BIMU4032
DERLEYİCİ TASARIMI

Deterministik Sonlu Otomat (DFA)

Sözlü olarak ya da bir kuralllı ifadeyle tanımlanan bir dizgiler kümesini tanıyan DFA'nın deterministik geçiş diyagramının oluşturulması oldukça zor olabilir. Bu nedenle NFA, kurallı ifadelerden DFA'lara geçişte bir basamak olarak kullanılır. Bir deterministik sonlu otomat üç bileşenden oluşmaktadır:

- Bir tanesi başlangıç, biri sonuç olabilecek durumlar kümesi.
- Girdi alfabesi.
- Bir durumdan diğer duruma geçişler (DFA'da Ɛ ile geçiş yapılmaz).

DFA ve NFA'nin farklari

- **DFA**

1. Tüm geçişler gerekircidir
   - Her geçiş sadece tek bir duruma gider
2. Her durum için mümkün olan tüm sembollerin (alfabe) geçişi tanımlanmalıdır
3. Son durum F'in elemanıysa input kabul edilir
4. Durumların sayısından dolayı bazen oluşturmak daha zordur
5. Pratik uygulaması yapılabilir

- **NFA**

1. Bazı geçişler non-deterministic olabilir
   - Bir geçiş, durumların altkümesine gidebilir
2. Tüm semboller için geçiş tanımlanması gerekmez
3. Son durumların bir tanesi F'in elemanıysa input kabul edilir
4. Oluşturmak DFA'ya göre daha kolaydır
5. Pratik uygulamasının deterministic olması gerekir (DFA'ya çevrilir)

Deterministik Sonlu Otomat (DFA)

$M_{1.1} = \langle Q, \Sigma, \delta, q_0, F \rangle$

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

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

$F = \{ q_2 \}$

δ : $\delta(q_0, 0) = q_0$

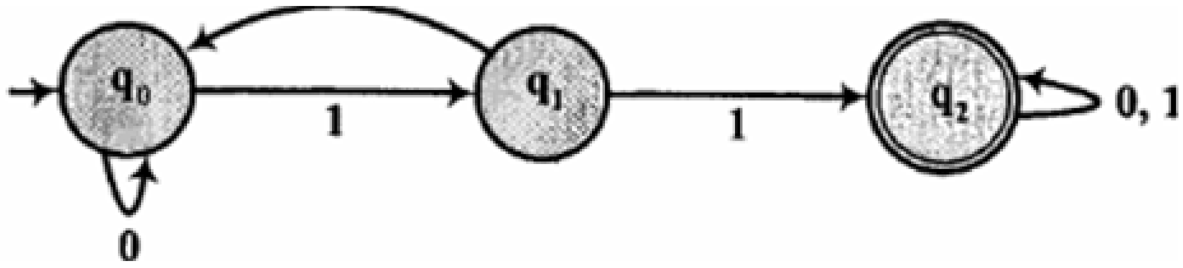
$\delta(q_0, 1) = q_1$

$\delta(q_1, 0) = q_0$

$\delta(q_1, 1) = q_2$

$\delta(q_2, 0) = q_2$

$\delta(q_2, 1) = q_2$



Deterministik Sonlu Otomat (DFA)

NFA modelinde;

- Bazı durumlardan bazı giriş simgeleri ile birden çok duruma geçilebilmektedir.
- Bazı durumlardan bazı giriş simgeleri ile de hiçbir duruma geçilememektedir.

Bu nedenle, deterministik modelde, başlangıç durumu ve uygulanan giriş dizgisi bilindiğinde makinenin hangi durumda bulunacağı kesin olarak bellidir. Buna karşılık, deterministik olmayan modelde başlangıç durumu ve uygulanan giriş dizgisi bilinen makinenin son durumu belirsiz olabilir.

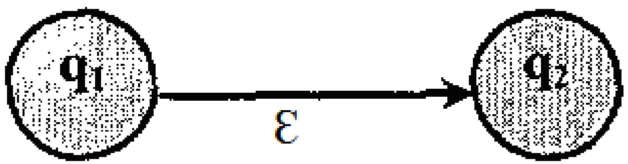
Deterministik Sonlu Otomat (DFA)

- Bir DFA'da aynı etikete sahip birden fazla çıkış durumu yoktur.
- Bulunulan durum ve bir sonraki giriş sembolü geçişe karar verir.
- DFA'da geçiş bağıntısı bir fonksiyondur ve şu şekilde ifade edilir: ***move* (*s, c*)**, s durumundan c geçiş sembolü ile ulaşılan yeni bir durumdur.
- DFA'lar NFA'lar ile aynı güce sahiptir. Bir DFA, NFA'nın özel bir durumudur ve herhangi bir NFA eşit bir DFA'ya dönüştürülebilir.

Ɛ Geçişi

FA'larda durum geçişleri, giriş simgelerinin işlenmesi ile gerçekleşir. Makine belli bir durumdayken, bir giriş simgesinin uygulanması makinenin bir sonraki duruma geçmesine neden olur. Ɛ geçişi ile FA'nın kullanımı kolaylaştırılır ve tanımı genişletilir. Ɛ geçişi soyut bir kavramdır ve boş simge olarak düşünülebilir.

Ɛ geçişi hiçbir giriş simgesi uygulanmadan (ya da işlenmeden) gerçekleşen durum geçişine karşı gelir. Bir makinenin q1 ve q2 durumları arasında Ɛ-geçişi varsa, q1 durumunda bulunan makinenin, hiçbir giriş simgesi işlenmeden, kendiliğinden q2 durumuna geçebileceği anlaşılır.



$$\delta(q_1, \varepsilon) = q_2$$

DFA ve NFA Denkliği

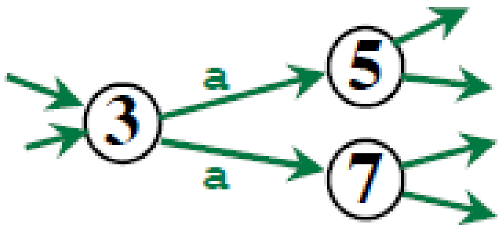
- Eğer her iki model ile tanınabilen kümeler sınıfı aynı ise bu iki model denktir.
- Eğer modellerden biri ile tanınabilen ancak diğeri ile tanınamayan kümeler varsa, bu iki model denk değildir.

Tanımlara göre her DFA aynı zamanda bir NFA olduğuna göre, deterministik bir FA tarafından tanınan, ancak hiçbir NFA tarafından tanınmayan bir küme düşünülemez.

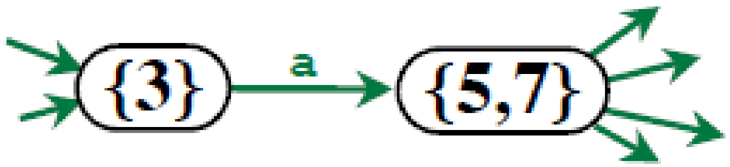
Bir NFA’nın DFA’ya Dönüştürülmesi

Dönüştürme işlemi öncelikle bir NFA’daki tüm olası geçişlerin gösterilmesi ile gerçekleştirilir. Bir durumda birden fazla geçiş seçeneği olduğu zaman, tüm seçenekler aynı anda alınır ve olası tüm sonraki durumların bir kümesi oluşturulur. Böyle bir NFA durumları kümesi tek bir DFA durumu oluşturacaktır. Verilen herhangi bir sembol için NFA’daki olası tüm sonraki durumların kümesi oluşturulacağı için bir NFA kümesinden diğer bir kümeye geçirecek tek bir geçiş (bu sembol ile etiketlenmiş) oluşturulmalıdır. Bu şekilde geçişler DFA’da deterministik olur.

Bir NFA'nın DFA'ya Dönüştürülmesi

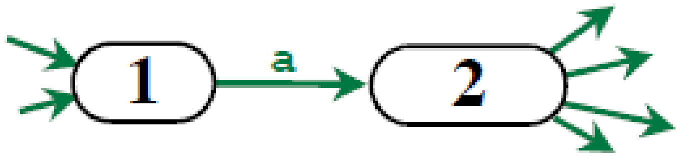


NFA

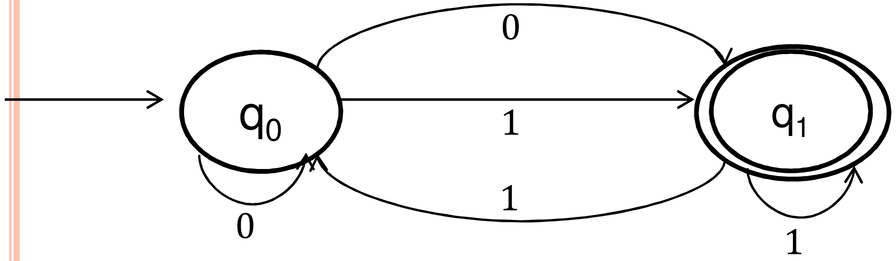


DFA

(The names of the states is arbitrary and can be changed later, if desired.)



Örnek : Aşağıdaki NFA’ya ait DFA’yı oluşturunuz.



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

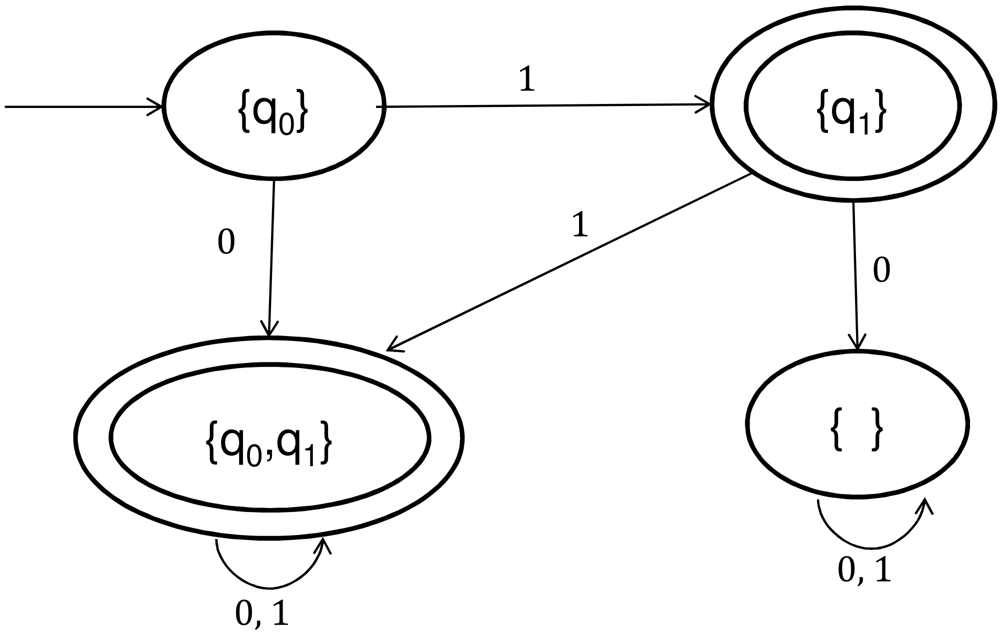
$\delta(q_0, 1) = \{q_1\}$

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

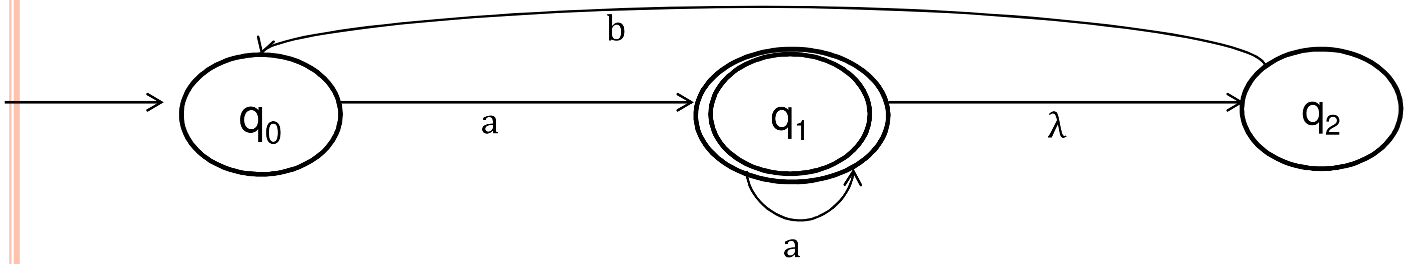
$\delta(\{q_0, q_1\},1)= \delta(q_0,1) \cup \delta(q_1,1)= \{q_0, q_1\}$

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

$\delta(q_1, 1) = \{q_0, q_1\}$



Örnek : Aşağıdaki NFA'yı eşdeğer DFA'ya dönüştürün.



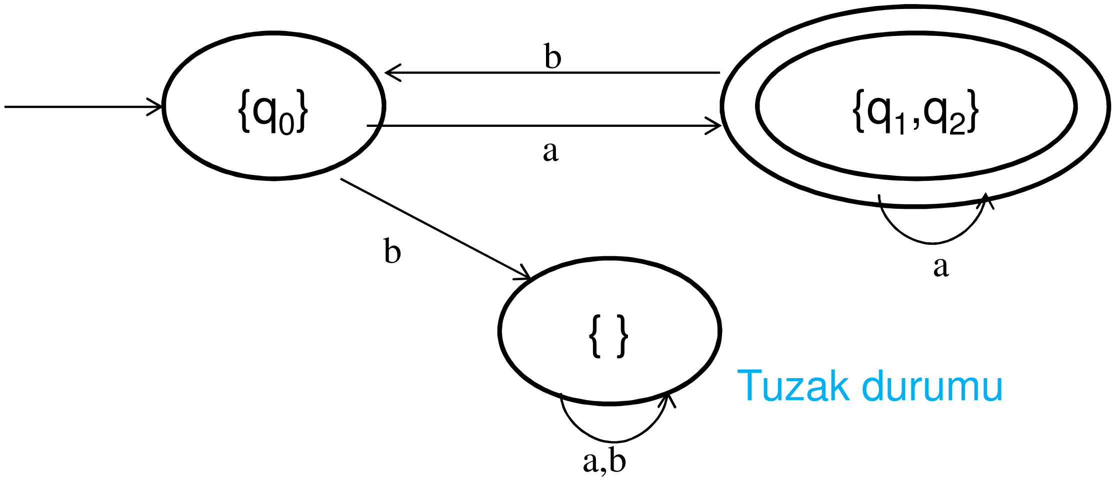
NFA q_0 durumu ile başladığı için DFA'nın başlangıcı $\{q_0\}$ olacaktır.

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

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

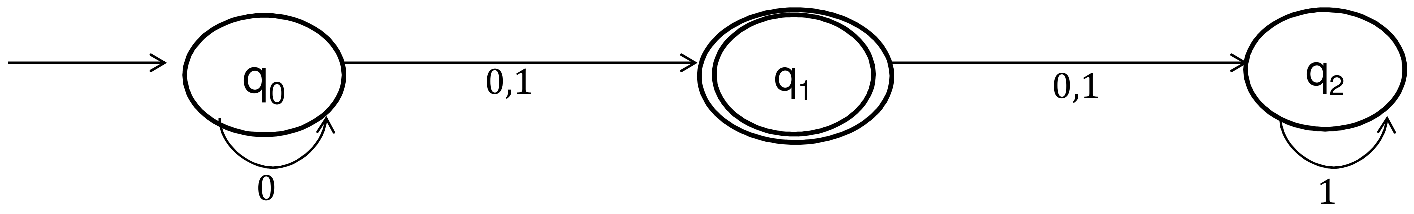
$\delta(\{q_1, q_2\}, a) = \{q_1, q_2\}$

$\delta(\{q_1, q_2\}, b) = \{q_0\}$



Tuzak durumu

Örnek : Aşağıdaki NFA'yı eşdeğer DFA'ya dönüştürün.



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

$\delta(q_0, 1) = \{q_1\}$

$\delta(q_1, 0) = \{q_2\}$

$\delta(q_1, 1) = \{q_2\}$

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

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

$\delta(\{q_0, q_1\},0) = \{q_0, q_1, q_2\}$

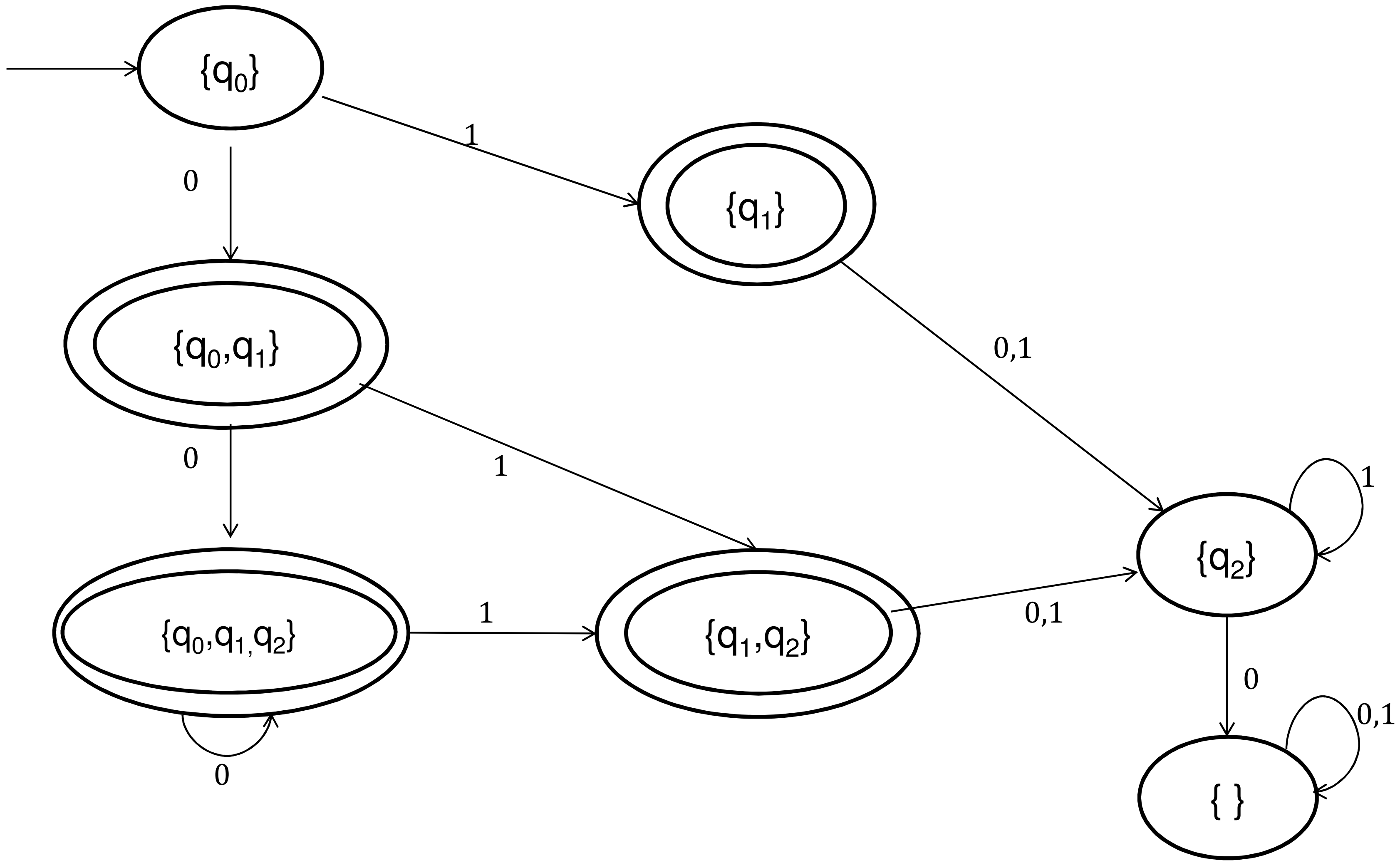
$\delta(\{q_0, q_1\},1) = \{q_1, q_2\}$

$\delta(\{q_0, q_1, q_2\},0) = \{q_0, q_1, q_2\}$

$\delta(\{q_0, q_1, q_2\},1) = \{q_1, q_2\}$

$\delta(\{q_1, q_2\},0) = \{q_2\}$

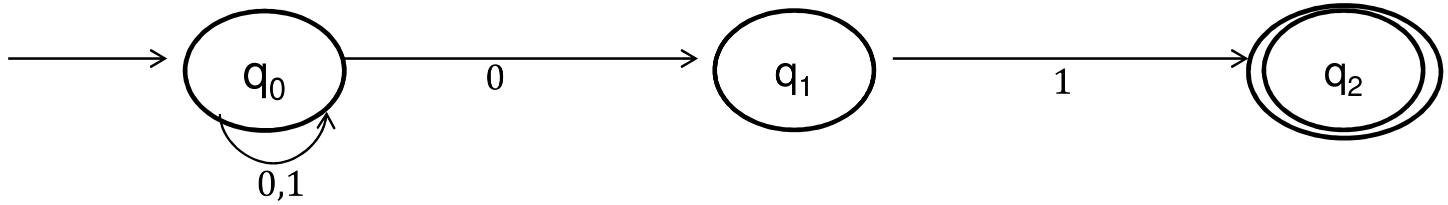
$\delta(\{q_1, q_2\},1) = \{q_2\}$



{q0}
1
0
{q1}
{q0,q1}
0,1
0
1
1
{q2}
{q0,q1,q2}
1
{q1,q2}
0,1
0
0,1
0
{ }

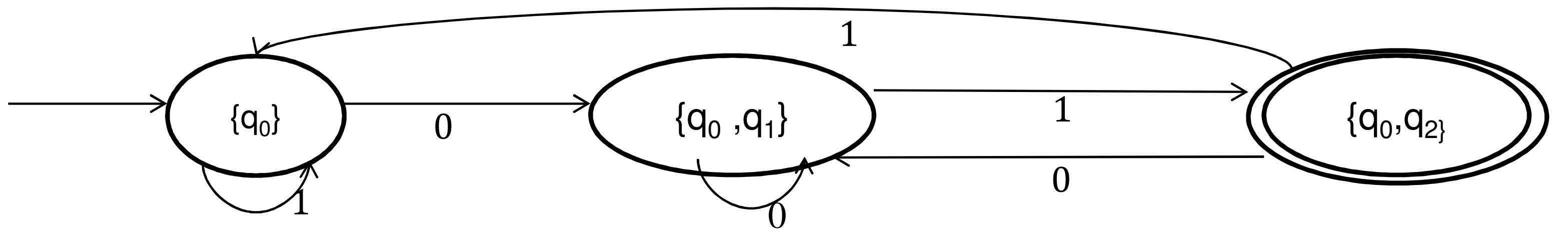
Verilen NFA’nın daha karmaşık olması durumunda NFA’nın tüm durumlarının alt kümelerini ele almaya gerek kalmayabilir. Böyle durumlarda, $\{q_0\}$ başlangıç durumundan başlayarak tek tek DFA’nın durumlarını oluşturabiliriz.

Örnek : Aşağıdaki NFA'yı eşdeğer DFA'ya dönüştürün.



δ	0	1
A Ø	Ø	Ø
B $\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
C $\{q_1\}$	Ø	$\{q_2\}$
*D $\{q_2\}$	Ø	Ø
E $\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
*F $\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$
*G $\{q_1, q_2\}$	Ø	$\{q_2\}$
*H $\{q_0, q_1, q_2\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$

δ	0	1
A	A	A
→B	E	B
C	A	D
*D	A	A
E	E	F
*F	E	B
*G	A	D
*H	E	F



λ-NFA’DAN NFA ELDE EDİLMESİ

Σ üzerinde tanımlanmış bir L dilini kabul eden herhangi bir M = (Q, Σ, δ, q_0, F) λ-NFA’DAN aynı L dilini kabul eden bir N = (Q’, Σ’, δ’, q_0’, F’) NFA elde edilebilir.

M’de sadece tek bir a sembolü ve sıfır ya da daha fazla λ alarak q_i’den q_j’ye giden bir yol varsa

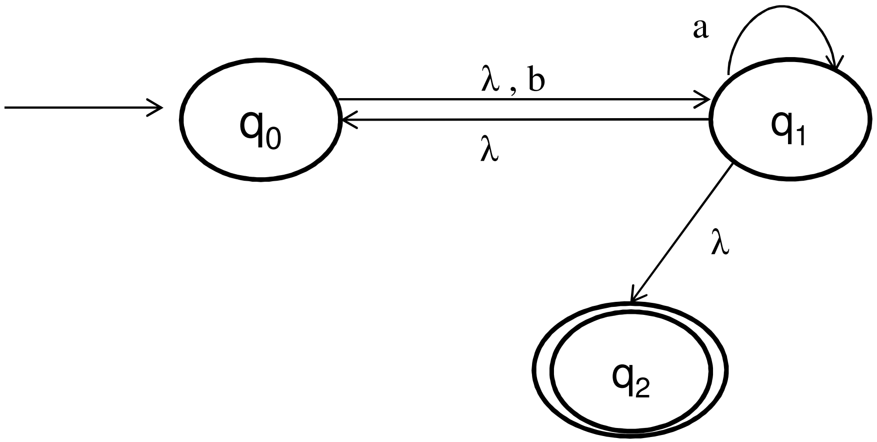
$\delta'(q_i, a) = q_j$ yazabiliriz.

F ’ = F U {q_0} → {M’de λ geçişleri kullanarak q_0’dan bir son duruma erişilebiliyorsa }

→ Aksi durumda F ’ = F olur.

Örnek:

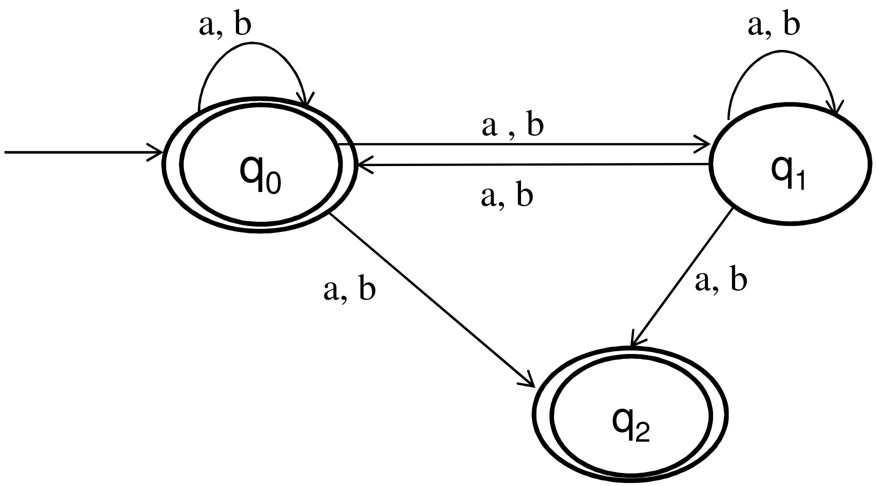
M



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

λ -NFA'dan elde edilen NFA ile λ -NFA aynı dizgi kümesini kabul eder.

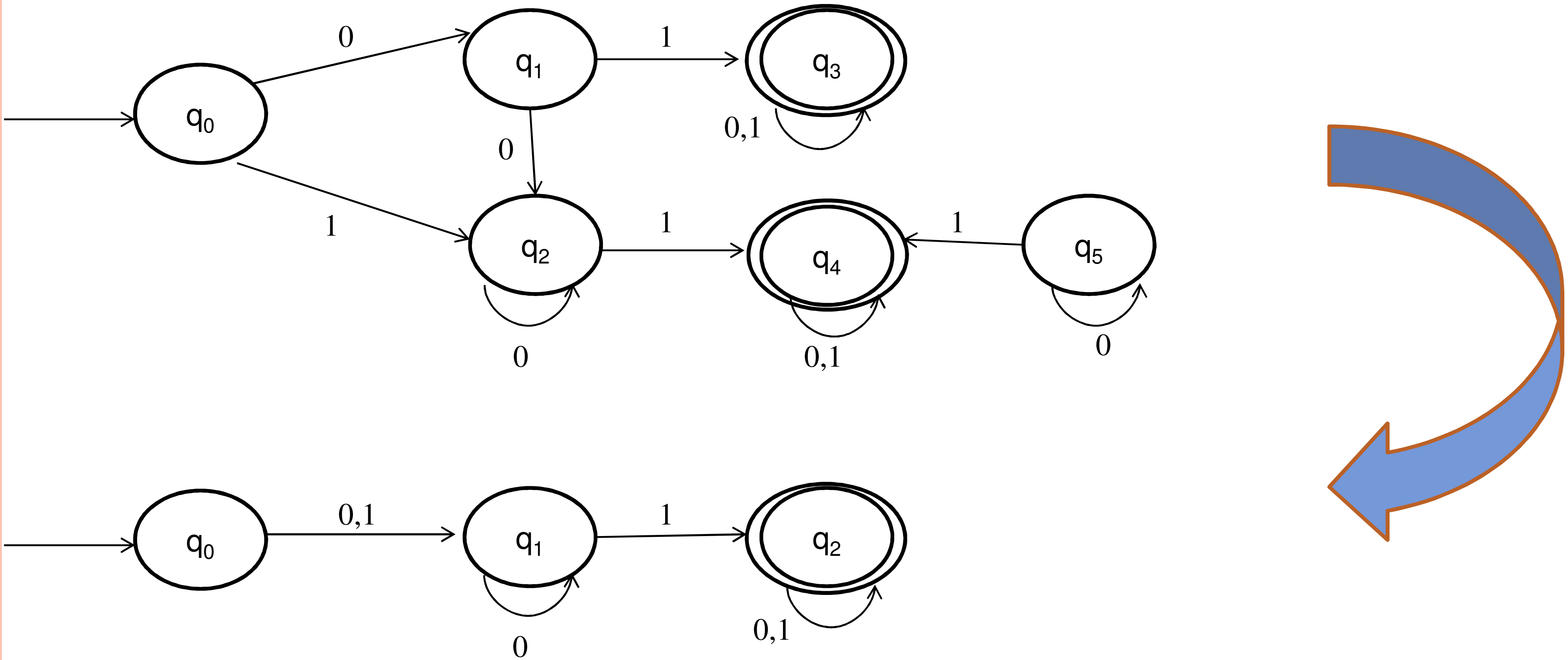
N



$F' = \{q_0, q_2\}$

SONLU OTOMATLARDA DURUM SAYISI İNDİRGEME

Her DFA tek bir dil tanımlar, ancak bunun tersi doğru değildir. Verilen bir dil için, bu dili kabul eden birçok DFA vardır. Bu eşdeğer otomatlar arasındaki en önemli farklılık durumlarının sayısıdır. Örneğin;



Bu iki DFA birbirine eşdeğerdir. q_5 durumu otomatta hiçbir işe yaramaz, çünkü q_0 başlangıç durumundan hiçbir zaman q_5 durumuna erişilemez. İkinci DFA'nın tercih edilme sebebi basitliği ve depolama kolaylığıdır. Bu nedenle DFA'larda mümkün olduğunca durum sayısını azaltmak gerekir.

Tanım: DFA'nın iki durumu p ve q için, tüm $w \in \Sigma^*$ olmak üzere

$\delta^*(p, w) \in F$ iken $\delta^*(q, w) \in F$

ve

$\delta^*(p, w) \notin F$ iken $\delta^*(q, w) \notin F$

ise p ve q durumları ayırt edilemeyen (indistinguishable) durumlardır.

$\delta^*(p, w) \in F$ ve $\delta^*(q, w) \notin F$ ise

p ve q durumları ayırt edilebilir (distinguishable) durumlardır.

p ve q durumları ayırt edilemeyen durumlar ve

q ve r durumları da ayırt edilemeyen durumlar ise

p ve r durumları da ayırt edilemez.

Bir DFA’nın durumlarının sayısını azaltmanın yöntemlerinden bir tanesi ayırt edilemeyen durumları bulmaya ve bunları birleştirmeye dayanır.

→Ayırt edilemeyen durum çiftlerinin bulunması yöntemi:

mark → işaretle yöntemi

1- Erişilemeyen tüm durumları DFA’dan çıkar. (Başlangıç durumundan başlayarak tüm basit yolları bul. Bu yollardan herhangi birine ait olmayan durum, erişilemeyen durumdur.)

2- Tüm (p,q) durum çiftlerini incele. Eğer $p \in F$ ve $q \notin F$ ya da tam tersi ise (p,q) çiftini ayırt edilebilen durum olarak işaretle.

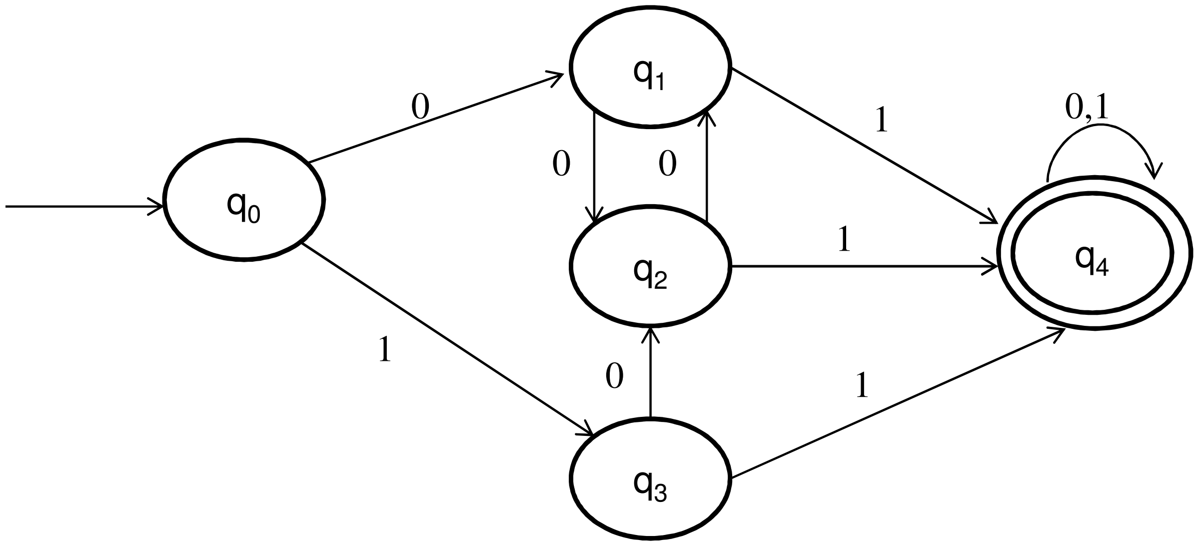
3- Daha önceden (2. aşamada) işaretlenmemiş tüm durum çiftleri işaretlenene kadar bu adımı tekrarla:

Tüm (p,q) çiftleri ve $a \in \Sigma$'lar için

$\delta(p, a) = p_a$ ve $\delta(q, a) = q_a$ hesapla.

Eğer (p_a, q_a) çifti ayırt edilebilen olarak işaretlenmişse (p, q) çiftini de işaretle.

- Örnek:



İşaretle yönteminin 2. aşamasına göre;
(q_0, q_4) (q_1, q_4) (q_2, q_4) (q_3, q_4) $\rightarrow$ ayırt edilebilen çiftler
Kalan çiftler:
(q_0, q_1) , (q_0, q_2) , (q_0, q_3) , (q_1, q_2) , (q_1, q_3) , (q_2, q_3)

3. aşamaya göre de;

$\left.\begin{array}{l}\delta(q_0, 1) = q_3 \\ \delta(q_1, 1) = q_4\end{array}\right\}$ (q_3, q_4) ayırt edilebilen çiftler olduğu için (q_0, q_1) de ayırt edilebilir.

Bu şekilde devam ederek;

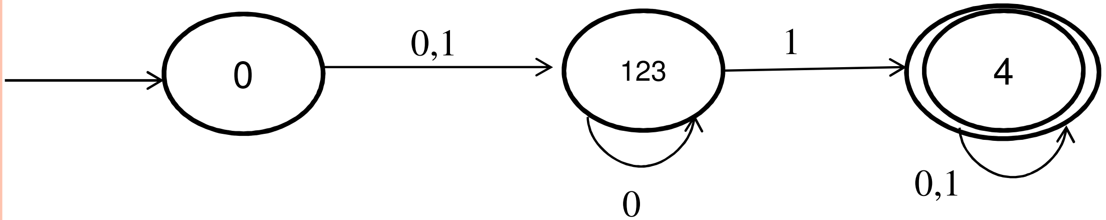
(q_0, q_1) , (q_0, q_2) , (q_0, q_3) , (q_0, q_4) , (q_1, q_4) , (q_2, q_4),(q_3, q_4)

durumları ayırt edilebilen durumlar olarak işaretlenir.

Buna göre ayırt edilemeyen durumlar:

(q_1, q_2) , (q_1, q_3) , (q_2, q_3) → q_1, q_2, q_3 durumları ayırt edilemeyen durumlardır. (Birleştirilir)

$\{q_0\}$, $\{q_1, q_2, q_3\}$, $\{q_4\}$



minimum durumlu DFA

(q_0, q_2)

$\delta(q_0, 1) = q_3$
$\delta(q_2, 1) = q_4$ $\Big\}$ (q_3, q_4)

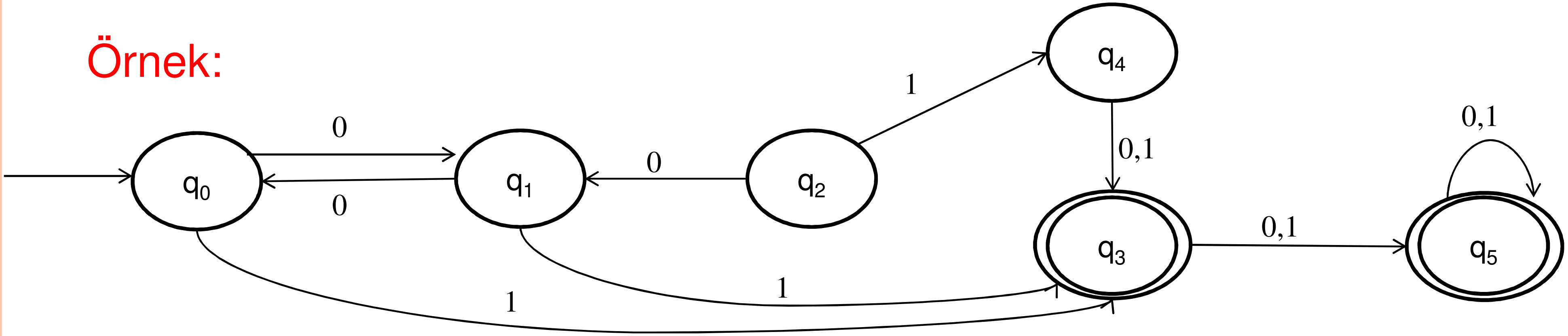
(q_0, q_3)

$\delta(q_0, 1) = q_3$
$\delta(q_3, 1) = q_4$ $\Big\}$ (q_3, q_4)

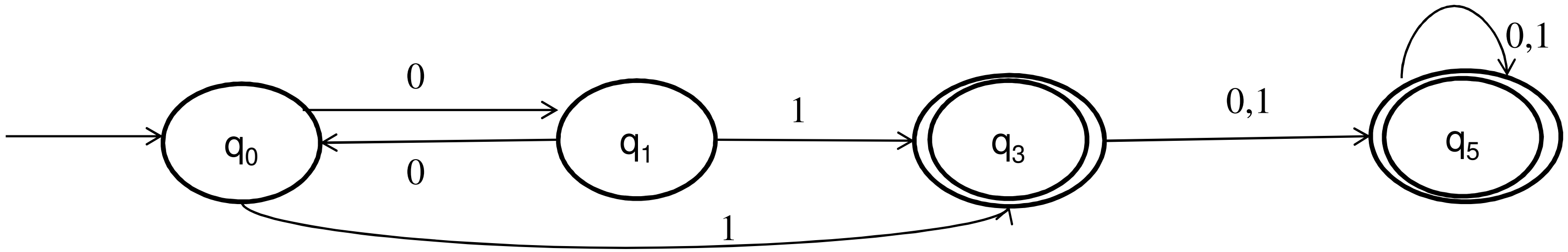
(q_1, q_2)

$\delta(q_1, 1) = q_4$ $\quad$ $\delta(q_1, 0) = q_2$
$\delta(q_2, 1) = q_4$ $\quad$ $\delta(q_2, 0) = q_1$

Örnek:



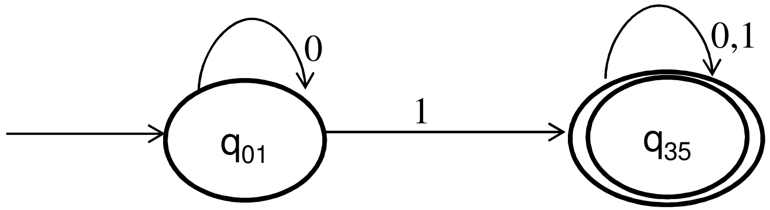
q_2 ve q_4 durumları erişilemeyen durumlar olduğu için öncelikle bu durumlar DFA’dan çıkarılır.



(q_0, q_3) , (q_1, q_3) , (q_0, q_5) , (q_1, q_5) ayırt edilebilen çiftler olarak belirlenir.

(q_0, q_1) , (q_3, q_5) ayırt edilemeyen durumlar olarak belirlenir.

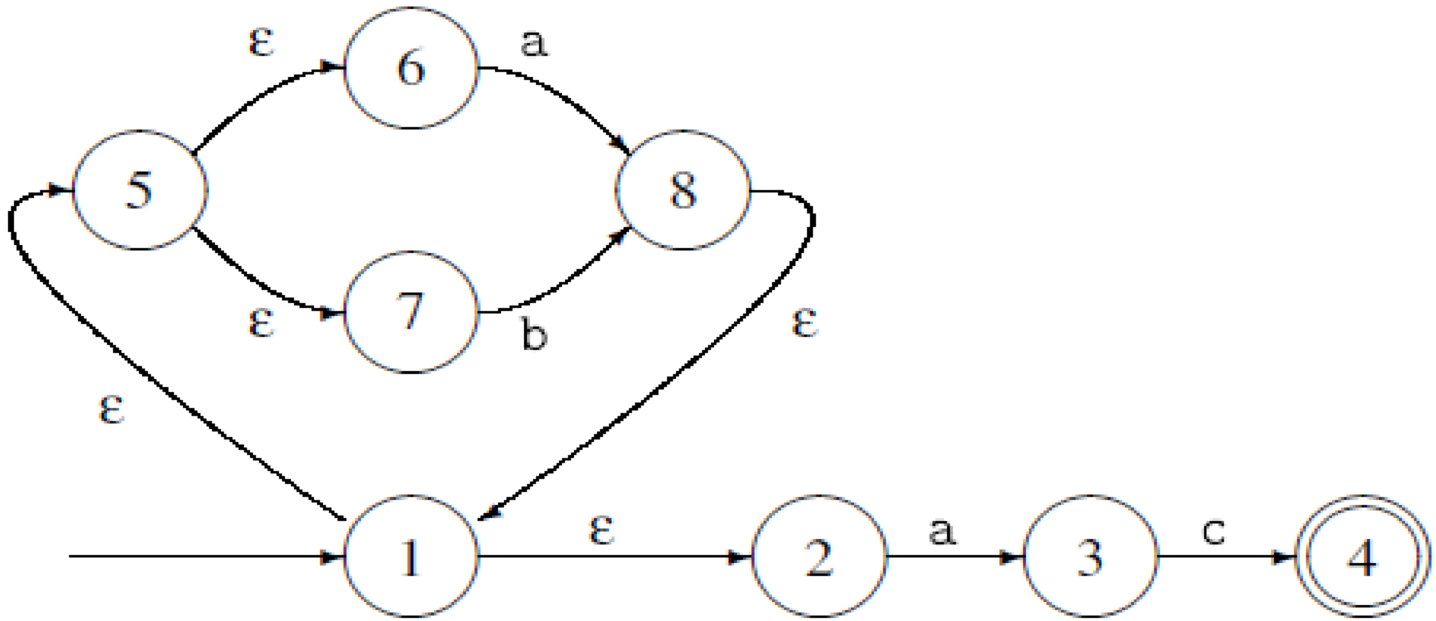
- Bu şekilde minimal DFA şu şekilde belirlenir:



Ɛ-closure

Bir küme durumun **ε-kapanma** (ε-closure)'sı her hangi bir sayıda ε-geçişini kullanarak erişilebilen tüm durumlar ile genişleyen küme olarak tanımlanır.

NFA 'nın {1} durumu kümesinin Ɛ-kapanışı(Ɛ-closure)'nın hesaplanması:



Şekil 1

Ɛ-closure

$$\varepsilon\text{-}closure(M) = M \cup \{t \mid s \in \varepsilon\text{-}closure(M) \text{ and } s^{\varepsilon}t \in T\}$$

T, NFA'daki geçişlerin grubudur.

$$F(X) = M \cup \{t \mid s \in X \text{ and } s^{\varepsilon}t \in T\}$$

M = {1} için →

$$\begin{aligned} F(\emptyset) &= \{1\} \cup \{t \mid s \in \emptyset \text{ and } s^{\varepsilon}t \in T\} \\ &= \{1\} \end{aligned}$$

$$\begin{aligned} F(\{1\}) &= \{1\} \cup \{t \mid s \in \{1\} \text{ and } s^{\varepsilon}t \in T\} \\ &= \{1\} \cup \{2,5\} = \{1,2,5\} \end{aligned}$$

$$\begin{aligned} F(\{1,2,5\}) &= \{1\} \cup \{t \mid s \in \{1,2,5\} \text{ and } s^{\varepsilon}t \in T\} \\ &= \{1\} \cup \{2,5,6,7\} = \{1,2,5,6,7\} \end{aligned}$$

$$\begin{aligned} F(\{1,2,5,6,7\}) &= \{1\} \cup \{t \mid s \in \{1,2,5,6,7\} \text{ and } s^{\varepsilon}t \in T\} \\ &= \{1\} \cup \{2,5,6,7\} = \{1,2,5,6,7\} \end{aligned}$$

Sabit bir noktaya ulaşıldı ve çözüm bulundu. Bu nedenle, Ɛ-kapanışı ({1}) = {1, 2, 5, 6, 7} sonucuna ulaşılır.

Alt Küme Oluşturma

DFA'daki her bir durum NFA'daki durumların bir alt kümesidir.

S durumları, $s_0 \in S$ başlangıç durumu, $F \subseteq S$ tanıma durumları, T geçişleri ve $\sum$ alfabesi ile bir NFA verilsin. Buna göre S' durumları, başlangıç durumu s'_0, tanıma durumları F' ve geçiş fonksiyonu *move* ile eşit bir DFA :

$$
\begin{aligned}
s'_0 &= \varepsilon\text{-closure}(\{s_0\}) \\
move(s', c) &= \varepsilon\text{-closure}(\{t \mid s \in s' \text{ and } s^c t \in T\}) \\
S' &= \{s'_0\} \cup \{move(s', c) \mid s' \in S', c \in \Sigma\} \\
F' &= \{s' \in S' \mid s' \cap F \neq \emptyset\}
\end{aligned}
$$

Alt Küme Oluşturma

Sonuçta:

$$\begin{aligned} move(s_0', \mathtt{a}) &= \varepsilon\text{-}closure(\{t \mid s \in \{1,2,5,6,7\} \text{ and } s^{\mathtt{a}}t \in T\}) \\ &= \varepsilon\text{-}closure(\{3,8\}) \\ &= \{3,8,1,2,5,6,7\} \\ &= s_1' \end{aligned}$$

$$\begin{aligned} move(s_0', \mathtt{b}) &= \varepsilon\text{-}closure(\{t \mid s \in \{1,2,5,6,7\} \text{ and } s^{\mathtt{b}}t \in T\}) \\ &= \varepsilon\text{-}closure(\{8\}) \\ &= \{8,1,2,5,6,7\} \\ &= s_2' \end{aligned}$$

$$\begin{aligned} move(s_0', \mathtt{c}) &= \varepsilon\text{-}closure(\{t \mid s \in \{1,2,5,6,7\} \text{ and } s^{\mathtt{c}}t \in T\}) \\ &= \varepsilon\text{-}closure(\{\}) \\ &= \{\} \end{aligned}$$

elde edilir.

Alt Küme Oluşturma

Unutulmamalıdır ki NFA durumlarının boş kümesi bir DFA durumu değildir. Bu nedenle c üzerinden bir geçiş s'_0 için mevcut değildir.

Şu anda S' için s'_1 ve s'_2 eklendi. Şu andaki S' kümesi { $s'_{0,}$ s'_1, s'_2 } şeklindedir. Şimdi s'_1 alınır, işaretlenir ve geçişleri hesaplanır.

$$\begin{aligned} move(s'_1, \mathtt{a}) &= \varepsilon\text{-}closure(\{t \mid s \in \{3,8,1,2,5,6,7\} \text{ and } s^{\mathtt{a}}t \in T\}) \\ &= \varepsilon\text{-}closure(\{3,8\}) \\ &= \{3,8,1,2,5,6,7\} \\ &= s'_1 \end{aligned}$$

$$\begin{aligned} move(s'_1, \mathtt{b}) &= \varepsilon\text{-}closure(\{t \mid s \in \{3,8,1,2,5,6,7\} \text{ and } s^{\mathtt{b}}t \in T\}) \\ &= \varepsilon\text{-}closure(\{8\}) \\ &= \{8,1,2,5,6,7\} \\ &= s'_2 \end{aligned}$$

$$\begin{aligned} move(s'_1, \mathtt{c}) &= \varepsilon\text{-}closure(\{t \mid s \in \{3,8,1,2,5,6,7\} \text{ and } s^{\mathtt{c}}t \in T\}) \\ &= \varepsilon\text{-}closure(\{4\}) \\ &= \{4\} \\ &= s'_3 \end{aligned}$$

Alt Küme Oluşturma

Mevcut durumlara s’$_3$ eklendi. Şimdi s’$_2$ alınır, işaretlenir ve geçişleri hesaplanır.

$$\begin{aligned}
move(s'_2, \mathtt{a}) &= \varepsilon\text{-}closure(\{t \mid s \in \{8,1,2,5,6,7\} \text{ and } s^{\mathtt{a}}t \in T\}) \\
&= \varepsilon\text{-}closure(\{3,8\}) \\
&= \{3,8,1,2,5,6,7\} \\
&= s'_1
\end{aligned}$$

$$\begin{aligned}
move(s'_2, \mathtt{b}) &= \varepsilon\text{-}closure(\{t \mid s \in \{8,1,2,5,6,7\} \text{ and } s^{\mathtt{b}}t \in T\}) \\
&= \varepsilon\text{-}closure(\{8\}) \\
&= \{8,1,2,5,6,7\} \\
&= s'_2
\end{aligned}$$

$$\begin{aligned}
move(s'_2, \mathtt{c}) &= \varepsilon\text{-}closure(\{t \mid s \in \{8,1,2,5,6,7\} \text{ and } s^{\mathtt{c}}t \in T\}) \\
&= \varepsilon\text{-}closure(\{\}) \\
&= \{\}
\end{aligned}$$

Alt Küme Oluşturma

Yeni bir eleman mevcut durumlara eklenmedi. Bu nedenle işaretlenmemiş elemanlardan s'$_3$ alınır, işaretlenir ve geçişleri hesaplanır.

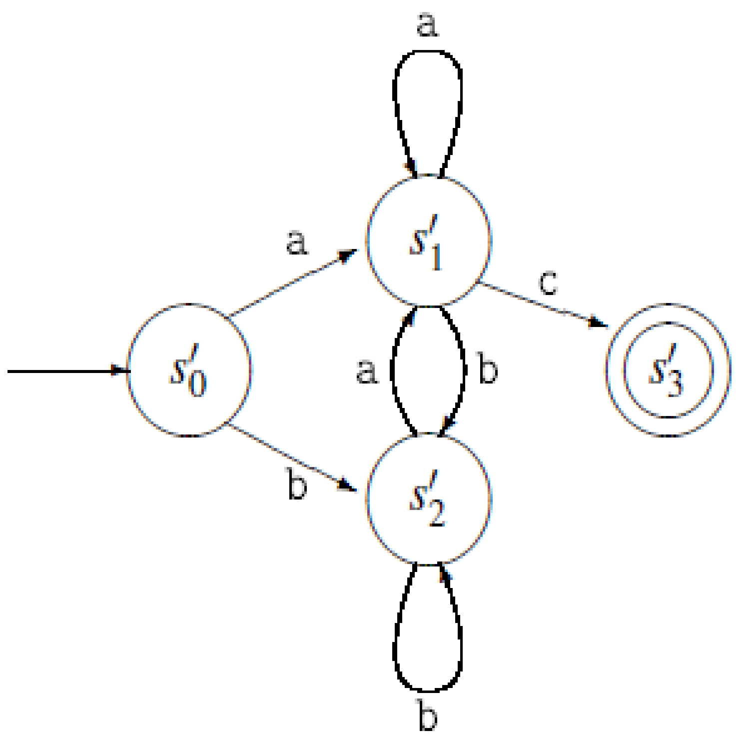
$$\begin{aligned} move(s'_3, \mathtt{a}) &= \varepsilon\text{-}closure(\{t \mid s \in \{4\} \text{ and } s^{\mathtt{a}} t \in T\}) \\ &= \varepsilon\text{-}closure(\{\}) \\ &= \{\} \end{aligned}$$

$$\begin{aligned} move(s'_3, \mathtt{b}) &= \varepsilon\text{-}closure(\{t \mid s \in \{4\} \text{ and } s^{\mathtt{b}} t \in T\}) \\ &= \varepsilon\text{-}closure(\{\}) \\ &= \{\} \end{aligned}$$

$$\begin{aligned} move(s'_3, \mathtt{c}) &= \varepsilon\text{-}closure(\{t \mid s \in \{4\} \text{ and } s^{\mathtt{c}} t \in T\}) \\ &= \varepsilon\text{-}closure(\{\}) \\ &= \{\} \end{aligned}$$

Alt Küme Oluşturma

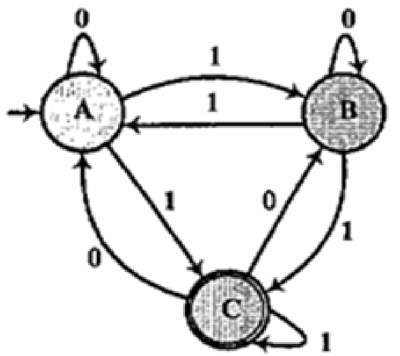
İşlemler tamamlandığında S’ = {s’$_0$, s’$_1$, s’$_2$, s’$_3$ }elde edilir. Bu durumlardan sadece s’$_3$ NFA’nın 4. durumunun tanıma özelliğini içerir. Şekil-2 tamamlanmış DFA’yı göstermektedir.



Şekil 2

Örnek 1

Aşağıda verilen NFA'yı bir DFA'ya dönüştürünüz



$M_{1.6} = < Q, \Sigma, \delta, q_0, F >$

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

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

$q_0 = A$

$F = \{C\}$

δ :

$\delta(A, 0) = A$

$\delta(A, 1) = BC$

$\delta(BC, 0) = AB$

$\delta(BC, 1) = AC$

$\delta(AB, 0) = AB$

$\delta(AB, 1) = ABC$

$\delta(AC, 0) = AB$

$\delta(AC, 1) = BC$

$\delta(ABC, 0) = AB$

$\delta(ABC, 1) = ABC$

	0	1
→A	A	BC
BC	AB	AC
AB	AB	ABC
AC	AB	BC
ABC	AB	ABC

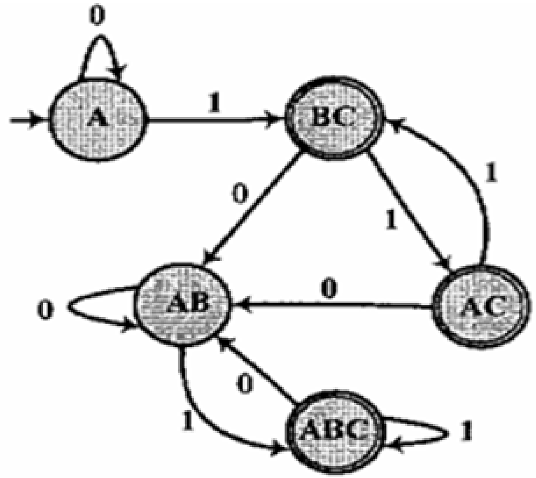
$M_{D1.6} = < Q, \Sigma, \delta, q_0, F >$

$Q = \{A, BC, AB, AC, ABC\}$

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

$q_0 = A$

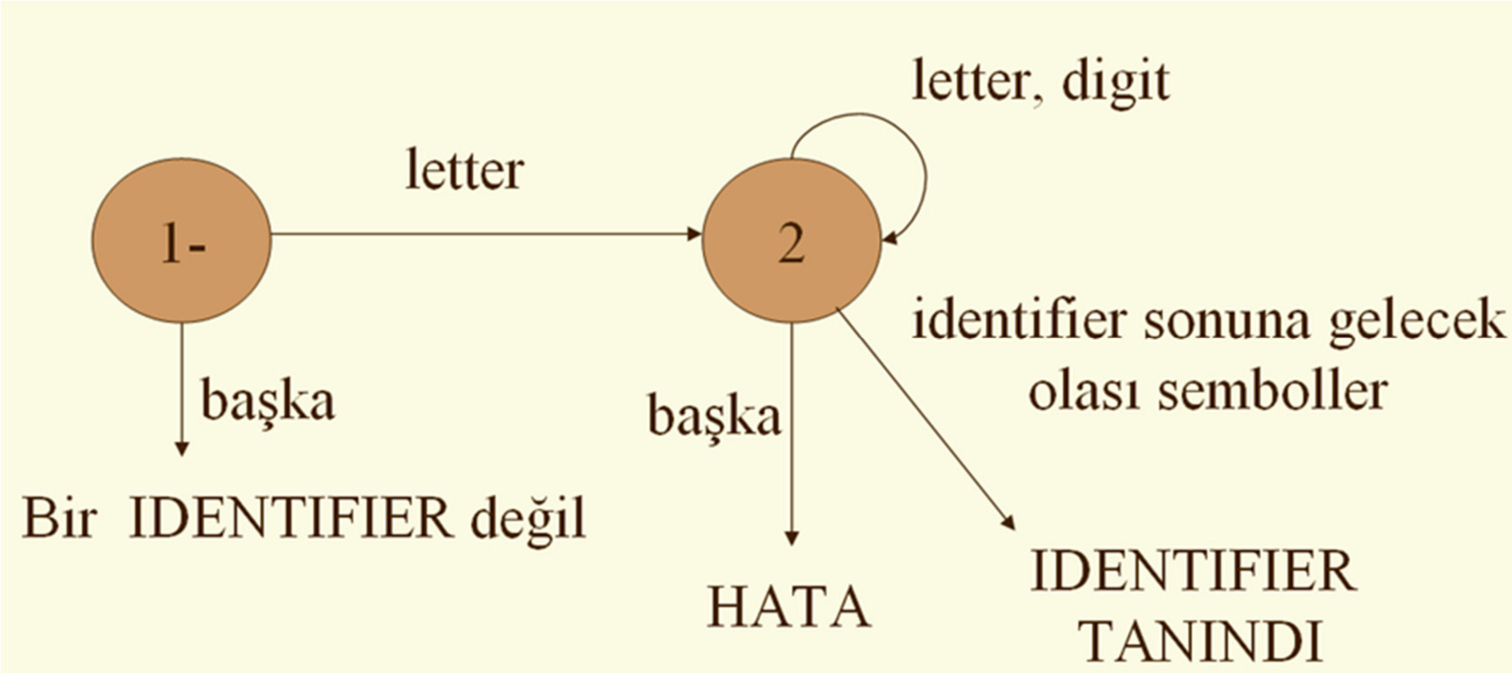
$F = \{BC, AC, ABC\}$



Örnek 2

Identifier → letter (letter | digit)*

Yukarıda tanımlanan bir programlama diline ilişkin “identifier” ları tanıyan F.A. tasarlayınız.



Örnek 3

Aşağıdaki kurallı ifade için bir FA oluşturunuz.

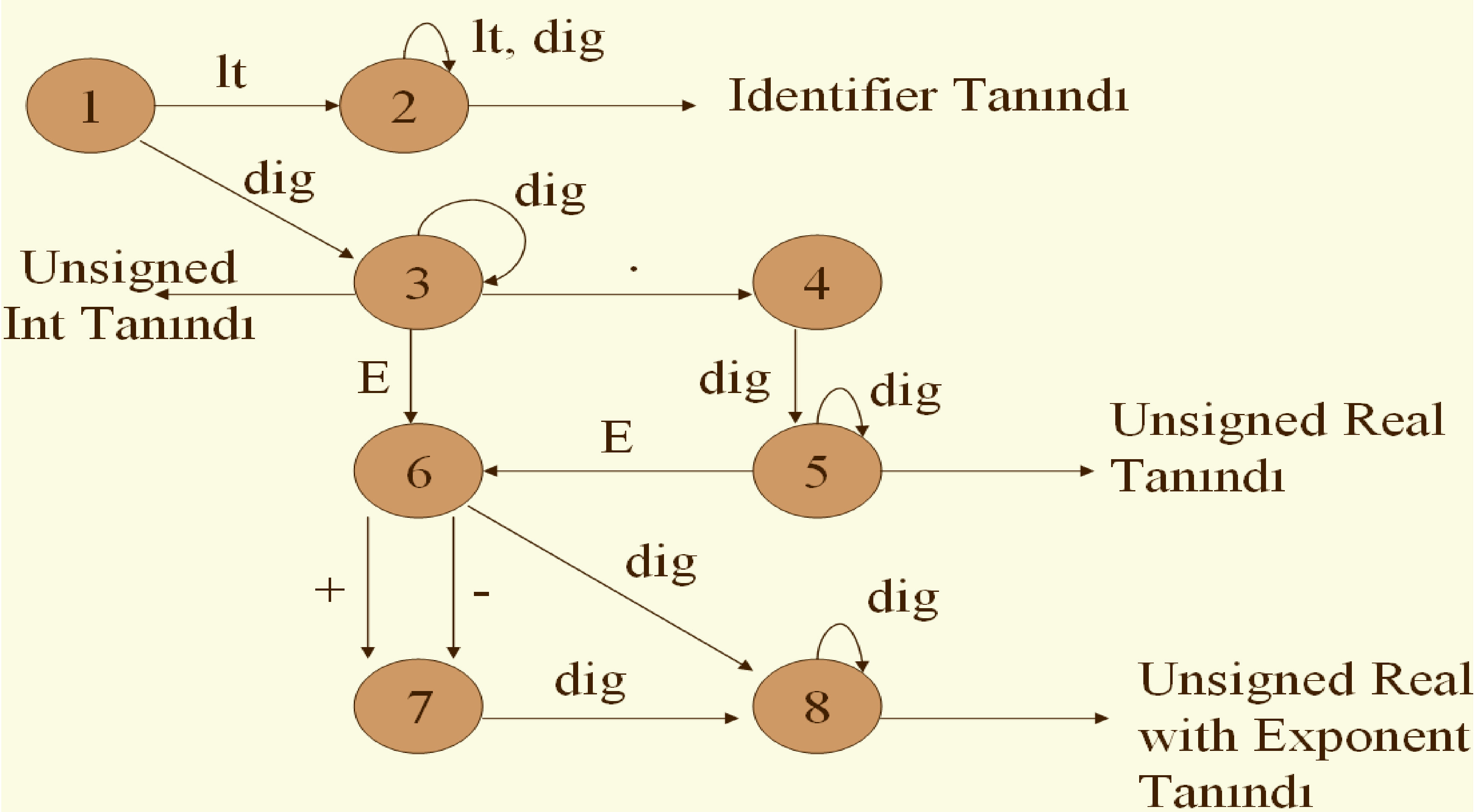
identifer→letter (letter | digit)*

unsigned-int → digit(digit)*

unsigned-real1 → digit(digit)*.digit(digit)*

unsigned-real2 → unsigned-real1 E[+|-]? digit(digit)* | unsigned-int E[+|-]? digit(digit)*

Örnek 3



1
lt
2
lt, dig
Identifier Tanındı
dig
dig
3
.
4
Unsigned
Int Tanındı
E
dig
dig
6
E
5
Unsigned Real
Tanındı
dig
+
-
dig
7
dig
8
Unsigned Real
with Exponent
Tanındı