  ile işaretlenebilir. Bu geçiş  $\epsilon$ -geçişi olarak adlandırılır.

Sonlu bir makine, bir giriş stringinin özel bir string grubunun üyesi olup olmadığını belirlemek için kullanılır. Bunu yapmak için otomatın durumlarından birisi başlangıç durumu olarak seçilir.



# Non-Deterministik Finite Automata (NFA)

- NFA'lar, belirli sonlu otomatların (DFA) tersine her durumdan gidişin karışık olduğu ve her durum için bir sonraki karakterde nereye gidileceğinin belirli olmadığı otomatlardır.
- Basitçe DFA kurallarına uymayan bütün otomatlar NFA olarak adlandırılabilir.

# Non-Deterministik Finite Automata (NFA)

Bir NFA bir dizi  $Q$  durumu içerir. Bu durumlardan birisi,  $q_0 \in Q$ , otomatin başlangıç durumu olarak adlandırılır. Ayrıca bir  $\delta$  geçiş grubuna sahip olunur. Her bir  $\delta$  geçişi, bir çift  $q_1$  ve  $q_2$  durumunu birbirine bağlar ve bir sembol ile etiketlenir. Bu sembol  $\Sigma$  alfabesindeki karakterlerden birisi ya da  $\epsilon$  sembolüdür.

# Non-Deterministik Finite Automata (NFA)

□ Bir NFA  $M = (Q, \Sigma, \delta, q_0, F)$  beşlisi ile tanımlanır.

$Q$ : Mevcut durumların sonlu kümesi

$\Sigma$  : Sembollerin sonlu kümesi

$q_0 \in Q$  : Başlangıç durumu

$F \subseteq Q$  : Kabul durumları kümesi

DFA'dan farkı geçiş fonksiyonudur:

NFA'da  $\delta: Q \times (\Sigma \cup \{\lambda\}) \rightarrow 2^Q$  (Q'nun altkümeleri)

DFA'da:  $\delta: Q \times \Sigma \rightarrow Q$

NFA'da geçiş fonksiyonu ikinci argüman olarak  $\lambda$ -geçişi alabilir. Yani NFA bir girdi sembolü almadan da bir durumdan diğerine geçebilir.

# Non-Deterministik Finite Automata (NFA)

Genelde sonlu otomatları göstermek için grafiksel bir yapı kullanılır. Durumlar bu durumu tanımlayan bir sayı ya da ismi içeren daireler ile gösterilir. Bu isim ya da sayı işlemsel önceliğe sahip değildir. Sadece amaçların tanımlanması için kullanılır. Kabul edilen durumlar (çıkışlar) tek daire yerine çift daire ile gösterilir.

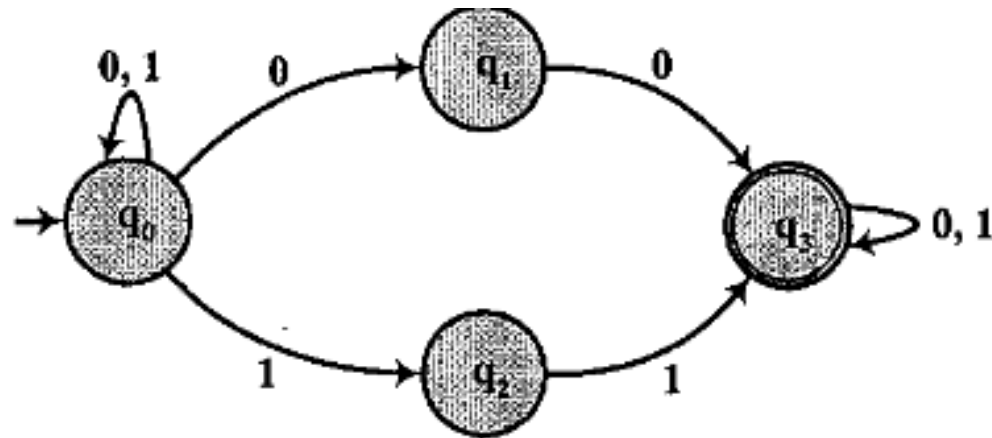
# Non-Deterministik Finite Automata (NFA)

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

$Q = \{q_0, q_1, q_2, q_3\}$

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

$F = \{q_3\}$



$\delta(q_0, 0) = \{q_0, q_1\}$

$\delta(q_0, 1) = \{q_0, q_2\}$

$\delta(q_1, 0) = \{q_3\}$

$\delta(q_1, 1) = \{\} = \emptyset$

$\delta(q_2, 0) = \emptyset$

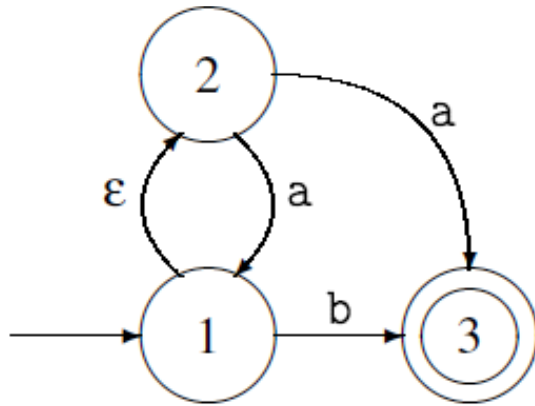
$\delta(q_2, 1) = \{q_3\}$

$\delta(q_3, 0) = \{q_3\}$

$\delta(q_3, 1) = \{q_3\}$

# Non-Deterministik Finite Automata (NFA)

Bir geiş iki durumu birbirine baėlayan bir ok ile gsterilir. Bu ok geişı tetikleyen sembol ile etiketlenir. Bařlangı durumunu iřaretleyen ok bir geiř deėildir ve bu nedenle bir sembol ile iřaretlenmez.



Yandaki řekil, 3 durumu bulunan bir NFA'yı gstermektedir. Durum 1 bařlangı durumu, durum 3 de kabul edilen bir ıkıřtır. Durum 1 ve 2 arasında bir  $\epsilon$  geiř vardır.

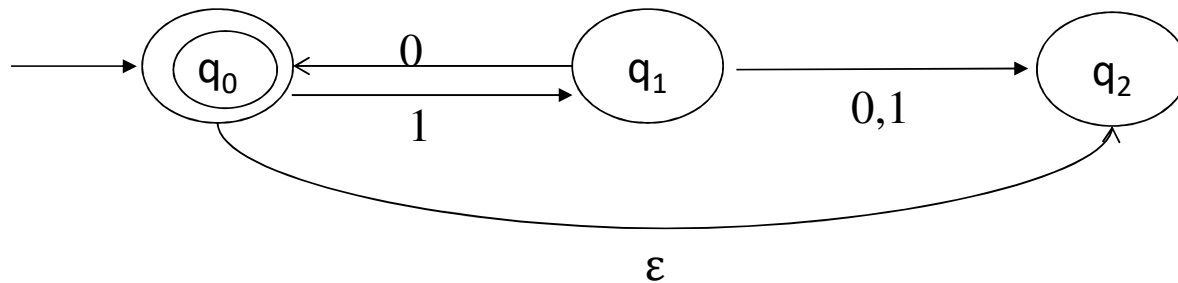
# Non-Deterministik Finite Automata (NFA)

Bu NFA, kurallı ifade  $a^*(a|b)$  ile tanımlanan dili tanır. Örnek olarak aab aşağıdaki geçişler ile tanınabilir.

from	to	by
1	2	$\epsilon$
2	1	a
1	2	$\epsilon$
2	1	a
1	3	b

Girişin sonunda durum 3'te olur. Bu kabul edilen bir durumdur. Bu nedenle bu string bu NFA ile kabul edilir.

# Non-Deterministik Finite Automata (NFA)



$\epsilon$ -geçişi var

$\delta(q_2, 0)$  yok  $\rightarrow \delta(q_2, 0) = \{ \}$

Bu otomatin kabul ettiği dizgiler:  $\lambda$ , 1010, 101010

Kabul edilmeyen dizgiler: 110, 10100

10 için iki alternatif var. Bir tanesi  $q_0$ 'a diğeri de  $q_2$  durumuna gider.  $q_2$  kabul durumu olmadığı halde bu dizgi kabul edilmektedir.

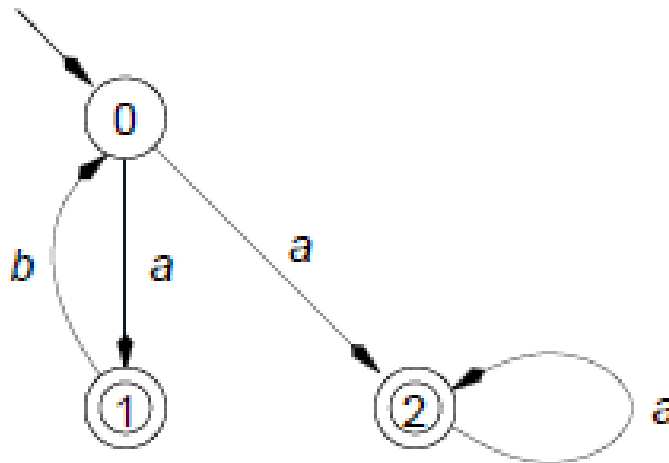


# Non-Deterministik Finite Automata (NFA)

NFA'da  $\delta(q_1, a) = \{q_0, q_2\}$

NFA'da  $\delta(q_i, a)$  kümesi boş olabilir. Yani belirli bir durum için geçiş tanımlanmayabilir.

Aşağıdaki NFA örneğinde 0 durumu, a sembolünü alarak 2 farklı duruma gitmiştir.

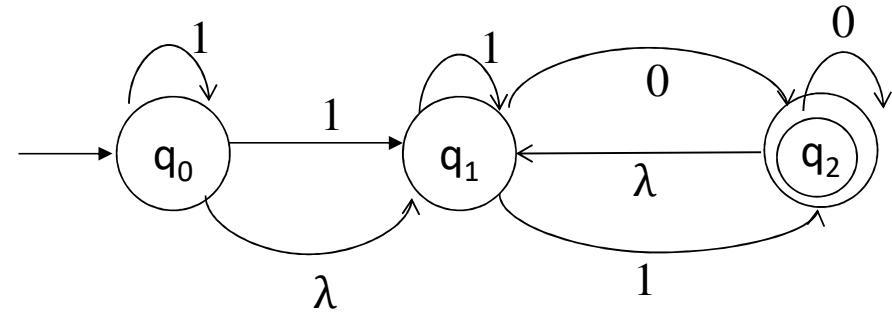


# Non-Deterministik Finite Automata (NFA)

Verilen bir NFA ile kabul edilecek bir stringe karar verecek bir program, bu kabullerden herhangi birinin string olup olmadığını görmek için tüm olası yolları kontrol etmelidir. Bu, başarılı bir yol bulunana kadar tekrar geriye dönerek ya da tüm olası durumlar için aynı anda çalışarak gerçekleştirilir. Bu yöntemlerin her ikisi de NFA'yı etkili tanıyıcı yapmak için oldukça zaman tüketici yöntemlerdir. Bu nedenle NFA'lar sadece kurallı ifadeler ve DFA'lar arasında bir basamak olarak kullanılır. Bu basamağın kullanılma nedeni bir kurallı ifadeden bir NFA'yı oluşturmanın bir DFA'yı oluşturmaya göre daha basit olmasıdır.

Örnek:  $\delta$  geçiş fonksiyonu şu şekilde verilmiş olsun:

	0	1	$\lambda$
$q_0$	$\emptyset$	$\{q_0, q_1\}$	$\{q_1\}$
$q_1$	$\{q_2\}$	$\{q_1, q_2\}$	$\emptyset$
$*q_2$	$\{q_2\}$	$\emptyset$	$\{q_1\}$

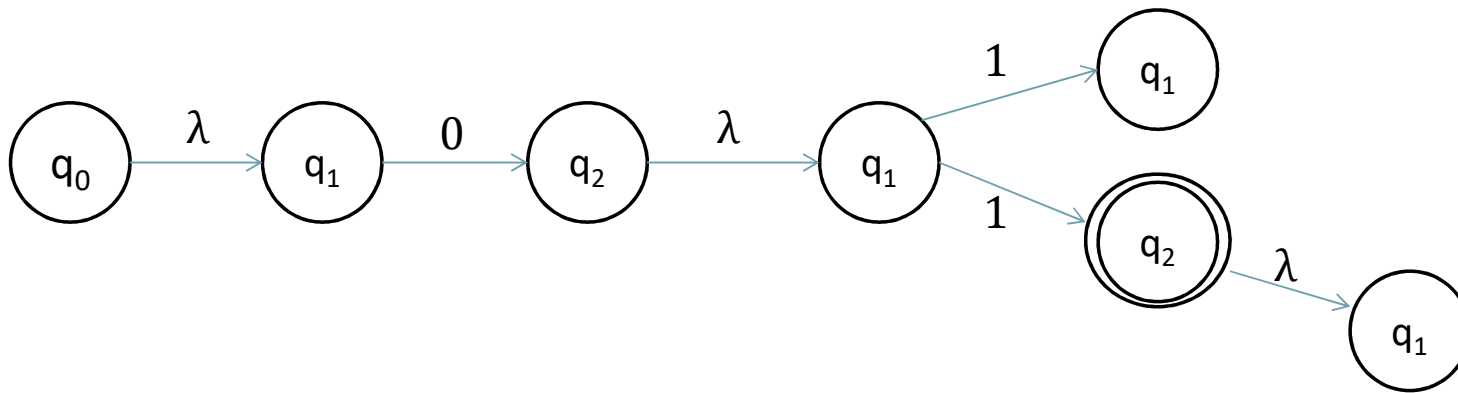


Örneğin; 01 girdisi için NFA'da 3 hesaplama yolu vardır:

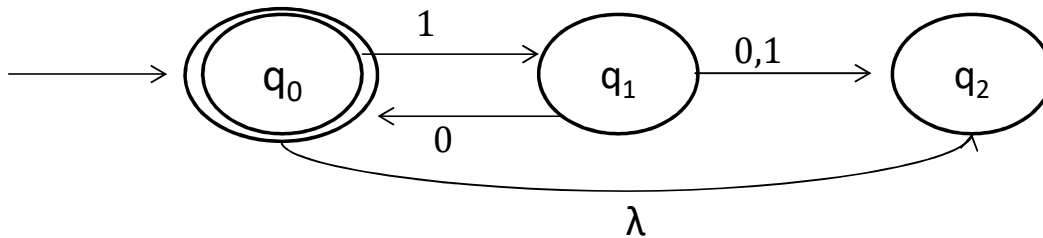
- $q_0 \xrightarrow{\lambda} q_1 \xrightarrow{0} q_2 \xrightarrow{\lambda} q_1 \xrightarrow{1} q_1$
- $q_0 \xrightarrow{\lambda} q_1 \xrightarrow{0} q_2 \xrightarrow{\lambda} q_1 \xrightarrow{1} q_2$
- $q_0 \xrightarrow{\lambda} q_1 \xrightarrow{0} q_2 \xrightarrow{\lambda} q_1 \xrightarrow{1} q_2 \xrightarrow{\lambda} q_1$

Burada sadece 2. yol kabul durumu ile biter.

Bu örnek için ağaç yapısı şu şekildedir:



En az bir yol kabul durumuna gittiği için 01 girdisi bu NFA tarafından kabul edilir.



**Bu otomat tarafından  
kabul edilen dil?**

Dil: Bir  $M = (Q, \Sigma, \delta, q_0, F)$  NFA'sı tarafından kabul edilen  $L$  dili şu şekilde tanımlanır:

$$L(M) = \{w \in \Sigma^* : \delta^*(q_0, w) \cap F \neq \emptyset\}$$

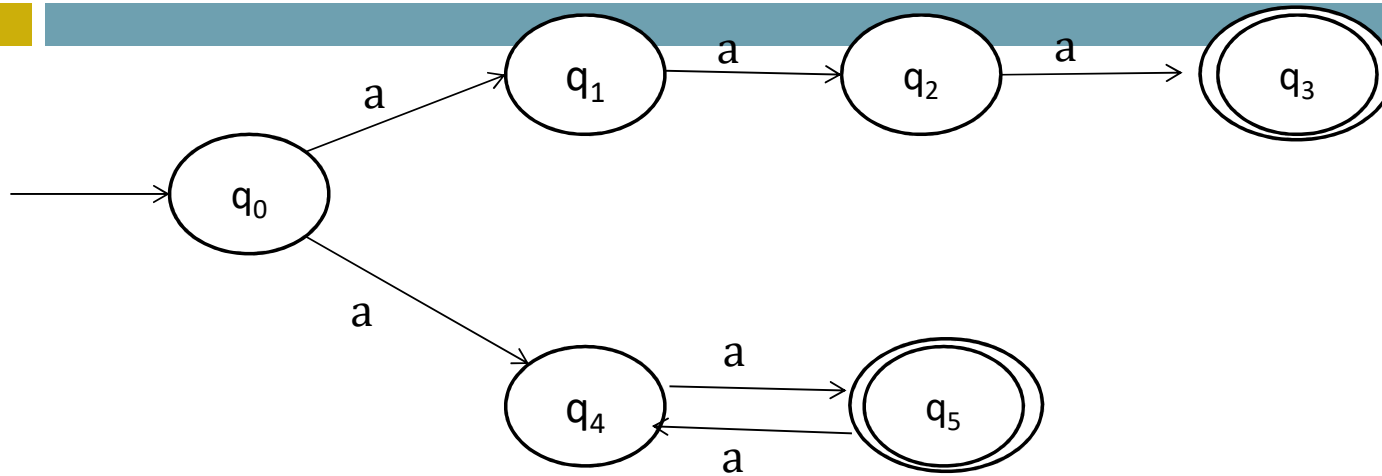
Grafikten görüldüğü gibi bu NFA'nın kabul durumunda durabilmesinin tek yolu 10 dizgisinin tekrarlı şekli ya da boş dizgidir.

$$L = \{(10)^n : n \geq 0\}$$

Örneğin 110 dizgisi için  $\delta(q_2, 0)$  tanımlanmadığından  $q_2$  durumunda kalır. Böyle bir duruma ölü konfigürasyon denir.

$\delta^*(q_0, 110) = \emptyset$  bu dizgi kabul edilmez.

Örnek:



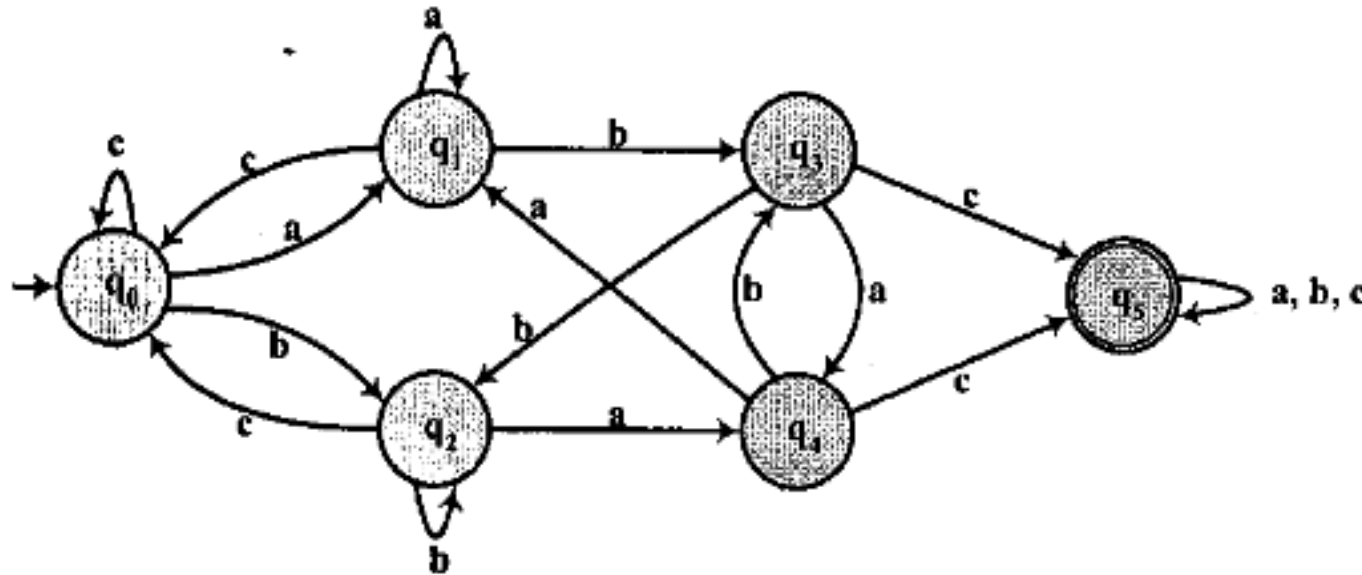
Bu NFA'nın kabul ettiği dil  $L = \{a^3\} \cup \{a^{2^n} : n \geq 1\}$

# Non-Deterministik Finite Automata (NFA)

{a, b, c } alfabelinde, içinde abc ya da bac altdizgisi bulunan dizgiler kümesi.

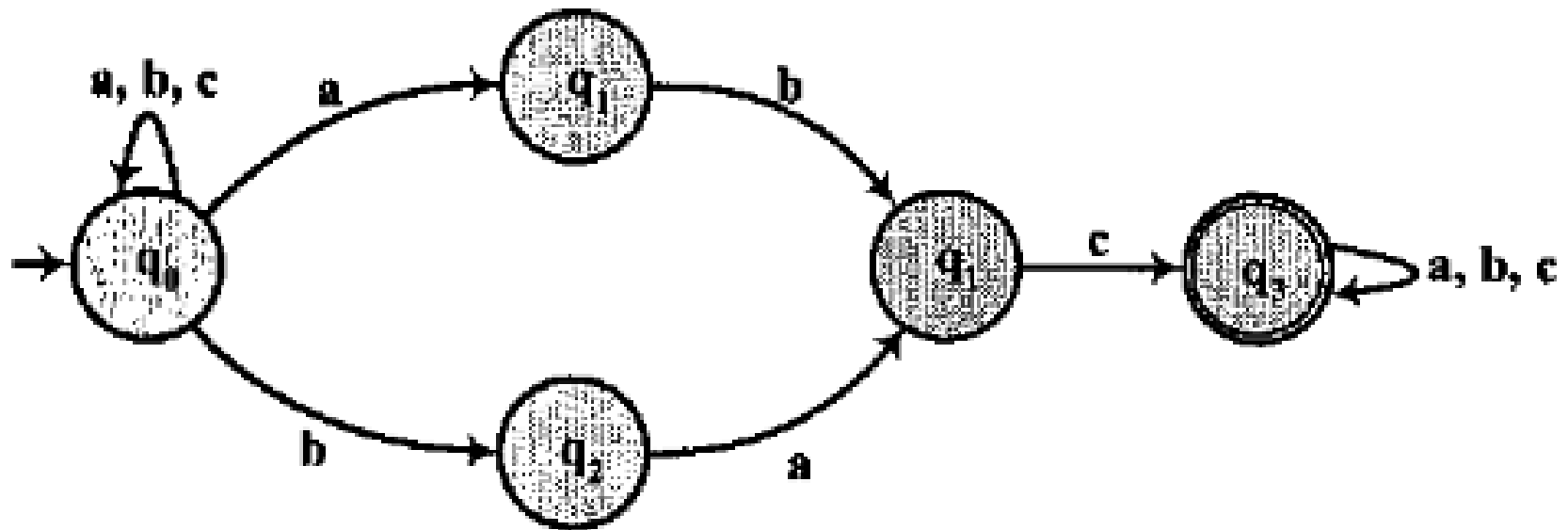
M'nin tanıdığı dizgilerden birkaç örnek alırsak;  $T(M) = \{abc, bac, cbabcb, bccaabcbac, \dots\}$

M'nin deterministik geçiş diyagramına bakıldığında sözlü tanımın oldukça kolay olmasına karşın, geçiş diyagramının oldukça karmaşık, olduğu görülmektedir.



# Non-Deterministik Finite Automata (NFA)

Deterministik modelin kullanım güclüğü nedeniyle, kullanımı daha kolay ve daha esnek bir model olan NFA model geliştirilmiştir.





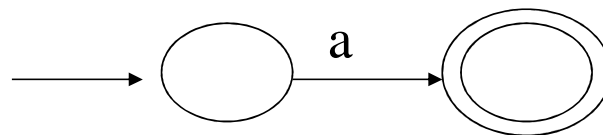
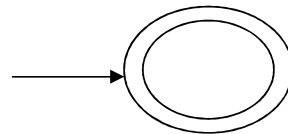
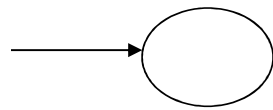
# Bir Kurallı İfadenin NFA'ya Dönüştürülmesi

Bir kurallı ifade için NFA, onun alt ifadelerinden oluşmuş NFA'lardan oluşturulur. Kesin olması için her bir alt ifadeden bir NFA parçası oluşturulur ve bu parçalar daha büyük parçalar içerisinde birleştirilirler. Bir parça tam bir NFA değildir. Bu nedenle tam bir NFA yapmak için gerekli bileşenler eklenerek yapı tamamlanacaktır.

Bir NFA parçası durumlar arasındaki geçişler ile durum sayısını içerir ve ek olarak iki tane tamamlanmamış geçişi içerir: İlki parça içine diğeri parçanın dışına doğrudur. Gelen yarım geçiş bir sembol ile işaretlenmez fakat dışarı giden yarım geçiş ya  $\epsilon$  sembolü ya da alfabe sembolü ile etiketlenir. Bu yarım geçişler giriş ve çıkış parçalarıdır ve diğer parçalara bağlamak için kullanılır.

# Bir Kurallı İfadenin NFA'ya Dönüştürülmesi

*Base step:* Give an NFA that accepts each of the simple or “base” languages,  $\emptyset$ ,  $\{\epsilon\}$ , and  $\{a\}$  for each  $a \in \Sigma$ .

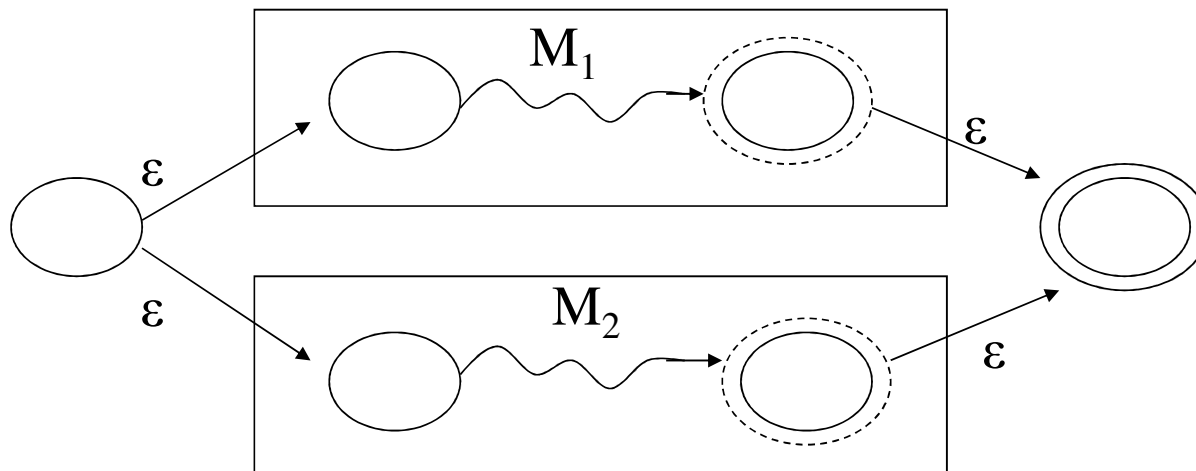


# Bir Kurallı İfadenin NFA'ya Dönüştürülmesi

Inductive step: For each of the operations -- union, concatenation and Kleene star -- show how to construct an accepting NFA.

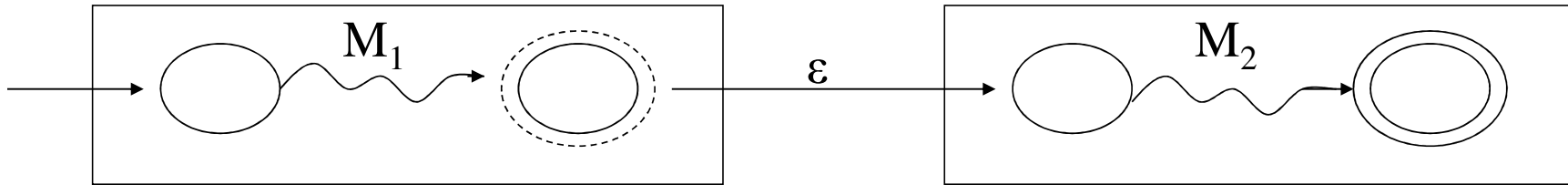
Closure under union:

$L(r_1+r_2)$  için otomat



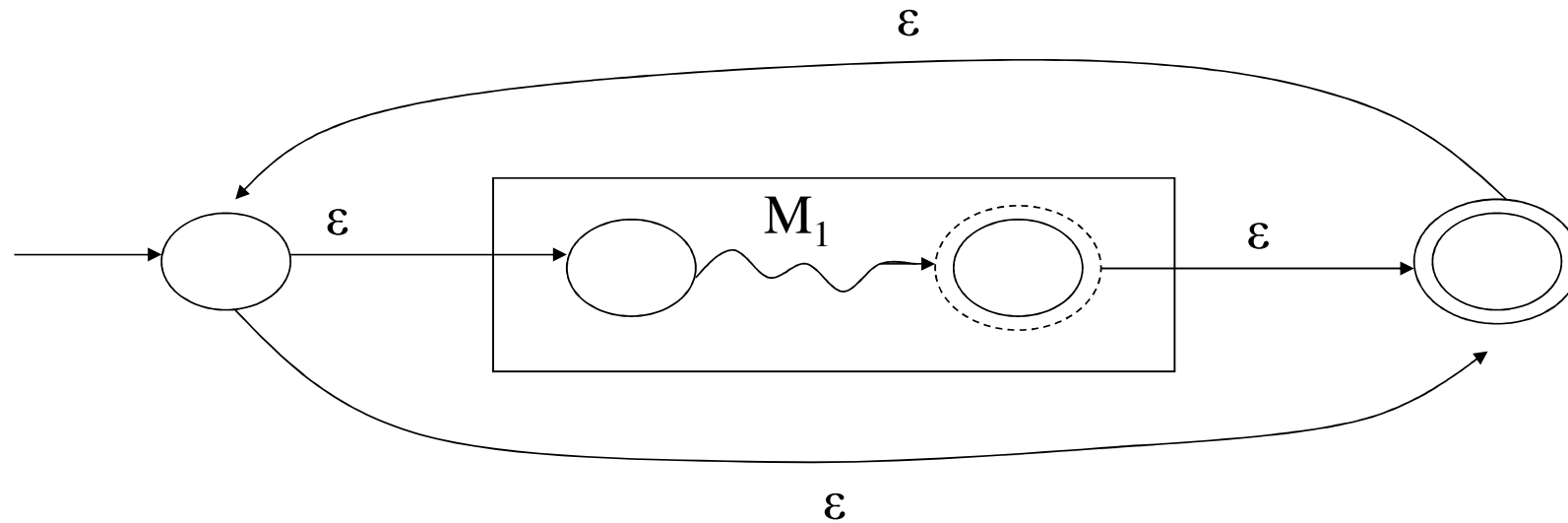
# Bir Kurallı İfadenin NFA'ya Dönüştürülmesi

Closure under concatenation:  $L(r_1 r_2)$



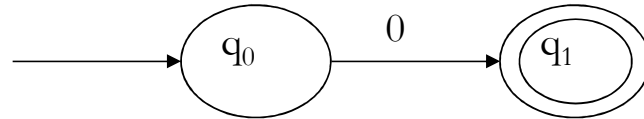
# Bir Kurallı İfadenin NFA'ya Dönüştürülmesi

Closure under Kleene Star:  $L(r_1^*)$

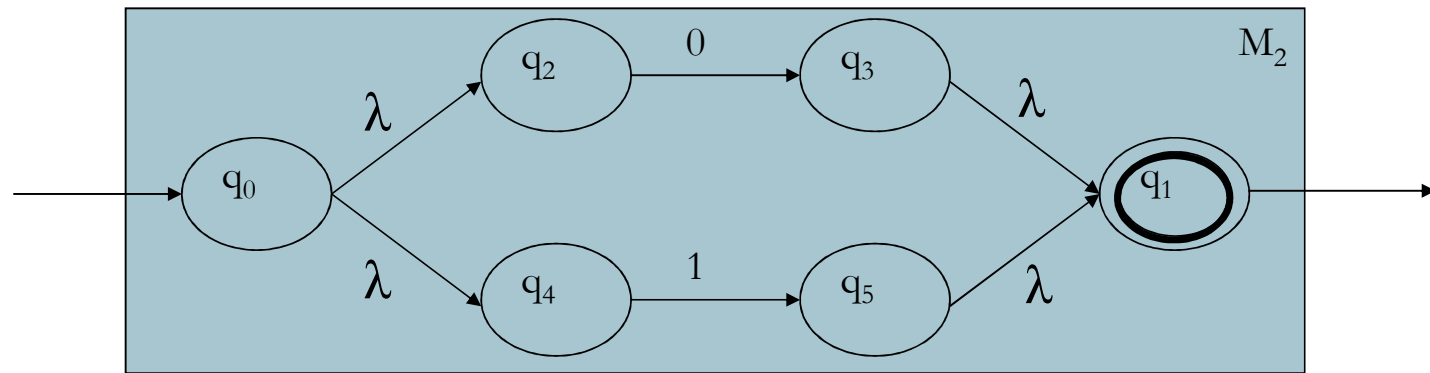


## Örnek: Düzgün ifade $\rightarrow$ $\lambda$ NFA

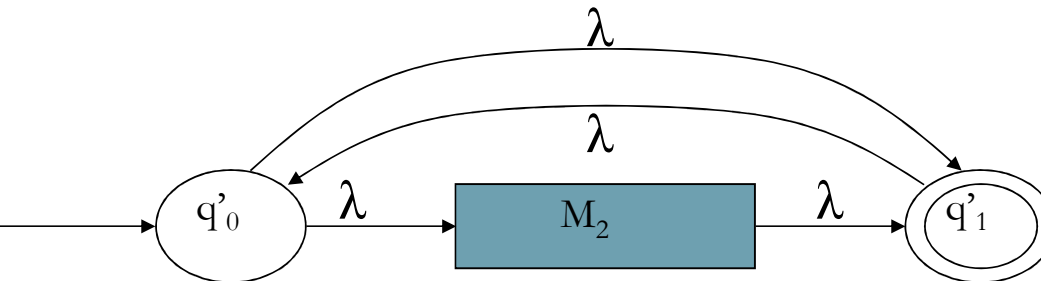
□  $R_1 = 0$



□  $R_2 = 0 + 1$



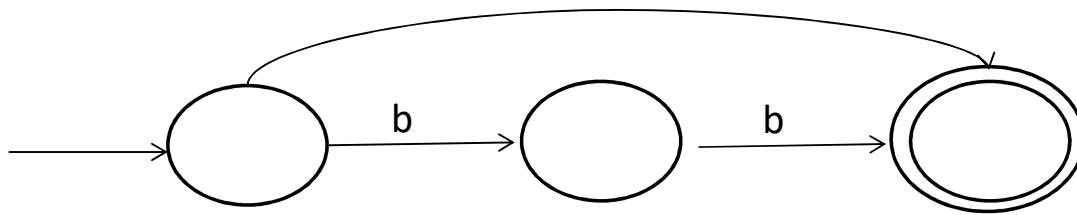
□  $R_3 = (0 + 1)^*$



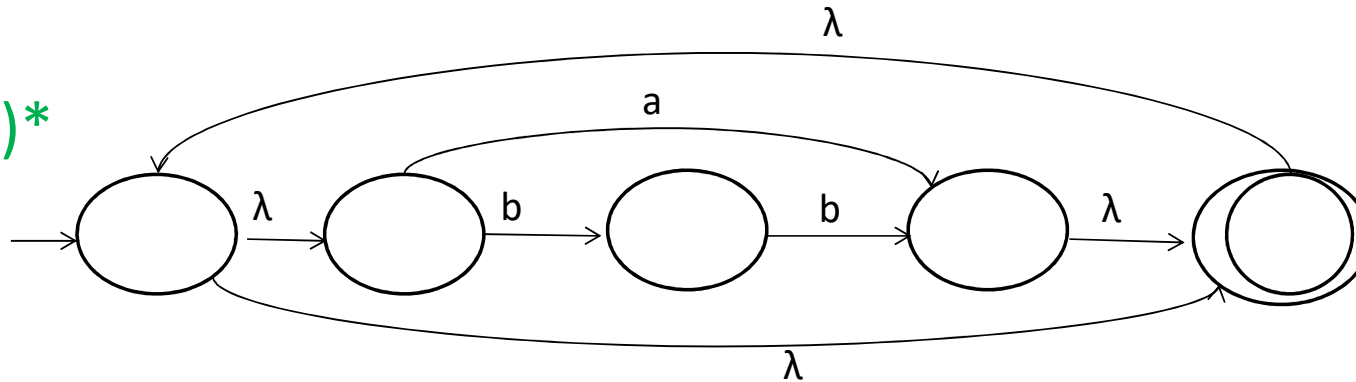
**Örnek:**  $r=(a+bb)^*(ba^*+\lambda)$  olmak üzere  $L(r)$  dilini kabul eden NFA bulun.

$L(a+bb)$  dilini kabul eden otomat  $M_1$  olsun.

$M_1$  :

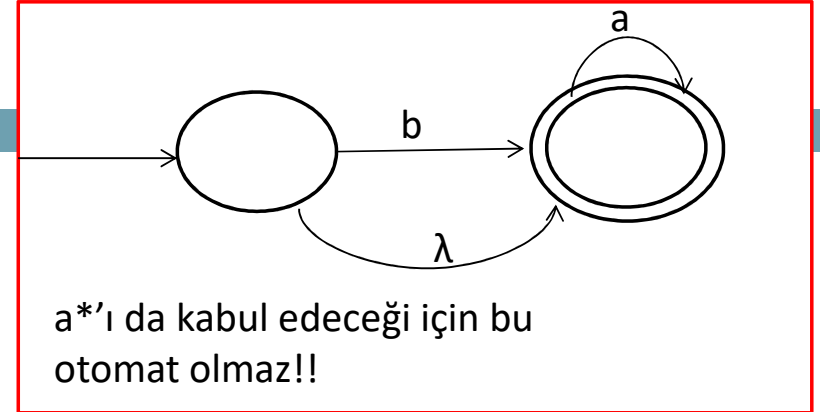
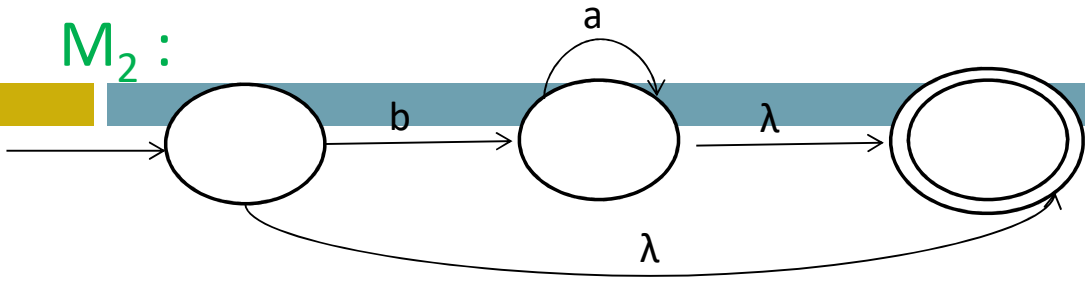


$L(a+bb)^*$

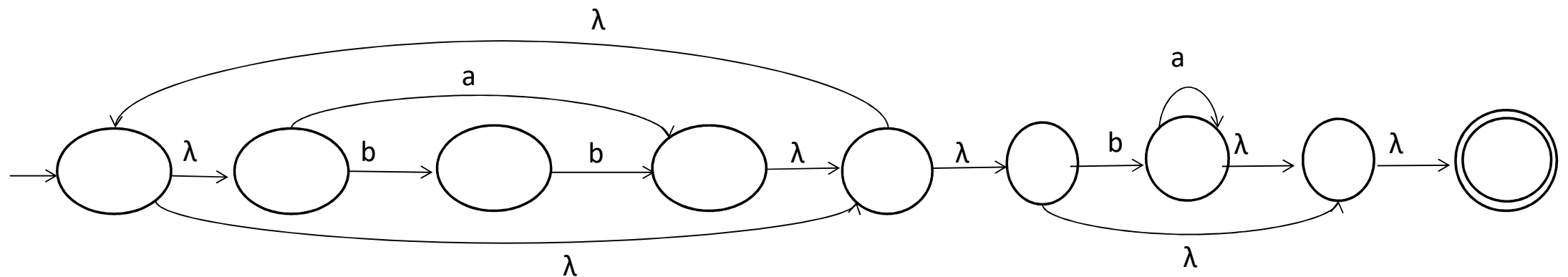


□  $L(ba^* + \lambda)$  dilini kabul eden otomat  $M_2$  olsun.

$M_2$  :



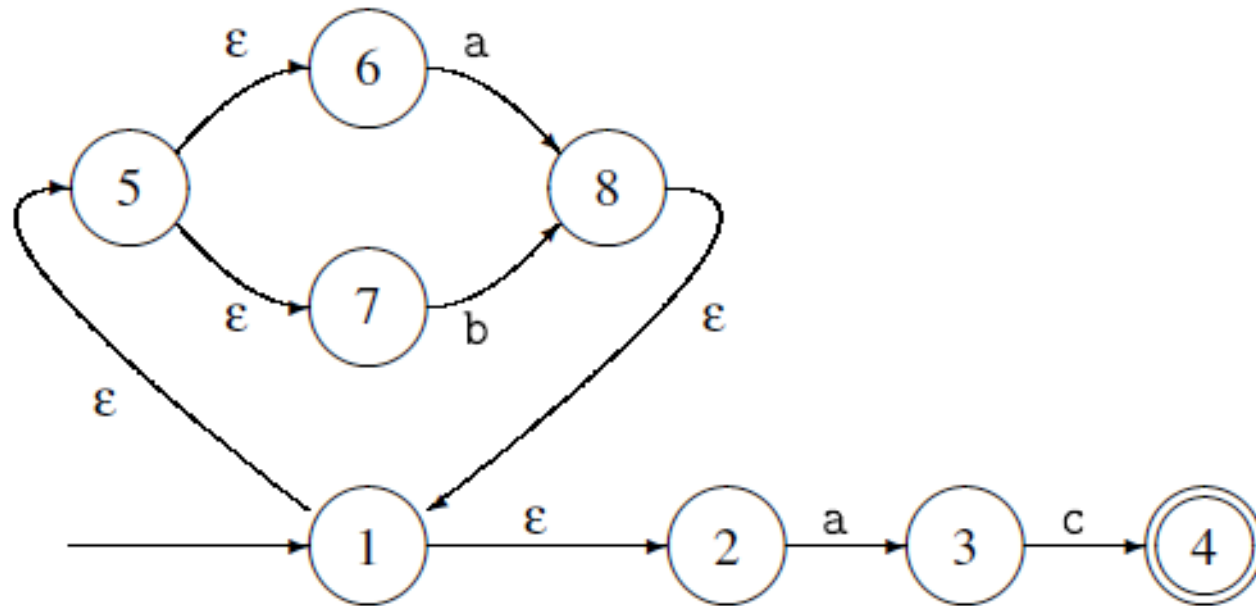
$M_1$  ve  $M_2$ 'yi birleştirerek otomatı çizelim:





# Bir Kurallı İfadenin NFA'ya Dönüştürülmesi

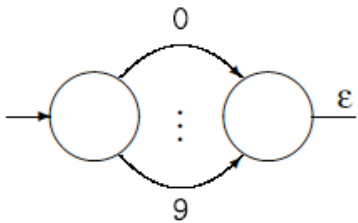
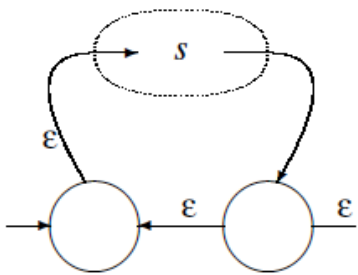
Bu şekilde oluşturulmuş kurallı ifade  $(a|b)^*ac$  için NFA aşağıdaki diyagramda gösterilmiştir.



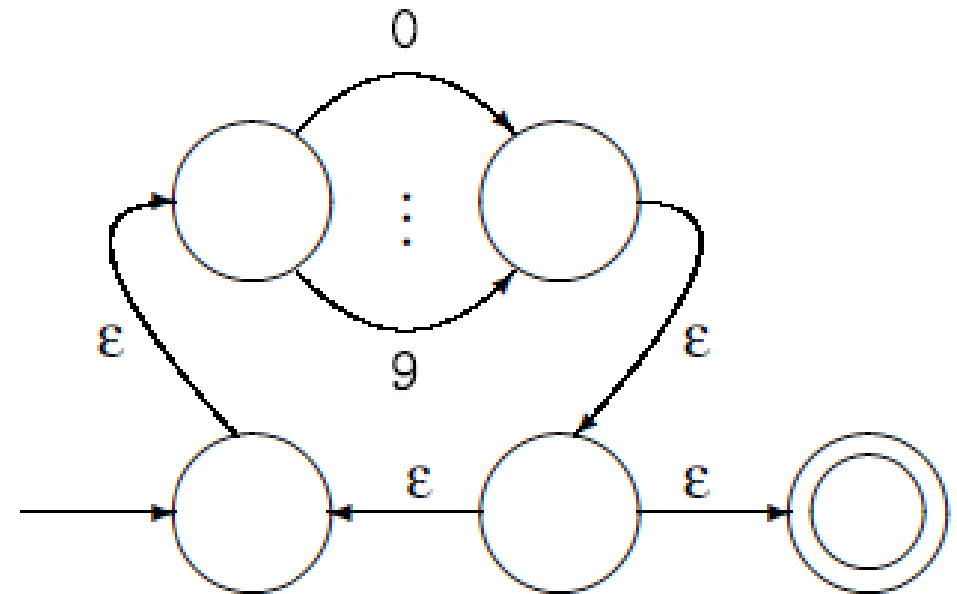
# İyileştirmeler

Bir önceki slaytta gördüğümüz yapılaşmayı tüm kısa yollar ile herhangi bir kurallı ifade için kullanabiliriz. Örneğin  $S^+$  yı  $SS^*$  a çevirmek için  $[0-9]^*$  u  $0 \mid 1 \mid 2 \mid \dots \mid 9^*$  a ve  $S^?$  'ni  $S \mid \epsilon$  ye çevirmek için gibi. Ancak, bu bazı ifadeler için oldukça büyük NFA'lar ile sonuçlanacaktır. Bu nedenle kısa gösterimler için bir kaç iyileştirilmiş yapılandırma kullanılır. Ayrıca, kurallı ifade  $\epsilon$  için alternatif bir yapılaşma gösterecektir. Bu yapılaşma bütününyle şekilde kullanılan formülü izlemez. Yerine, hat-segment gösterimi birleştirilmiş NFA parçalarının yarım-geçişlerin sadece bağlayacak  $\epsilon$  için NFA parçasını gösterir.  $[0-9]$  için oluşturmada, dikey elipsler  $[0-9]$  'daki sayıların her biri için geçişin bulunduğunu göstermektedir. Bu yapılaşma diğer karakter grupları için genelleştirilebilir. Örneğin  $[a-zA-Z0-9]$ .

# İyileştirmeler

Regular expression	NFA fragment
$\epsilon$	—
$[0-9]$	
$s^+$	

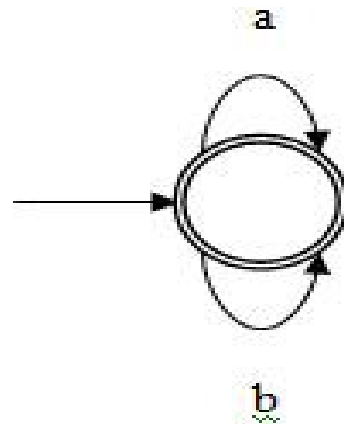
Örnek olarak  $[0-9]^+$  için iyileştirilmiş bir NFA, şekilde gösterilmiştir.



# Bazı kurallı ifadeler için elde edilmiş otomatlar

- $a^*b^*$  kurallı ifadesini sonlu otomat olarak göstermeye çalışalım:
- Bu ifadeden üretilebilecek örnekler:  
 $\epsilon, ab, aab, abb, aabb, aaab, abbb \dots$

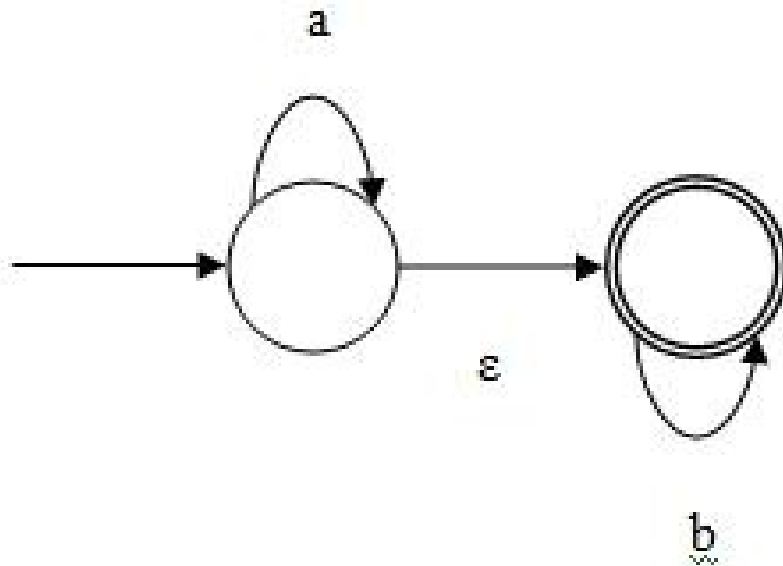
Yapılacak hatalı çizimlerden birisi aşağıdaki gibidir:



Bu otomat  $ba$  gibi bir kelimeyi de kabul eder ki bu yanlıştır.

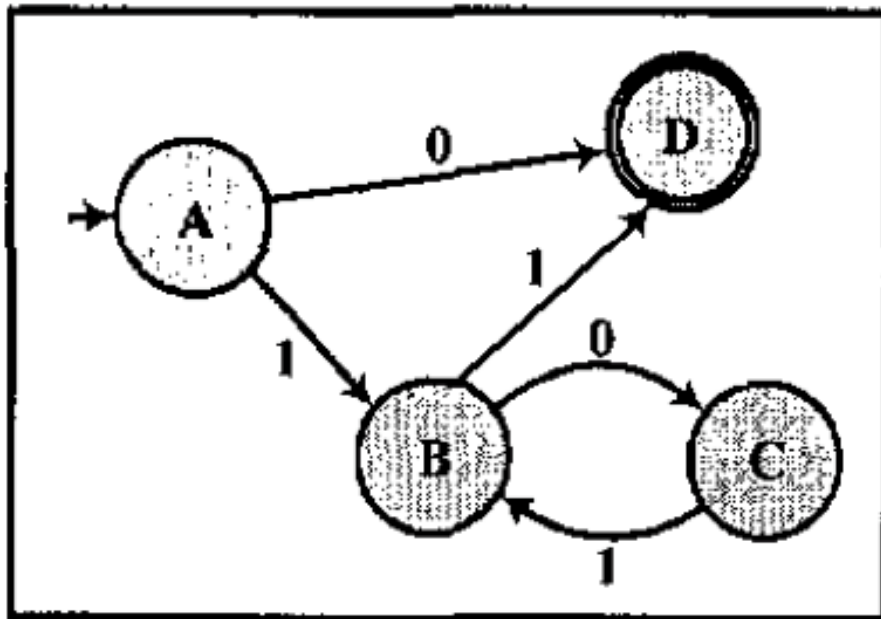
# Bazı kurallı ifadeler için elde edilmiş otomatlar

- Doğru çizim aşağıdaki şekildedir:

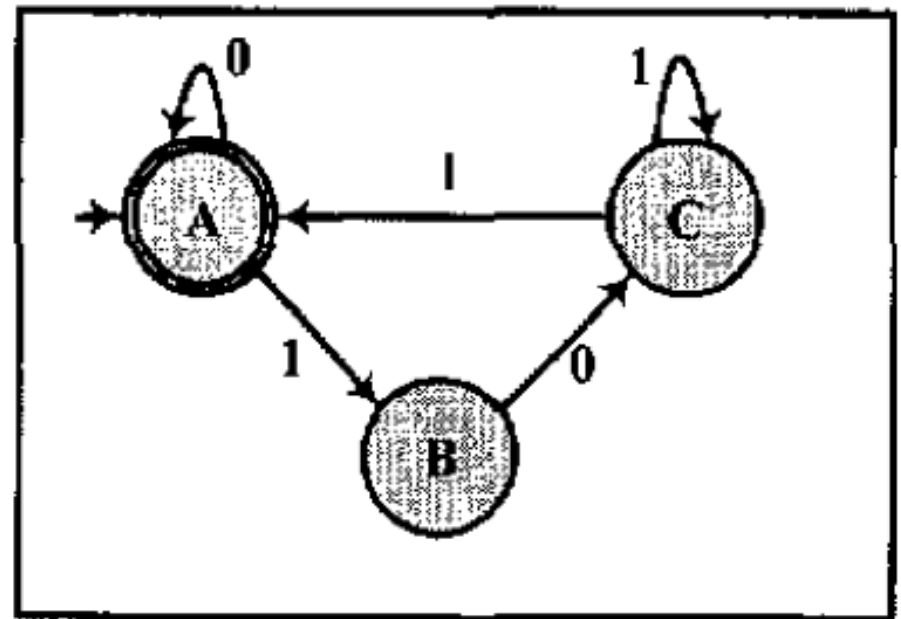


# Bazı kurallı ifadeler için elde edilmiş otomatlar

a)  $P_1 = 0 + 1(01)^* 1$

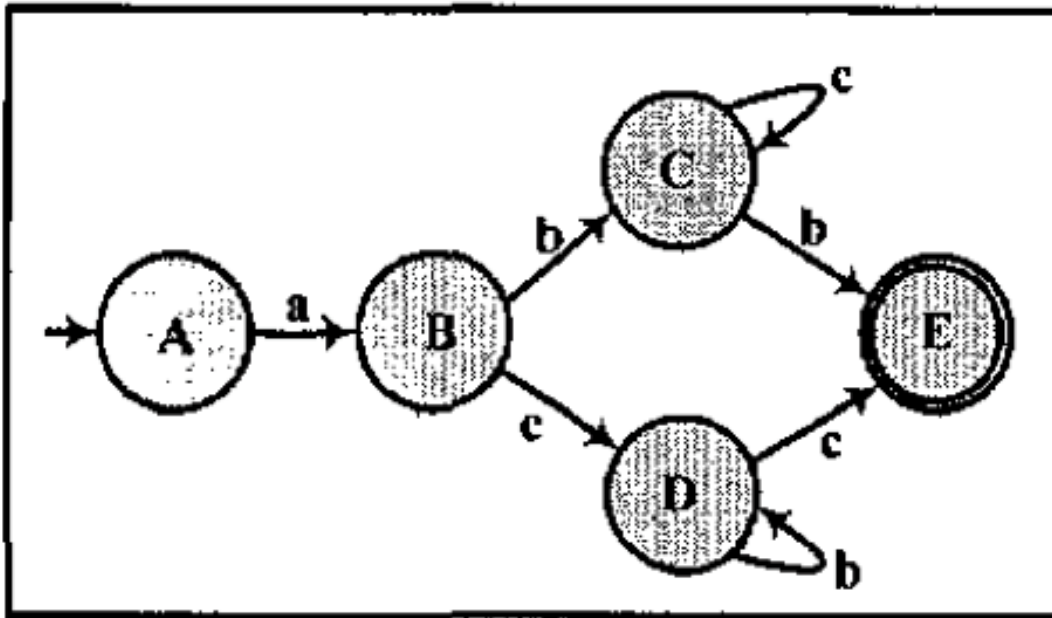


b)  $P_2 = (0 + 101^* 1)^*$

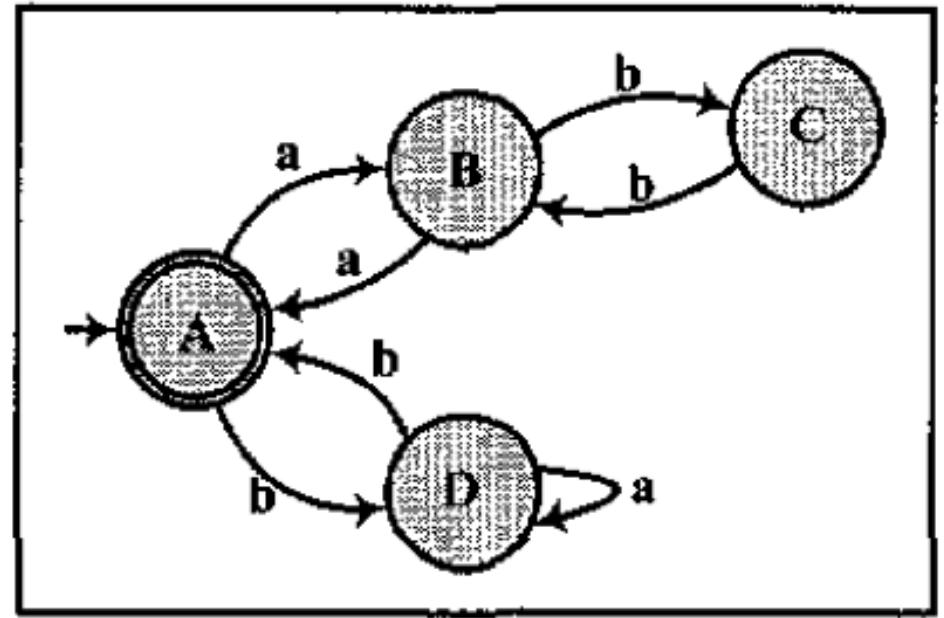


## Bazı kurallı ifadeler için elde edilmiş otomatlar

c)  $P_3 = a(bc^*b + cb^*c)$

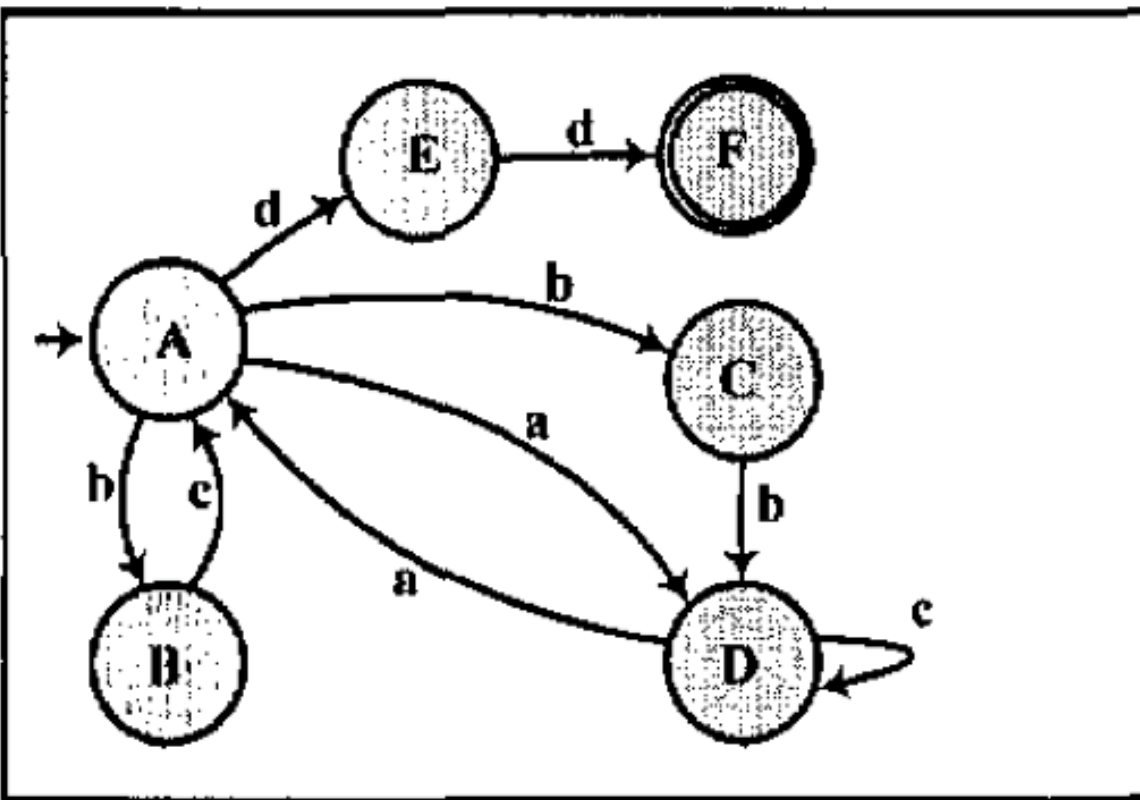


d)  $P_4 = (a(bb)^*a + ba^*b)^*$

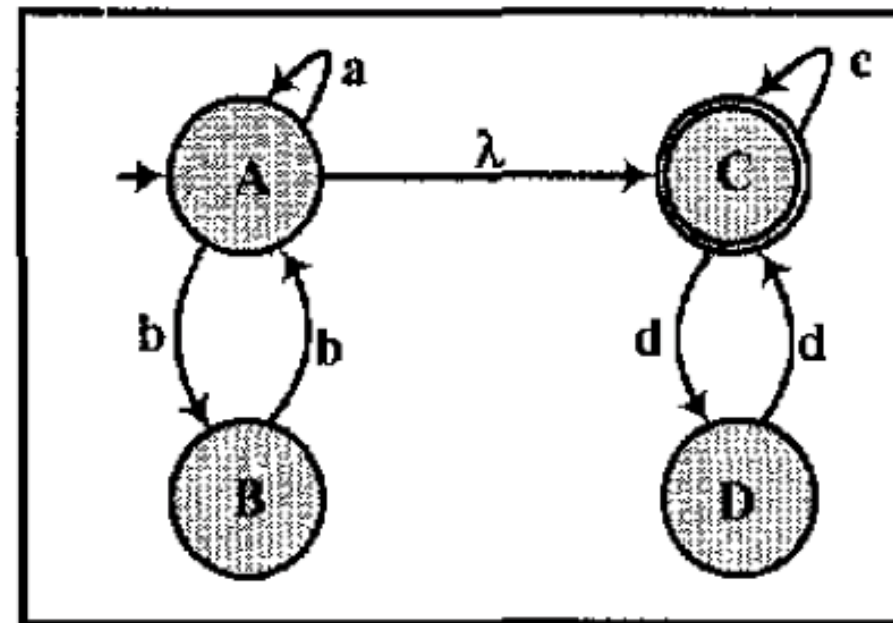


## Bazı kurallı ifadeler için elde edilmiş otomatlar

e)  $P_5 = (bc + (a + bb)c^*a)^* dd$

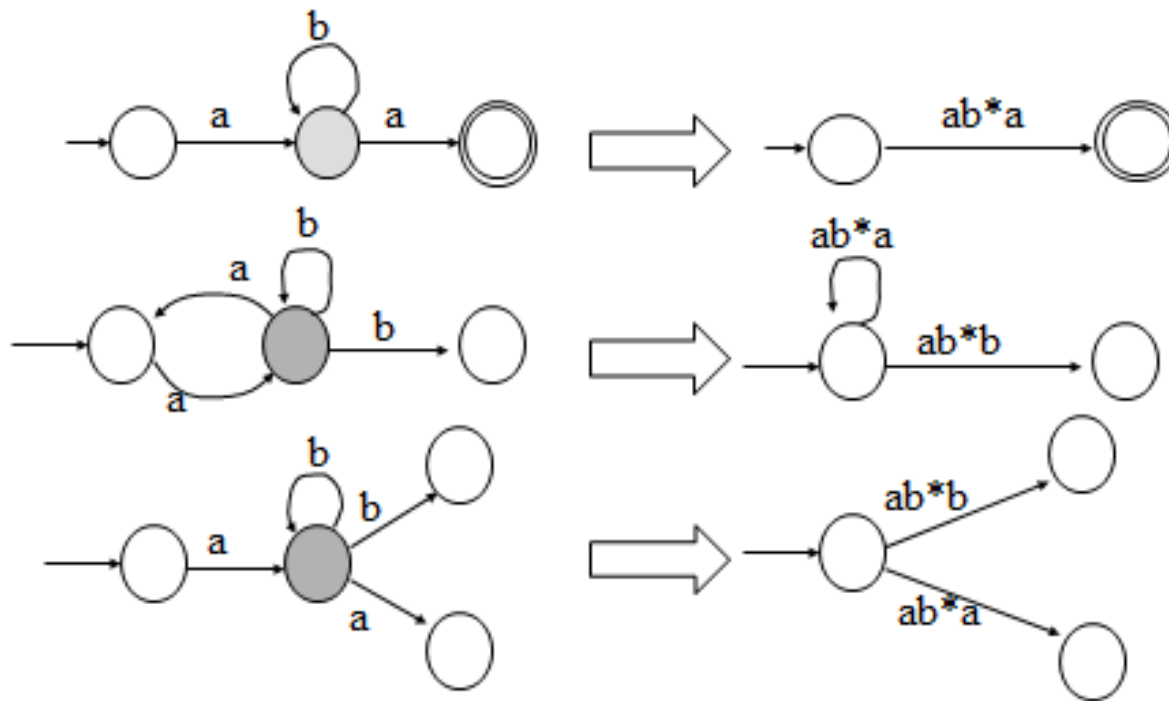


f)  $P_6 = (a + bb)^* (c + dd)^*$

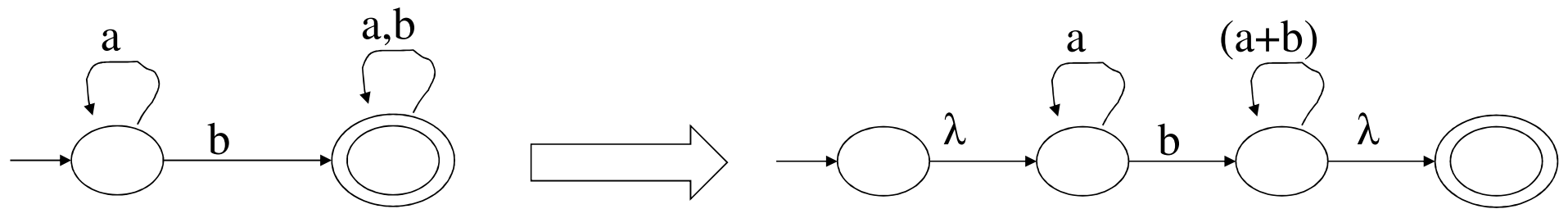




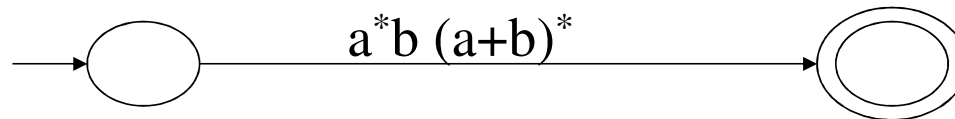
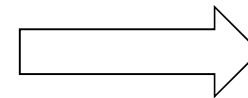
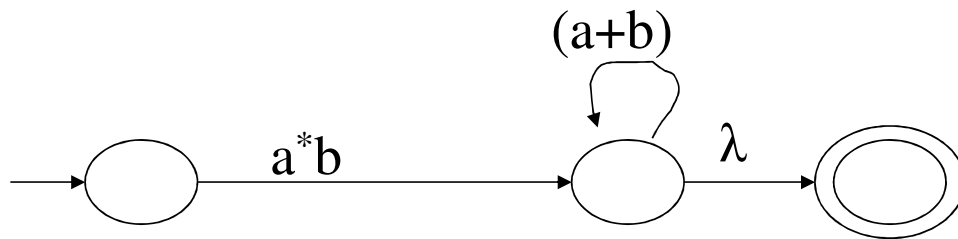
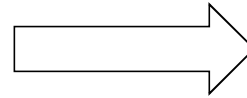
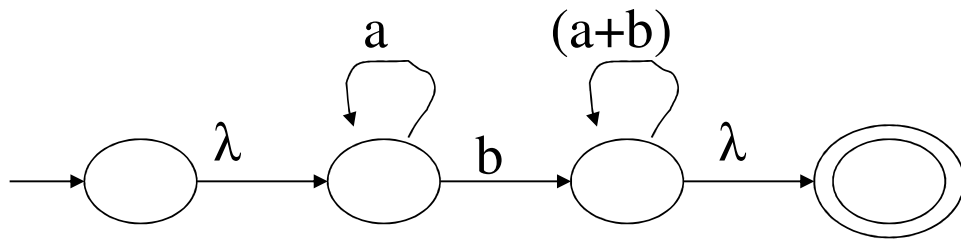
# DFA'nın Kurallı İfadeye Dönüştürülmesi



# DFA'nın Kurallı İfadeye Dönüştürülmesi

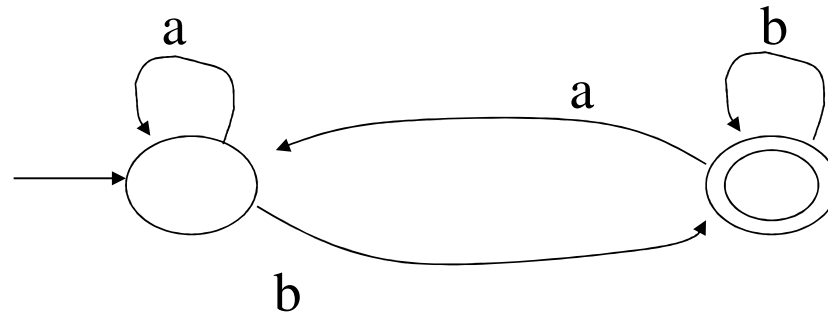


# DFA'nın Kurallı İfadeye Dönüştürülmesi



# DFA'nın Kurallı İfadeye Dönüştürülmesi

Find a regular expression that corresponds to the language accepted by the following DFA.



# DFA'nın Kurallı İfadeye Dönüştürülmesi

