CENG 306 Biçimsel Diller ve Otomatlar
Formal Languages and Automata

Hazırlayan: M.Ali Akçayol - Gazi Üniversitesi

Bilgisayar Mühendisligi Bölümü

Konular

- Düzenli ve Düzenli olmayan diller - Regular and Non-regular Languages
- Pumping Lemma and Its Application
- Durum İndirgeme - State Minimization
- Myhill-Nerode Theorem
- Table-Filling Algorithm

Regular and Non-regular Languages

- Düzenli diller bazı işlemler (**birleşim, kesişim, Kleene star, complement, concatenation**) icin kapalıdır.
- Düzenli diller, düzenli ifadeler (regular expressions) ile veya sonlu otomatlar ile (deterministic veya nondeterministic) belirlenebilir.

Regular and Non-regular Languages

- Düzenli diller bazı işlemler (**birleşim, kesişim, Kleene star, complement, concatenation**) icin kapalıdır.
- Düzenli diller, düzenli ifadeler (regular expressions) ile veya sonlu otomatlar ile (deterministic veya nondeterministic) belirlenebilir.

Örnek:

$\sum = \{0, 1, 2, 3, \ldots, 9\}$, $L \subseteq \sum^*$ *olarak tanımlı olsun ve sadece 2'ye veya 3'e bölünebilen ve önünde 0 olmayan pozitif sayılara sahip olsun.*

(0, 3, 6, 244 $\in L$ *ve 1, 03, 00* $\notin L$*)*

Bu dilin regular olduğunun ispatı 4 kısımda yapılabilir.

Regular and Non-regular Languages

Örnek: (devam)

- $\sum = \{0, 1, 2, 3, \ldots, 9\}$, $L \subseteq \sum^*$, *2'ye veya 3'e bölünebilen ve önünde 0 olmayan pozitif sayılara sahiptir*

1 L_1 *dili pozitif tamsayıların kümesi olsun*

$$L_1 = 0 \cup \{1, 2, \ldots 9\} \sum^* \quad \textit{(regular)}$$

Regular and Non-regular Languages

Örnek: (devam)

- $\sum = \{0, 1, 2, 3, ..., 9\}$, $L \subseteq \sum^*$, *2'ye veya 3'e bölünebilen ve önünde 0 olmayan pozitif sayılara sahiptir*

1 L_1 *dili pozitif tamsayıların kümesi olsun*

$$L_1 = 0 \cup \{1, 2, ... 9\} \sum^* \quad \textit{(regular)}$$

2 L_2 *dili 2'ye bölünebilen pozitif tamsayıların kümesi olsun*

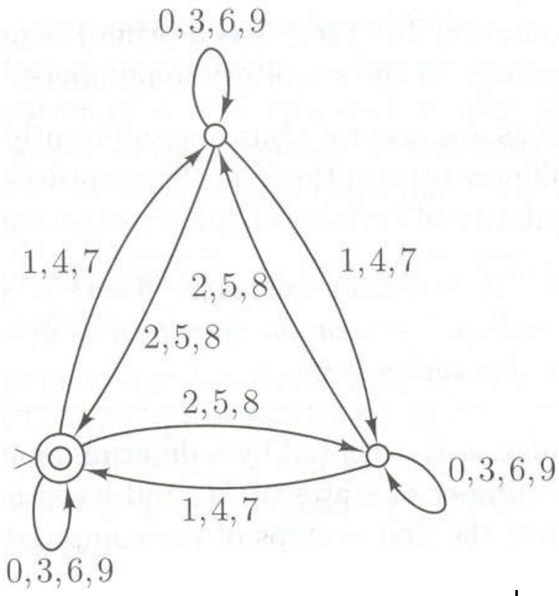
$$L_2 = L_1 \cap \sum^* \{0, 2, 4, 6, 8\} \quad \textit{(regular)}$$

Regular and Non-regular Languages

Örnek: (devam)

- $\Sigma = \{0, 1, 2, 3, ..., 9\}$, $L \subseteq \Sigma^*$, *2'ye veya 3'e bölünebilen ve önünde 0 olmayan pozitif sayılara sahiptir*

3 *3'e bölünebilen pozitif tamsayıların kümesi* olan dili *yandaki* otomat *tanır (regular)*

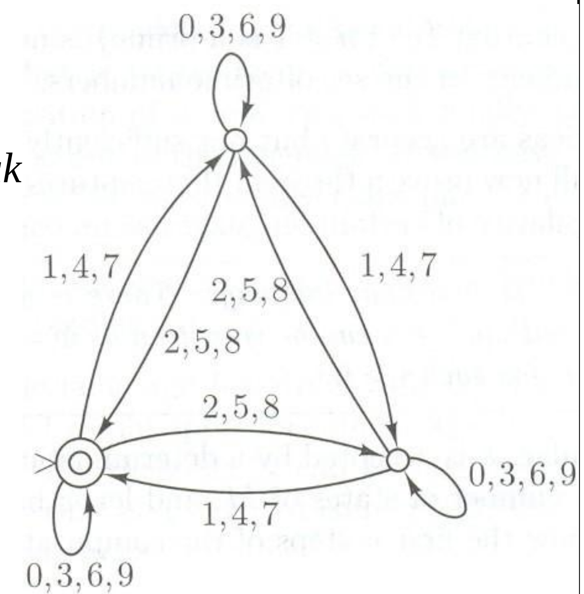


Regular and Non-regular Languages

Örnek: (devam)

- *Σ = {0, 1, 2, 3,..., 9}, $L \subseteq \Sigma^*$, 2'ye veya 3'e bölünebilen ve önünde 0 olmayan pozitif sayılara sahiptir*

3 *3'e bölünebilen pozitif tamsayıların kümesi olan dili yandak otomat tanır (regular)*



4 *L_3 bu otomat ile L_1'in kesişimidir. Sonuç olarak elde edilen dil regular dildir ve*

$L = L_2 \cup L_3$, şeklinde ifade edilir.

Regular and Non-regular Languages

- Bir dilin düzenli dil olduğunu gösteren yöntemler vardır ancak düzenli olmadığını göstermek için özel araçlara ihtiyaç vardır.

Regular and Non-regular Languages

- Bir dilin düzenli dil olduğunu gösteren yöntemler vardır ancak düzenli olmadığını göstermek için özel araçlara ihtiyaç vardır.
- **İki özellik** tüm düzenli dillerde bulunur, ancak düzenli olmayan dillerde bulunmaz;
  - Bir string soldan sağa doğru taranırken o string'in ilgili dile ait olup olmadığını belirlemek için gereken **toplam hafızanın bir sınırı vardır**, bu sınır sabittir ve ilgili string'e degil o dile baglıdır.

    Örnek: $L = \{ a^n b^n : n \geq 0 \}$ dili regular degildir. b'leri okumaya başladığında kaç tane a okuduğu belli değildir ve n için sınır değer yoktur.

Regular and Non-regular Languages

- Bir dilin düzenli dil olduğunu gösteren yöntemler vardır ancak düzenli olmadığını göstermek için özel araçlara ihtiyaç vardır.
- **İki özellik** tüm düzenli dillerde bulunur, ancak düzenli olmayan dillerde bulunmaz;
  - Bir string soldan sağa doğru taranırken o string'in ilgili dile ait olup olmadığını belirlemek için gereken **toplam hafızanın bir sınırı vardır**, bu sınır sabittir ve ilgili string'e degil o dile baglıdır.

    Örnek: L = { $a^n b^n$: n ≥ 0 } dili regular degildir. b'leri okumaya başladığında kaç tane a okuduğu belli değildir ve n için sınır değer yoktur.
  - Sonsuz sayıda string'e sahip olan düzenli diller, döngüye sahip **otomatlar** veya Kleene star içeren **regular expression'lar** tarafından gösterilebilir.

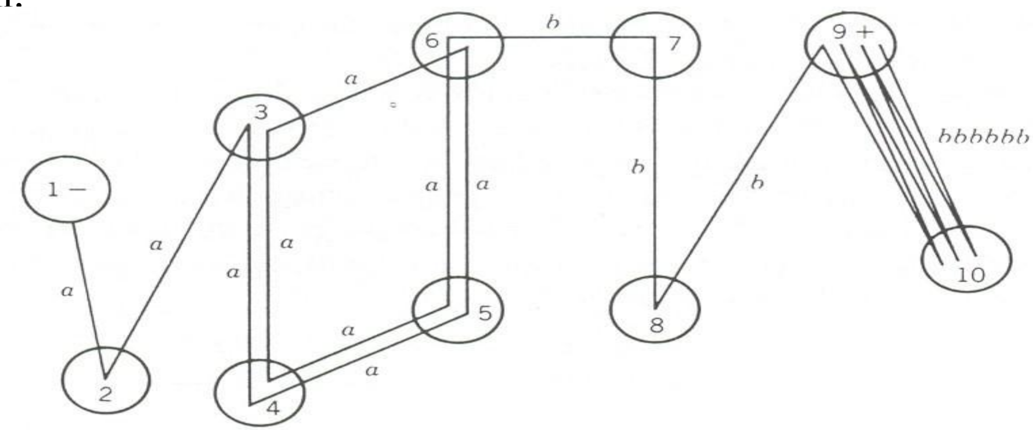
    Örnek: L = { a^n : n ≥ 1 asal sayı} regular değildir. (ispatı daha sonra verilecektir.)

Regular and Non-regular Languages

- ***$L = \{ a^n b^n : n \geq 0 \}$*** şeklinde tanımlanmış olsun. Eğer bu dil regular ise bir sonlu otomat tarafından tanınır.
- *$n = 96$* için *$a^{96}b^{96}$ olur.* Toplam 95 duruma sahip bir otomat bu dili tanıyor olsun.
- En az bir noktada yol daha önce geçtiği durumlara geri döner ve tekrar geçer.
- Bu tekrar geçişlere **loop** adı verilir.

Regular and Non-regular Languages

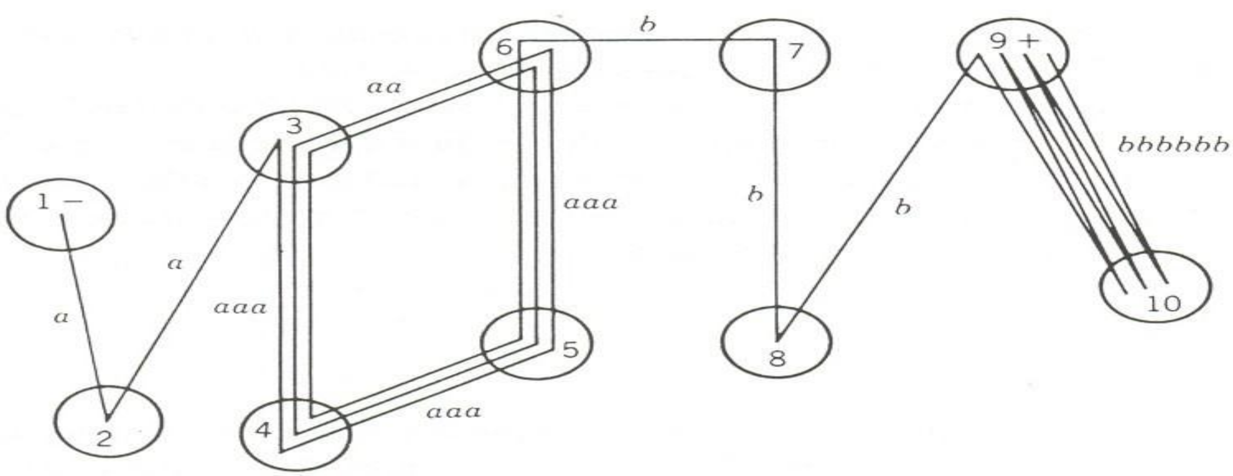
- $L = \{ a^n b^n : n \geq 0 \}$ şeklinde tanımlanmış olsun. Eğer bu dil regular ise bir sonlu otomat tarafından tanınır.
- $n = 96$ için $a^{96}b^{96}$ *olur.* Toplam 95 duruma sahip bir otomat bu dili tanıyor olsun.
- En az bir noktada yol daha önce geçtiği durumlara geri döner ve tekrar geçer.
- Bu tekrar geçişlere **loop** adı verilir.
- Aşağıdaki 10 durumlu otomat a^9b^9 için geçişleri vermektedir. Otomatta sadece geçilen yollar verilmiştir.



- $a^{13}b^9$ için nasıl bir yol izlenir ?

Regular and Non-regular Languages

- $a^{13}b^{9}$ *için 6-3-4-5 yolunda bir tur daha atılır.*
- a^{9}**$(a^{4})^{m}$**b^{9} *, $m \geq 0$ şeklinde tanımlanan tüm stringleri tanır. (Örn : $a^{25}b^{9}$)*
- *Bu şekilde bir string'in önündeki ve sonundakine bakmadan ortasına ekleme yapmaya* ***pumping*** *denilmektedir.*
- *string'in önündeki ve/veya sonundaki* ***boş olabilir****.*



Regular and Non-regular Languages

Teorem: *Sonsuz sayıda string'e sahip bir regular L dilinde, kendisini tanıyan otomatın durum sayısından fazla sembole sahip tüm stringler için* ***x, y, z*** *şeklinde üç substring tanımlanabilir ve tüm stringler* ***xy^nz*** *olarak parçalarla ifade edilebilir. (n = 1, 2, 3, ...)*

ispat: *L dilinde sonsuz string olduğu için bazı w stringleri kendisini tanıyan otomatın durum sayısından daha fazla sembole sahiptir. Bu stringler için otomat üzerinde* ***loop*** *oluşur.*

w string'leri x, y, z olarak üç kısımda incelenebilir;

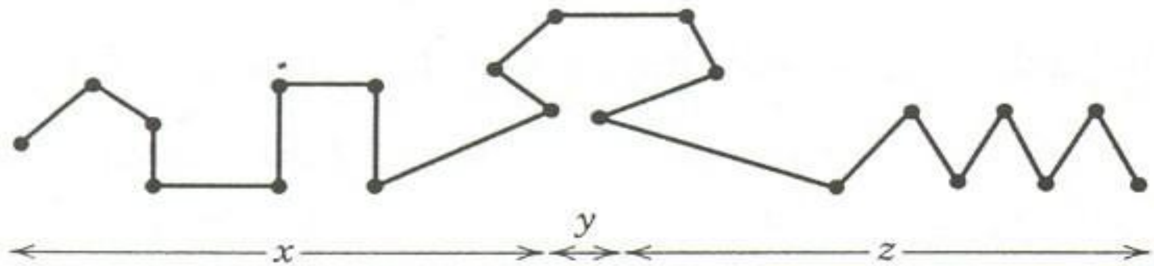
1. *x yeniden geçilen ilk duruma kadar olan sembolleri içersin. x eğer boş ise loop başlangıç durumunu da içine almıştır.*
2. *y string'i, x' den hemen sonra başlayıp loop'un sonuna kadar olan kısmı içersin. Bir loop oluştuğu için y boş olamaz.*
3. *z string'i loop'tan hemen sonra başlayıp w string'inin sonuna kadar olan kısmı içersin. Z boş ise loop sonuç durumunu da içine almıştır.*

Regular and Non-regular Languages

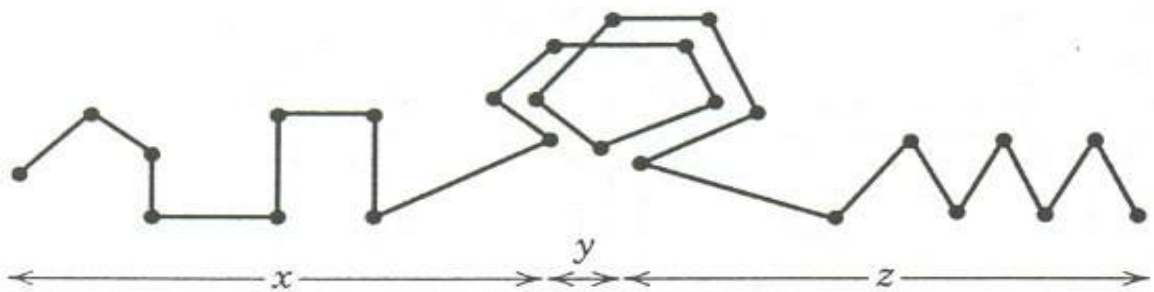
Teorem: (devam) *Sonsuz sayıda string'e sahip bir regular L dilindeki tüm stringler için* ***x, y, z*** *şeklinde üç string tanımlanabilir ve* ***tüm stringler xy^{n}z*** *olarak parçalarla ifade edilebilir. (n = 1, 2, 3, ...)*

xyz, xyyz, xyyyz, ...xy^{n}z string'leri tanınır.

xyz



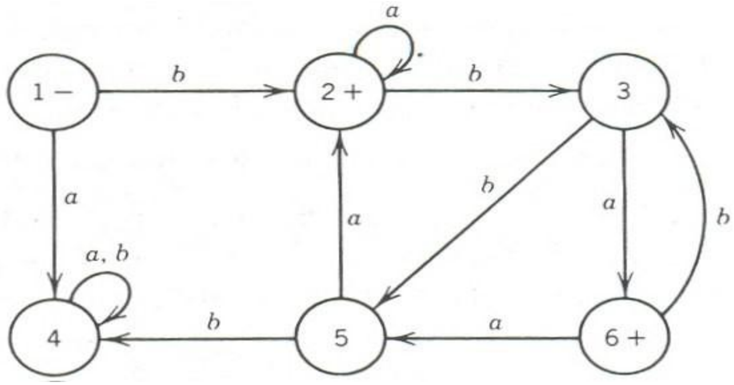
Xyyz



Regular and Non-regular Languages

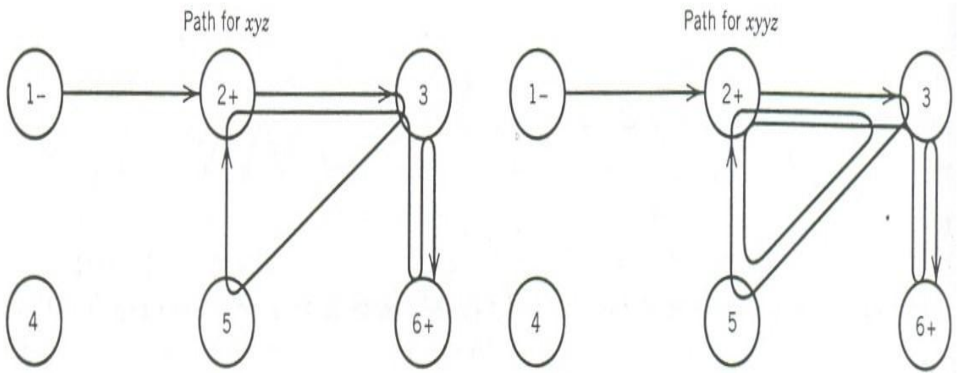
Örnek : Aşağıdaki otomatta "-" başlangıç ve "+" sonuç durumlarını göstermektedir ve

w = bbbababa string'ini tanır.



Durum sayısından fazla sembol oldugu için loop olmak zorundadır.

x = b, y = bba ve z = baba alınabilir.



Regular and Non-regular Languages

Örnek : *$L = \{a^n b^n : n \geq 0\}$ dili regular değildir. Eger L regular olsaydı tüm string'ler x, y, z olarak üç parçaya ayrılabilirdi.*

Tipik bir string aaa.....aaaabbbb......bbb şeklindedir.

- *Eger y, a'ları içerirse xyyz şeklindeki string daha fazla a içerir ve elde edilen string L diline ait değildir.*
- *Eger y, b'leri içerirse xyyz şeklindeki string daha fazla b içerir ve elde edilen string L diline ait değildir.*
- *Eger y, ortadaki a ve b'leri içerirse xyyz string'i iki tane "ab" substringine sahiptir. L dilindeki tüm stringler sadece bir tane "ab" substringine sahip olabileceği için bu string L diline ait değildir.*

Bunların sonucu olarak L dili regular değildir.

Pumping Lemma

Teorem: ***L*** *regular dil olsun. Dile bağlı olarak seçilen bir*

$\boldsymbol{n \geq 1}$ *için* $|\boldsymbol{w}| \geq \boldsymbol{n}$ *olacak şekilde bir* $\boldsymbol{w} \in \boldsymbol{L}$ *string'i vardır ve*

$$w = xyz,$$

$$y \neq e,$$

$$|xy| \leq n$$

olmak üzere *yeniden yazılabilir.* ***Her*** $\boldsymbol{i \geq 0}$ *için* $\boldsymbol{x y^{i} z} \in \boldsymbol{L}$ *olur.*

İspat: ***L*** *regular dil olduğundan deterministic finite automata* ***M*** *tarafından kabul edilir.* ***M*** *automata'nın* ***n*** *duruma sahip olduğunu varsayalım ve*

$|\boldsymbol{w}| = \boldsymbol{m}$, $\boldsymbol{m} \geq \boldsymbol{n}$ *olsun.*

M *automata'nın ilk* ***m*** *adımı aşağıdaki gibidir;*

$$(q_0, w_1 w_2 \ldots w_m) \vdash_M (q_1, w_2 \ldots w_m) \vdash_M \ldots \vdash_M (q_m, e)$$

Pumping Lemma

ispat: *(devam)*

$(q_0, w_1w_2...w_m) \vdash_M (q_1, w_2...w_m) \vdash_M ... \vdash_M (q_m, e)$

- *Burada* q_0 *başlangıç durumu ve* $w_1w_2...w_m$ *ilk m semboldür*
- M, n *adet duruma sahiptir ancak* $m \geq n$ *adet konfigürasyon vardır.*
- *Pigeonhole prensibine göre* $0 \leq i < j \leq m$ *olacak şekilde* i *ve* j *sayıları vardır ve* $q_i = q_j$ *olur.*
- $y = w_i w_{i+1} ... w_j$ *vardır ve* q_i *durumundan tekrar* q_i *durumuna geçiş yapar.*
- $i < j$ *oldugu için* y *boş olamaz.*
- y *string'i* w*'dan atılarak veya istendigi kadar tekrar edilerek bulunan stringler'de* M *tarafından tanınır.*
- $i \geq 0$ *olmak üzere* $xy^iz \in L$ *olur.*

Pumping Lemma

- Bir dilin regular olup olmadığını belirlemek icin kullanılır.
  - Öncelikle bir *n* sayısı belirlenir. (dili tanıyan ve en az duruma sahip otomat)
  - *n*'den uzun bir *w* string'i belirlenir.
  - *w* string'i *xyz* şeklinde parçalanır.
  - her $i \geq 0$ degeri için $xy^iz \in L$ olacak şekilde bir xy^iz bulunuyorsa L regulardır..
  - Eger bu şekilde bir değer hiçbir xy^iz için bulunamıyorsa *L* regular değildir.

Pumping Lemma

Örnek: (tekrar) $L = \{ a^i b^i : i \geq 0 \}$ *dili regular degildir.*

ispat:

- $w = a^n b^n \in L$ *olduğunu varsayalım.*
- *Pumping teoreminden* $w = xyz$ *yazılabilir.*
- $|xy| \leq n$ *alınırsa ve* $y \neq e$, $y = a^i$, $i > 0$ *değerleri için,*
- *y'nin çıkarıldığı string olan* $xz = a^{n-i} b^n$ *olur ve* L *diline ait değildir.*
- *Bu sonuç* $y = b^i$, $i > 0$ *için de aynı şekilde geçerlidir.*
- *Böylece bu dil regular değildir.*

Pumping Lemma

Örnek: $\boldsymbol{L = \{ a^k : k}$ ***asal sayı }*** *dili regular degildir.*

ispat:

- *Pumping teoreminden* $\boldsymbol{w = xyz}$ *yazılabilir.*
- $\boldsymbol{p, r \geq 0, q > 0}$ *için* $\boldsymbol{x = a^p, y = a^q}$, ***ve*** $\boldsymbol{z = a^r}$ *olsun.*
- *Teoremden* $\boldsymbol{xy^nz \in L}$ *olduğundan* $\boldsymbol{n \geq 0}$ *için* $\boldsymbol{p + nq + r}$

  asal sayı olmak zorundadır. (her $\boldsymbol{n}$ *sayısı için sağlanmalıdır!)*
- ***Özel olarak*** $\boldsymbol{n = p + 2q + r + 2}$ *için* $\boldsymbol{p + nq + r = (q + 1)(p + 2q + r)}$

  olur.
- *Burada iki çarpan da 1'den büyüktür ve böylece*

  $\boldsymbol{p + nq + r}$ *asal sayı olamaz!*
- $\boldsymbol{n = p + 2q + r + 2}$ *için elde edilen string* $\boldsymbol{L}$ *diline ait değildir.*
- *Öyle ise L dili regular değildir.*

Pumping Lemma

Örnek: ***L = { w*** $\in$ ***a, b}**** ***: w eşit sayıda a ve b'ye sahiptir }*** *dili regular değildir.*

ispat:

- *Bu ispat closure özelliği ile yapılabilir.*
- *Eger **L** dili regular ise, regular bir dil ile kesişim işlemi kapalı olur.*
- *Ancak* $\boldsymbol{L \cap a^*b^*}$ *kesişiminin sonucunda elde edilen dil*

  $\boldsymbol{\{a^n b^n : n \geq 0\}}$ *olur.*
- $\boldsymbol{\{a^n b^n : n \geq 0\}}$ *regular dil olmadığı için **L** dili de regular değildir.*

*Örnek: L = {**ab**, abba, **aabb**, abab, **aaabbb**,...},*

$L(a^*b^*) = \{a, b, aa, \mathbf{ab}, aab, bb, \mathbf{aabb}, abbbb, \mathbf{aaabbb}, ...\}$

$L \cap a^*b^* = \{\mathbf{ab, aabb, aaabbb, ...}\}$

EXERCISES: PUMPING LEMMA

Is the following regular or not. Why?

$L = \{ss^R : s \in \{a, b\}^*\}$

Teorem: ***L*** *regular dil olsun. Dile bağlı olarak seçilen bir*

n $\geq$ ***1*** *için* ***|w|*** $\geq$ ***n*** *olacak şekilde herhangi bir* ***w***$\in$ ***L*** *string'i vardır öyleki*

w = xyz*,* ***y*** $\neq$***e****,* ***|xy|***$\leq$ ***n***

olmak üzere *yeniden yazılabilir.* ***Her i*** $\geq$ ***0*** *için* $xy^iz \in L$ *olur.*

EXERCISES: PUMPING LEMMA

Is the following regular or not. Why?

$L = \{ss : s \in \{a, b\}^*\}$

Teorem: ***L*** *regular dil olsun. Dile bağlı olarak seçilen bir* $\boldsymbol{n \geq 1}$ *için* $|\boldsymbol{w}| \geq \boldsymbol{n}$ *olacak şekilde herhangi bir* $\boldsymbol{w} \in \boldsymbol{L}$ *string'i vardır öyleki*

$$w = xyz, \quad y \neq e, \quad |xy| \leq n$$

olmak üzere *yeniden yazılabilir.* ***Her*** $\boldsymbol{i \geq 0}$ *için* $xy^iz \in L$ *olur.*

EXERCISES: PUMPING LEMMA

Is the following regular or not. Why?

L = {ww' : w $\in$ {a, b}*}, where w' stands for w with each occurrence of a replaced by b, and vice versa.

Teorem: ***L*** *regular dil olsun. Dile bağlı olarak seçilen bir*

n $\geq$ ***1*** *için* ***|w|*** $\geq$ ***n*** *olacak şekilde herhangi bir* ***w***$\in$ ***L*** *string'i vardır öyleki*

$$w = xyz, \quad y \neq e, \quad |xy| \leq n$$

olmak üzere *yeniden yazılabilir.* ***Her i*** $\geq$ ***0*** *için* $xy^iz \in L$ *olur.*

EXERCISES: PUMPING LEMMA

$L = \{xyx^R : x, y \in \Sigma^*\}$ is regular or not. Why?

$L = \{xyx^R : x, y \in \Sigma^+\}$ is regular or not. Why?

Teorem: ***L*** *regular dil olsun. Dile bağlı olarak seçilen bir*

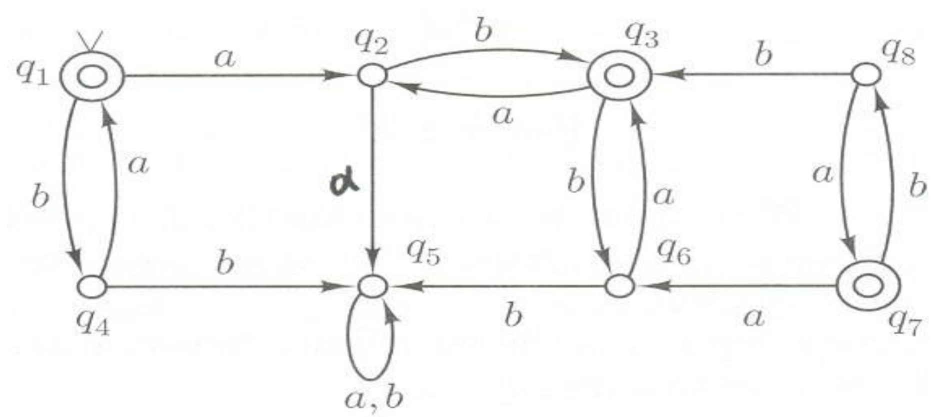
n $\geq$ ***1*** *için* ***|w|*** $\geq$ ***n*** *olacak şekilde herhangi bir* ***w***$\in$ ***L*** *string'i vardır öyleki*

$w = xyz, \; y \neq e, \; |xy| \leq n$

olmak üzere *yeniden yazılabilir.* ***Her i*** $\geq$ ***0*** *için* $xy^iz \in L$ *olur.*

State Minimization

- Bir M otomatı için birçok durumu gözardı etmenin kolay bir yolu olabilir.
- Aşağıdaki otomat $L=(ab\cup ba)^*$ dilini tanır.



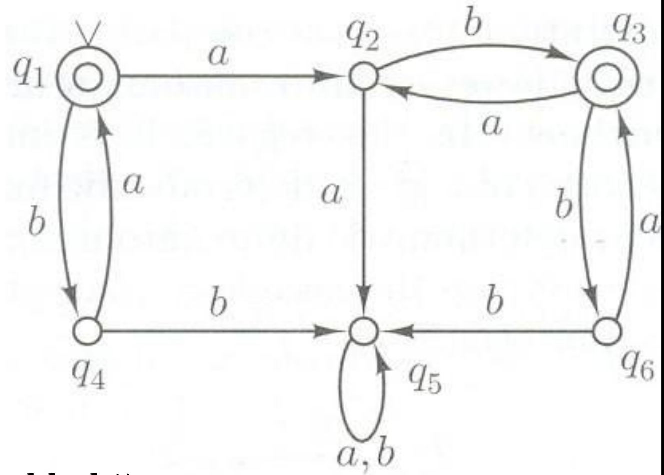
- Burada q_7 ve q_8 durumları unreachable (erişilemez) durumdur.
- Unreachable durumların tamamı otomat'tan doğrudan çıkarılabilir.
- NFA'nın DFA eşitini bulurken aynı optimizasyon yapılmaktadır.

State Minimization

- Unreachable durumları bulan algoritma aşağıdaki gibidir;

R := {s};
*while there is a state **p ∈R and a ∈ L and ∂(p, a)** such that ∂(p, a) ∉ R*
do add ∂(p, a) to R

- Bu algoritmayla silinen durumlardan sonra da otomat hala gereksiz durumlara sahip olabilir.



- Burada q_4 ve q_6 durumları denk'tir (**equivalent**).
- Bu yüzden bir durum olarak birleştirilebilir.
- q_4 ve q_6 denk durumlar aynı string için otomatı sonuç durumuna götürür.
- Denk durumlar aynı string için otomatı sonuç olmayan farklı diğer durumlara da götürebilir.

State Minimization

Tanım: L diline göre iki string'in denkliği

$L \subseteq L^*$ *ve* $x, y \in L^*$ *olsun. Eğer* $z \in L^*$ *ve* $xz \in L$ *iken* $yz \in L$ *olursa* x *ve* y, ***L diline göre denk***

olarak adlandırılır, $\boldsymbol{x \approx_L y}$ *şeklinde gösterilir.*

- *x ve y string'lerinin ikisi de L diline ait olabilir veya olmayabilir.*
- *Sonlarına eklenen herhangi bir string x ve y'i L diline ait yapabilir veya yapmayabilir.*

State Minimization

Örnek: *x bir string ve L bir dil ise,* [*x*] bu *L diline göre x'in sahip oldugu denk sınıfı göstersin.*

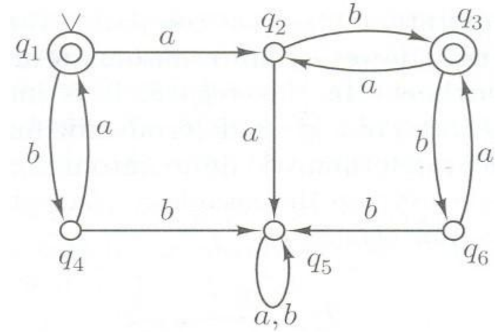
Aşağıdaki otomat $L = (ab \cup ba)^*$ *dilini tanır ve* $\approx_L$ *için 4 denk sınıf vardır;*

1. [e] = L

2.[a] = La

3.[b] = Lb

4.[aa] = L(aa ∪ bb)I*



- ***1. durumda*** *herhangi bir x∈ L için, x = e bile olsa xz∈ L yapan her z string'ide L diline aittir.*
- ***2. durumda*** *herhangi bir x∈ La string'i xz ∈ L olabilmesi için z string'inin* ***bL*** *şeklinde olması gereklidir.*
- ***3. durumda*** *z string'inin* ***bL*** *şeklinde olması gereklidir.*
- ***4. durumda*** *hiçbir z string'i L(aa ∪ bb) prefix'i için bu string'i dile ait yapamaz .*
- ***1. durumdaki kümeye ait tüm stringler aynı durumlara, ve diger 2., 3., veya 4. durumlardaki string kümeleri de aynı durumlara götürür.***

State Minimization

Tanım: M otomatına göre iki string'in denkliği

M = (K, L, δ, s, F) *bir deterministic automata olsun.*

Eger iki string $x, y \in \sum^*$ *M otomatını s başlangıç durumundan aynı duruma götürüyorsa* ***M otomatına göre denktir*** *ve* $\boldsymbol{x \sim_M y}$ *şeklinde gösterilir.*

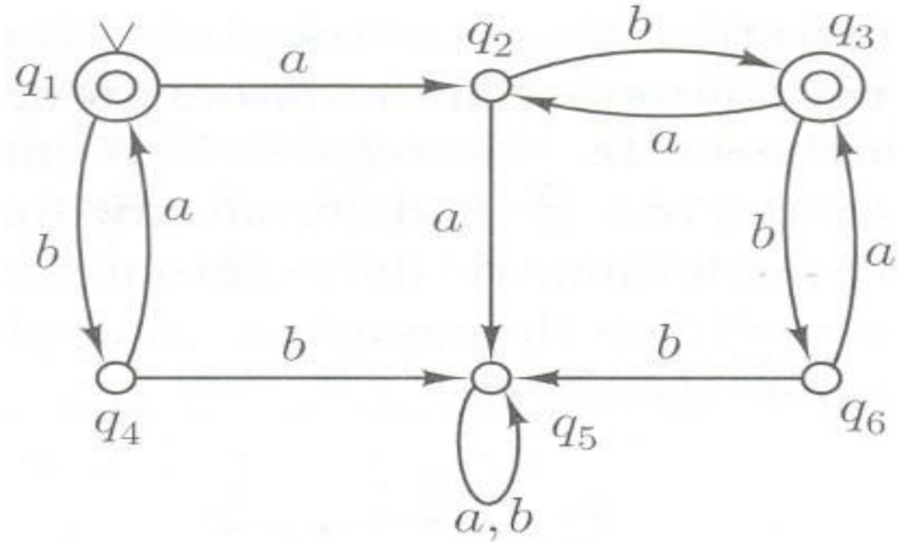
$$(s, x) \vdash^* (q, e) \quad ve \quad (s, y) \vdash^* (q, e) \quad ise\ x \sim_M y \quad olur.$$

- *Herhangi bir q durumu için denk sınıf* E_q *şeklinde gösterilir.*
- *Bu durumların s başlangıç durumundan erişilebilir olması zorunludur.*

State Minimization

Örnek: *Aşagıdaki otomat* $L = (ab \cup ba)^*$ *dilini tanır ve* $\sim_M$ *için 6* ***denk sınıf*** *vardır;*

1. $E_{q1} = (ba)^*$
2. $E_{q2} = La \cup a$
3. $E_{q3} = abL$
4. $E_{q4} = b(ab)^*$
5. $E_{q5} = L(bb \cup aa) \sum *$
6. $E_{q6} = abLb$



Teorem: *Deterministic* $M = (K, \text{L}, \delta, s, F)$ *otomatında herhangi* $x, y \in \sum^*$ *string'leri için* $\boldsymbol{x} \sim_M \boldsymbol{y}$ *ise* $\boldsymbol{x} \approx_{L(M)} \boldsymbol{y}$ *olur.*

Myhill-Nerode Theorem

Teorem: ***L*** $\subseteq \sum^*$ *regular dil olsun.* ***L dilini tanıyan ve*** $\approx_L$ ***içindeki denk sınıfların sayısına tam eşit sayıda duruma sahip olan*** *bir deterministic automata vardır.*

ispat: $x \in \sum^*$ *string'i için* $\approx_L$ *ilişkisi içinde denk sınıflar* $[x]$ *şeklinde gösterilir. Bu dil için* $M = (K, \sum, \delta, s, F)$ *otomatı aşağıdaki gibi oluşturulabilir;*

$K = \{ [x] : x \in \sum^* \}$, $\approx_L$ *altında denk sınıflar kümesi*

$s = [e]$, $\approx_L$ *altında e için denk sınıflar kümesi*

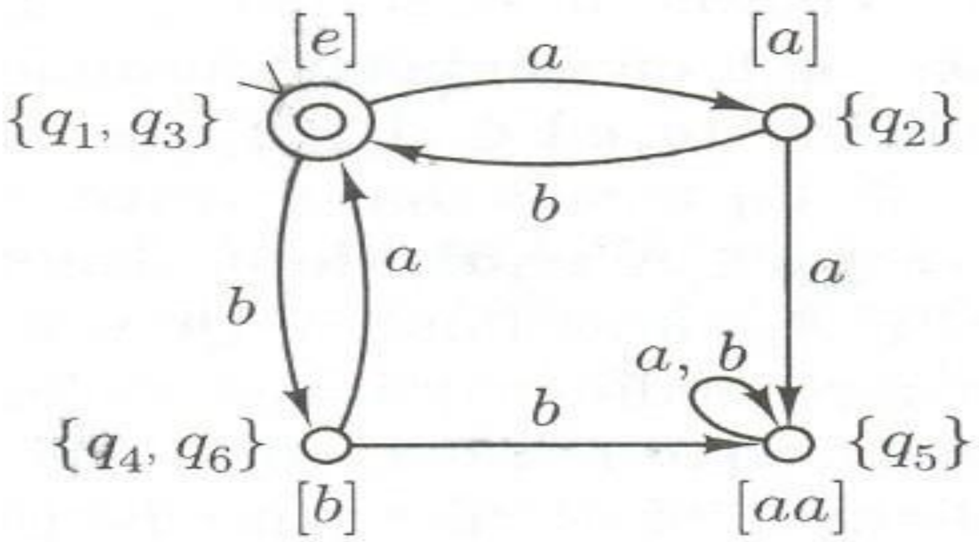
$F = \{ [x] : x \in L \}$,

Son olarak herhangi bir $[x] \in K$ *ve herhangi bir* $a \in \sum$ *için,*

$\delta([x], a) = [xa]$ *geçişleri tanımlanır.*

Myhill-Nerode Theorem

Örnek: *$L = (ab \cup ba)^*$ dilini tanıyan 6 duruma sahip olan deterministic otomat 4 durumla gösterilebilir.*

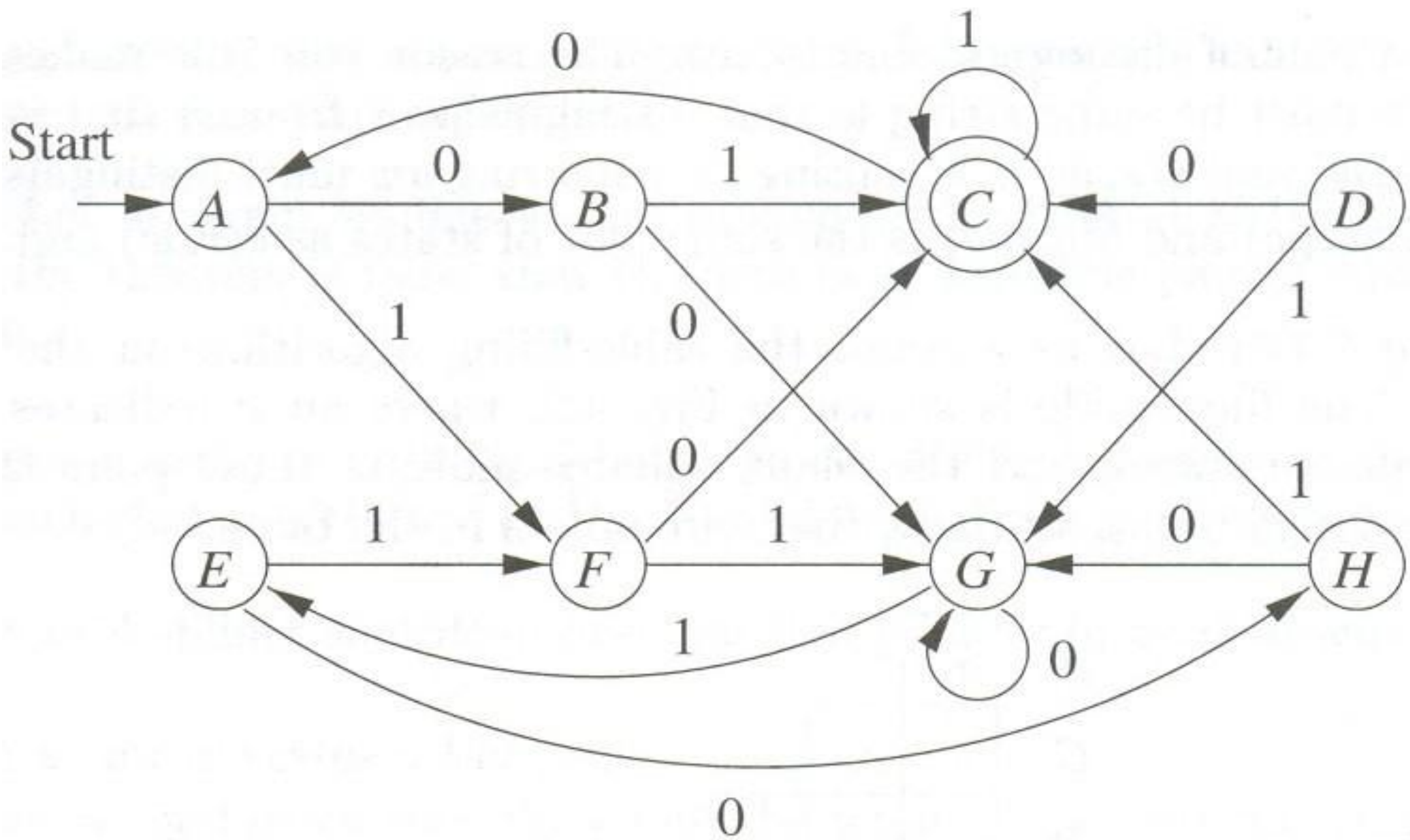


Corollary: *L bir regular dildir, eger $\approx_L$ sonlu sayıda denk sınıfa sahipse.*

i*spat :* *$L = L(M)$ regular ise, en az $\approx_L$ deki denk sınıfların sayısı kadar duruma sahip bir M deterministic sonlu otomat vardır.*

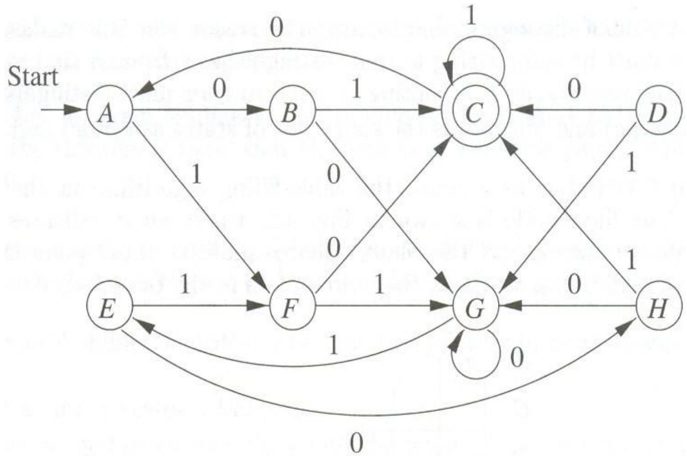
State Minimization

Örnek: *Aşagıdaki otomat'ın en az duruma sahip eşitini bulalım. Denk durum var mıdır ?*



State Minimization

Örnek: *Aşagıdaki otomat'ın en az duruma sahip eşitini bulalım.*



denk durum var mıdır ? C ve G denk degildir!

A ve G denk midir?

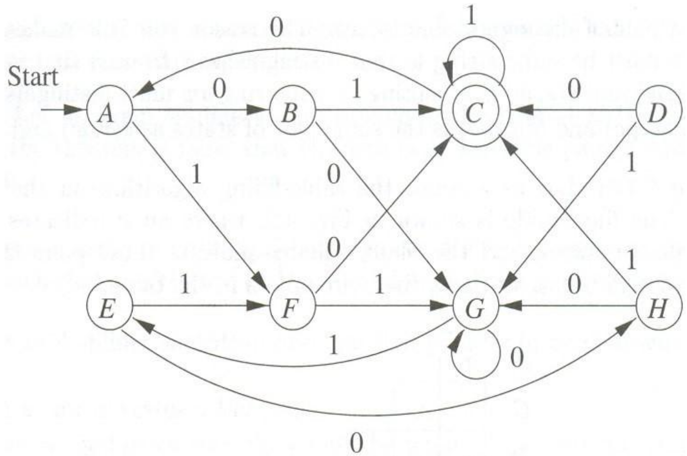
- e-string için ikisi denktir.

Çünkü ikisi de final state degildir.

- *0 için ikisi B ve G ye gider. İkisi de final state değildir ve denktir.*
- *1 için F ve E ye giderler. İkisi de final state değildir ve denktir.*

State Minimization

Örnek: *Aşagıdaki otomat'ın en az duruma sahip eşitini bulalım.*



denk durum var mıdır ? C ve G denk degildir!

A ve G denk midir?

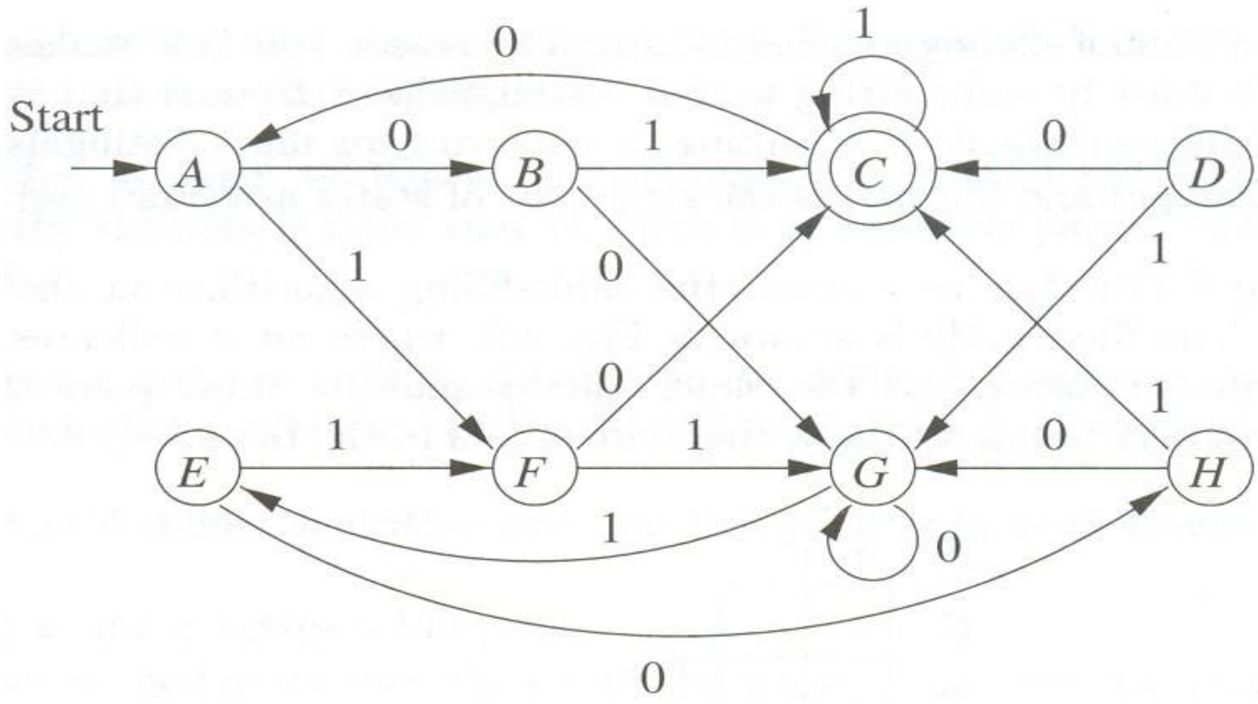
- e-string için ikisi denktir.

Çünkü ikisi de final state degildir.

- *0 için ikisi B ve G ye gider. İkisi de final state değildir ve denktir.*
- *1 için F ve E ye giderler. İkisi de final state değildir ve denktir.*
- *01 için sırasıyla C ve E ye giderler. C final state E değildir. Böylece* ***01 için denk değillerdir****.*
- *Herhangi bir string için seçilen iki durumdan birisi final state'e giderken digeri gitmiyorsa denk olmadıkları ispat edilmiş olur.*

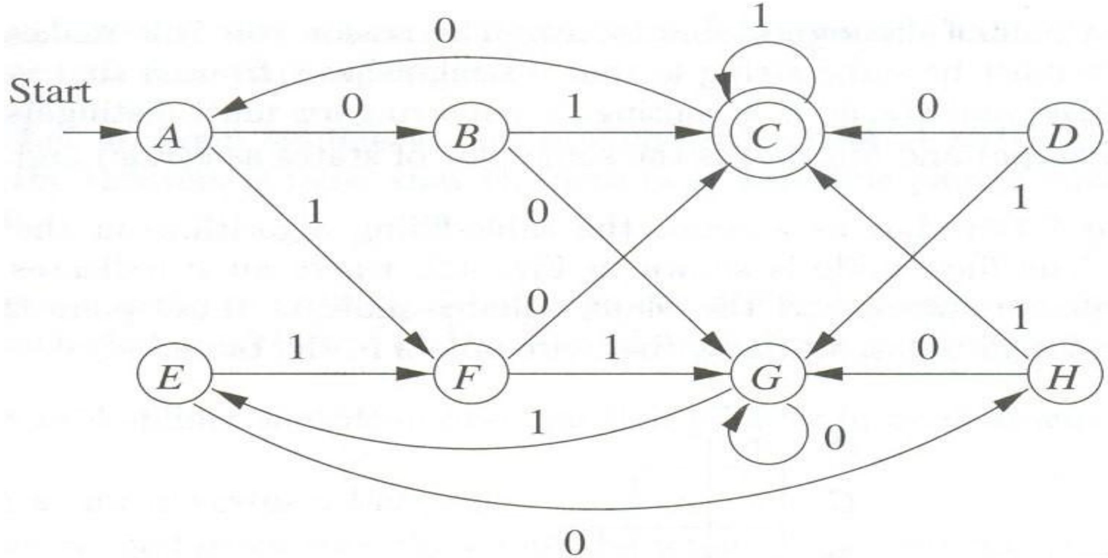
State Minimization

Örnek: *Aşagıdaki otomat'ın en az duruma sahip eşitini bulalım.*



State Minimization

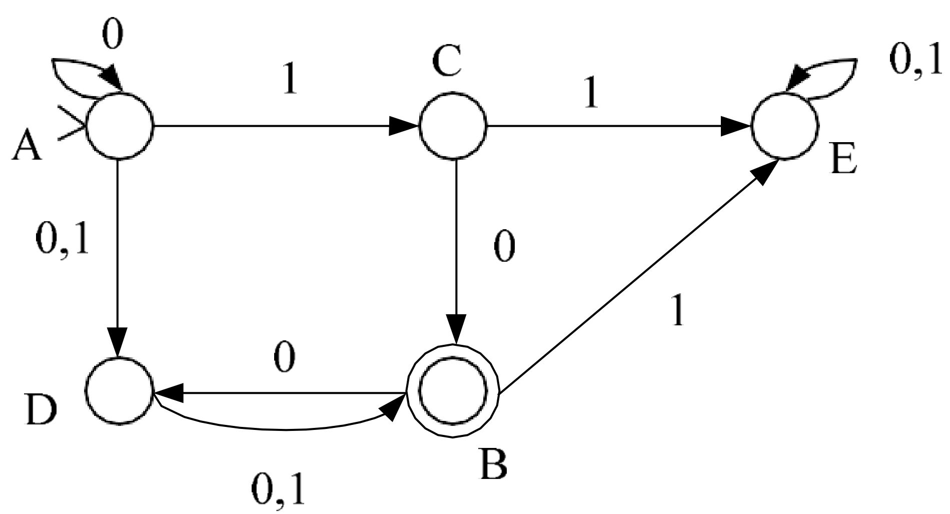
Örnek: *Aşagıdaki otomat'ın en az duruma sahip eşitini bulalım.*



- *A ve E durumlarına bakalım. İkisi de final state olmadıgı için e-string için denktirler.*
- *1 için ikisi de F' ye gider. 1'le başlayan tüm stringler için denktirler.*
- *0 için B ve H' ye giderler. İkisi de final state olmadıgı için denktir.*
- *01 için ikiside C'ye 00 için ikisi de G'ye gider. δ(A, ∑ *)= δ(E, ∑ *)*

State Minimization

- *Denk durumların bulunması için tüm durumları tek tek test etmek gerekir. Bu zor ve zaman alıcıdır. Hata yapma olasılığı yüksektir.*
- *Bunu sistematik yapmak için* ***Tablo Doldurma Algoritması*** *kullanılır.*
- *Durumlar kendi kendisini içermeyecek şekilde bir* ***tablo*** *hazırlanır.*
- *Başlangıçta final state'ler ile diğerleri denk değil olarak işaretlenir.*



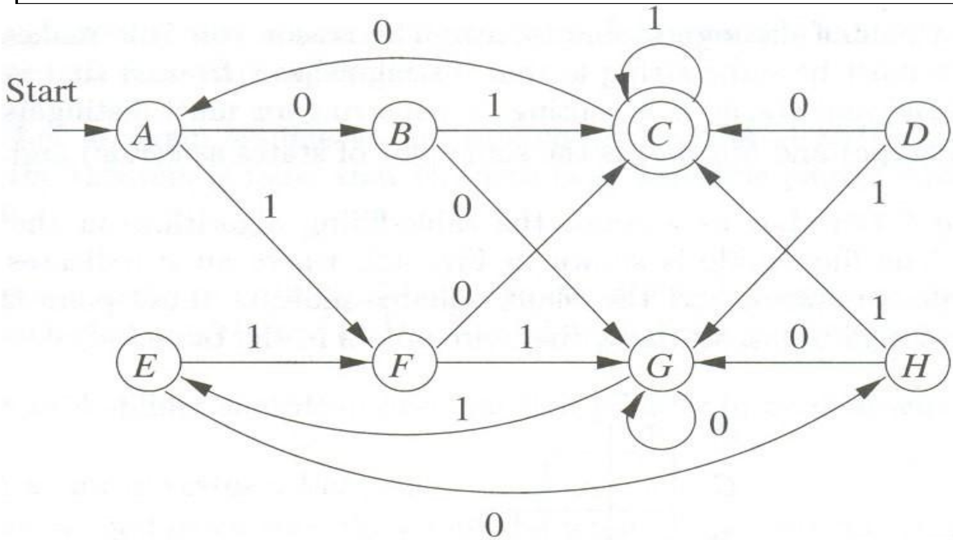
	A	B	C	D
B	x			
C		x		
D		x		
E		x		

State Minimization

Örnek:

- *C final state'tir. C ile diğer durumlar arasına x konur.*
- *Diğer durum çiftlerinde 0 ve/veya 1 girişleri için final state ve diğer durumlara gidiliyorsa x ile işaretlenir.*

  Örn: E ve F 0 için H ve C'ye gider. C final state ve H final state degil E ve F çifti işaretlenir.
- *Örnegin A ve G için 1 girişinde F ve E'ye gider. E-F çifti işaretli oldugu (denk olmadığı) için A-G işaretlenir.*



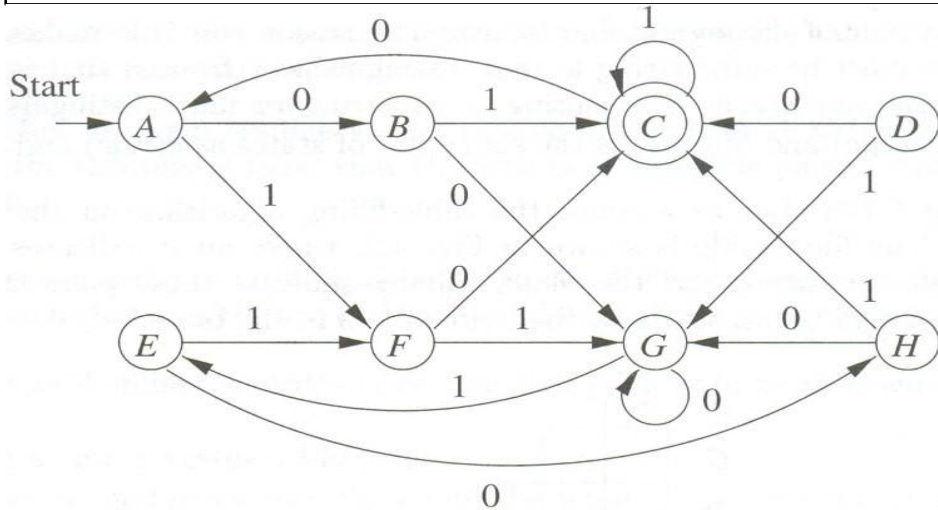
	A	B	C	D	E	F	G
B							
C							
D							
E							
F							
G							
H							

State Minimization

Örnek:

- *C final state'tir. C ile diğer durumlar arasına x konur.*
- *Diğer durum çiftlerinde 0 ve/veya 1 girişleri için final state ve diğer durumlara gidiliyorsa x ile işaretlenir.*

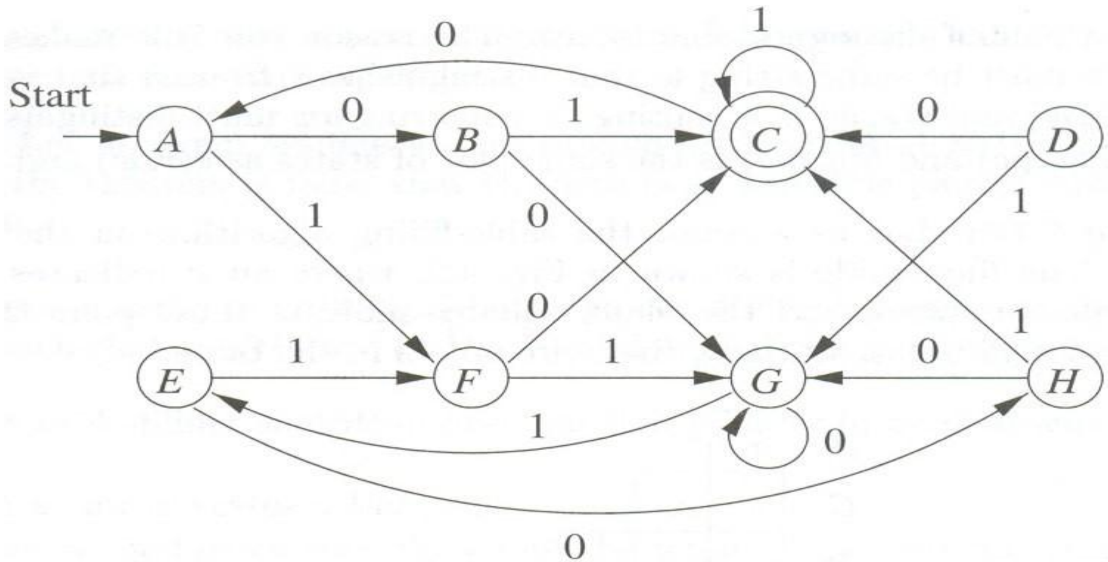
  Örn: E ve F 0 için H ve C'ye gider. C final state ve H final state degil E ve F çifti işaretlenir.
- *Örnegin A ve G için 1 girişinde F ve E'ye gider. E-F çifti işaretli oldugu (denk olmadığı) için A-G işaretlenir.*



B	*x*						
C	*x*	*x*					
D	*x*	*x*	*x*				
E		*x*	*x*	*x*			
F	*x*	*x*	*x*		*x*		
G	*x*	*x*	*x*	*x*	*x*	*x*	
H	*x*		*x*	*x*	*x*	*x*	*x*
	A	*B*	*C*	*D*	*E*	*F*	*G*

State Minimization

Örnek: *Aşagıdaki otomat'ın en az duruma sahip eşitini bulalım.*

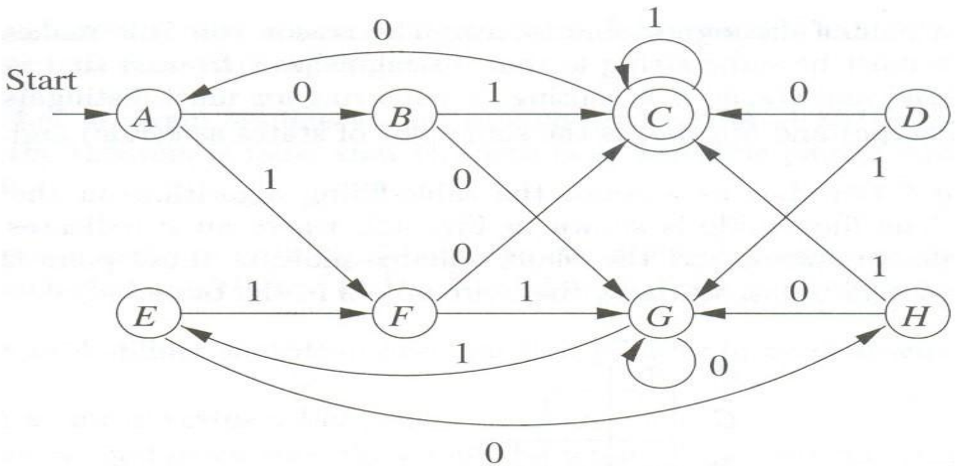


- *A ve E durumlarına bakalım. İkisi de final state olmadıgı için e-string için denktirler.*
- *1 için ikisi de F' ye gider. 1'le başlayan tüm stringler için denktirler.*
- *0 için B ve H' ye giderler. İkisi de final state olmadıgı için denktir.*
- *01 için ikiside C'ye 00 için ikisi de G'ye gider. δ(A, ∑ *)= δ(E, ∑ *)*

State Minimization

Örnek: (devam)

- *A-E, B-H ve D-F durumları denk durumlardır ve birleştirilebilir.*

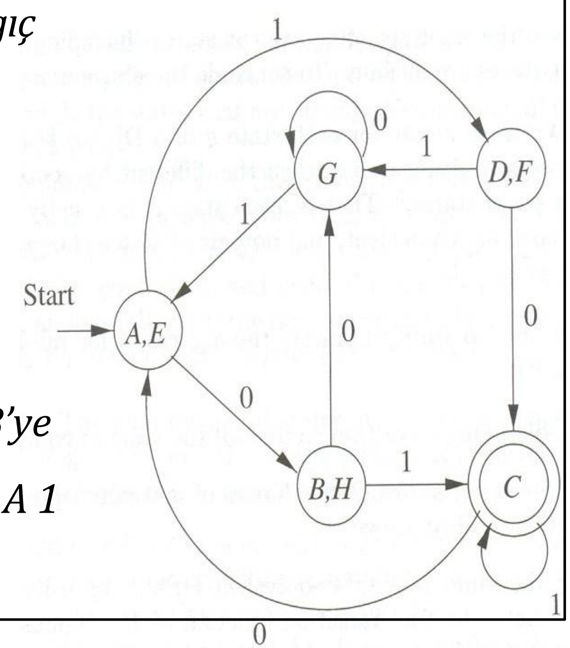


B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G	x	x	x	x	x	x	
H	x		x	x	x	x	x
	A	B	C	D	E	F	G

State Minimization

Örnek: (devam)

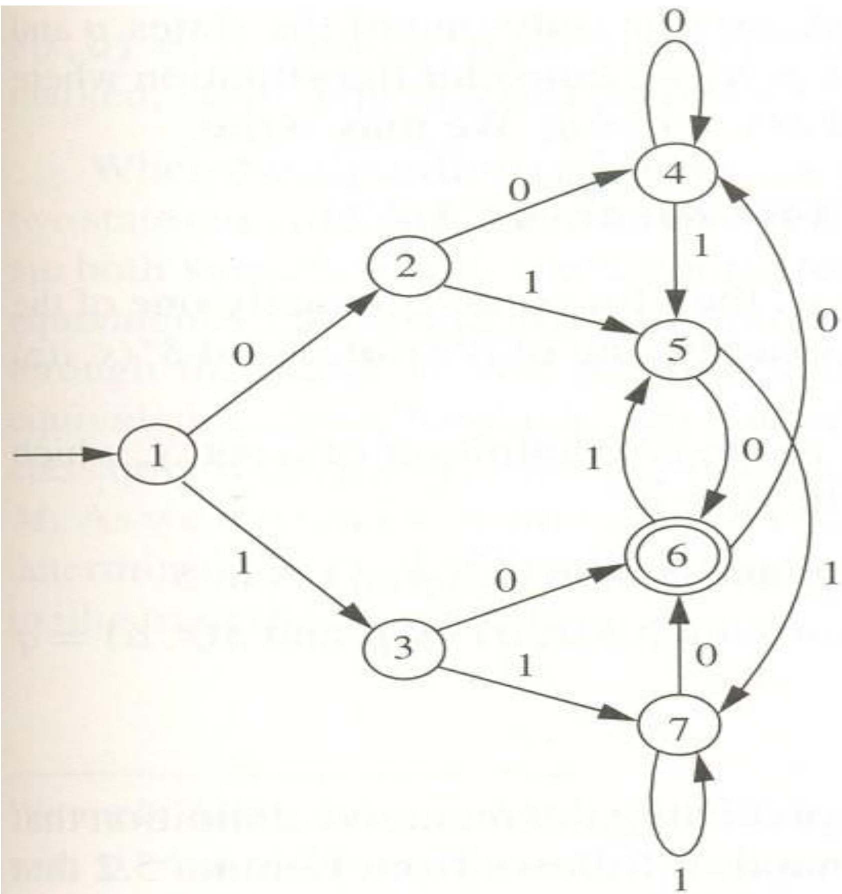
- *States kümesinin partition kümesi, ({A, E}, {B, H}, {C}, {D, F}, {G}) olarak oluşturulur.*
- *Denk durumlar transitive'dir.* ***p-q denk ise ve q-r denk ise p-r 'de denktir****.*
- *Partition kümesindeki her bir eleman bir durum olarak oluşturulur.*
- *Başlangıç durumu {A, E} dir. Çünkü A orijinal otomatta başlangıç durumudur.*
- *C final state'dir. Çünkü orijinal otomatta final state'dir.*
- *Yeni oluşturulan her bir durumun tüm girişler için geçişleri düzenlenir.*
- *Örnek: {A, E} 0 için {B, H}'ye geçer. Orijinal otomatta A 0 için B'ye E ise H'ye geçer. {A, E} 1 için {D, F}'ye geçer. Orijinal otomatta A 1 için F'ye ve E 1 için F'ye geçer.*

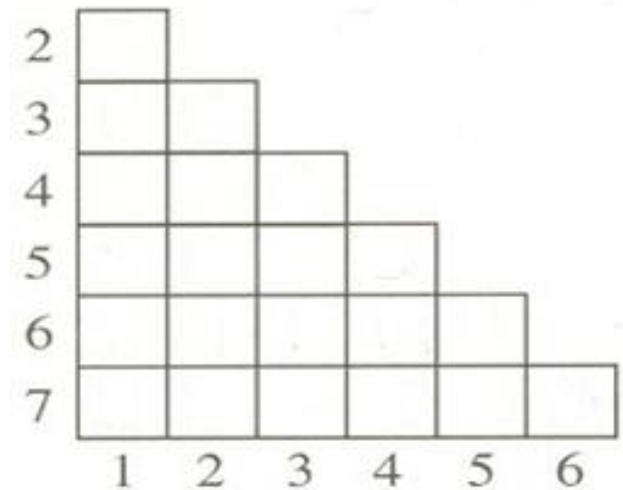


State Minimization

Örnek:

- *Aşağıdaki otomata denk minimum duruma sahip otomatı bulunuz.*

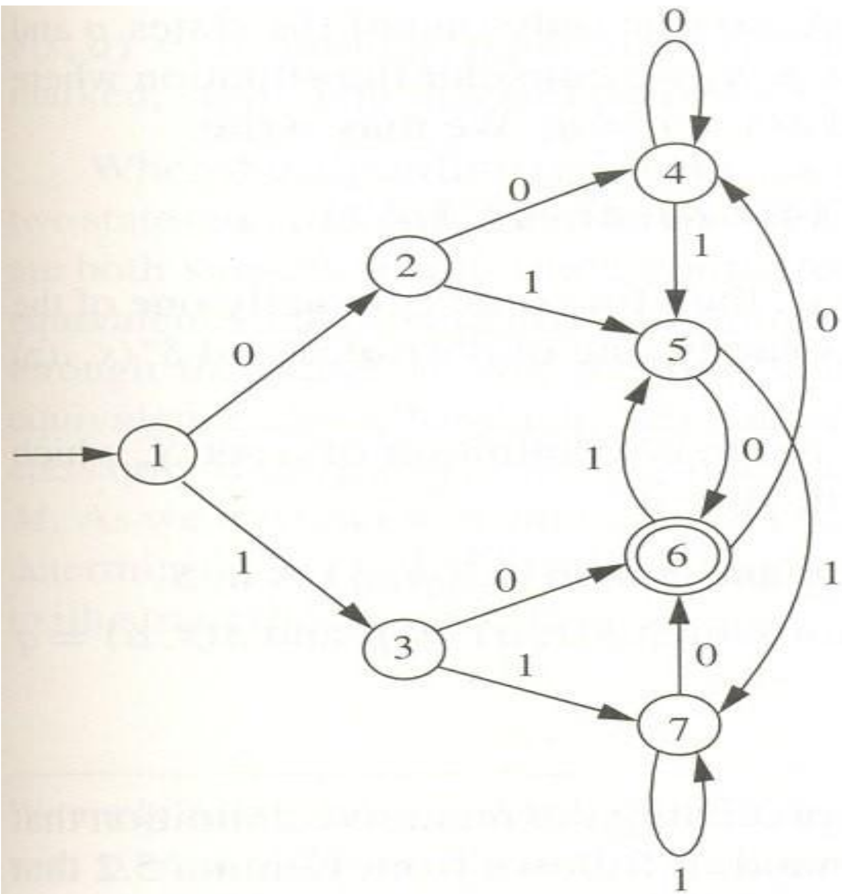




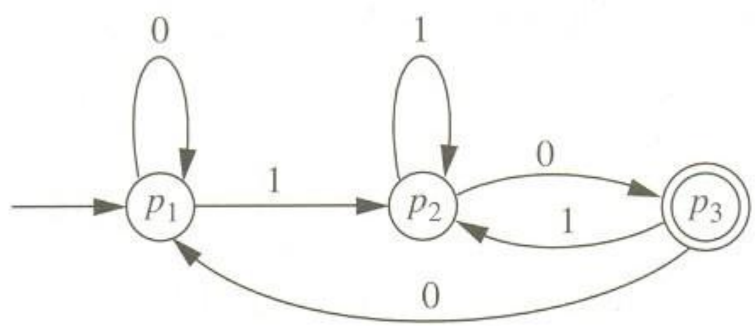
State Minimization

Örnek:

- *Aşağıdaki otomata denk minimum duruma sahip otomatı bulunuz.*



2						
3	2	2				
4			2			
5	2	2		2		
6	1	1	1	1	1	
7	2	2		2		1
	1	2	3	4	5	6



Ödev

- Problemleri çözünüz 2.4.4, 2.4.5 (sayfa 90)
- Problemleri çözünüz 2.5.3 (sayfa 101)