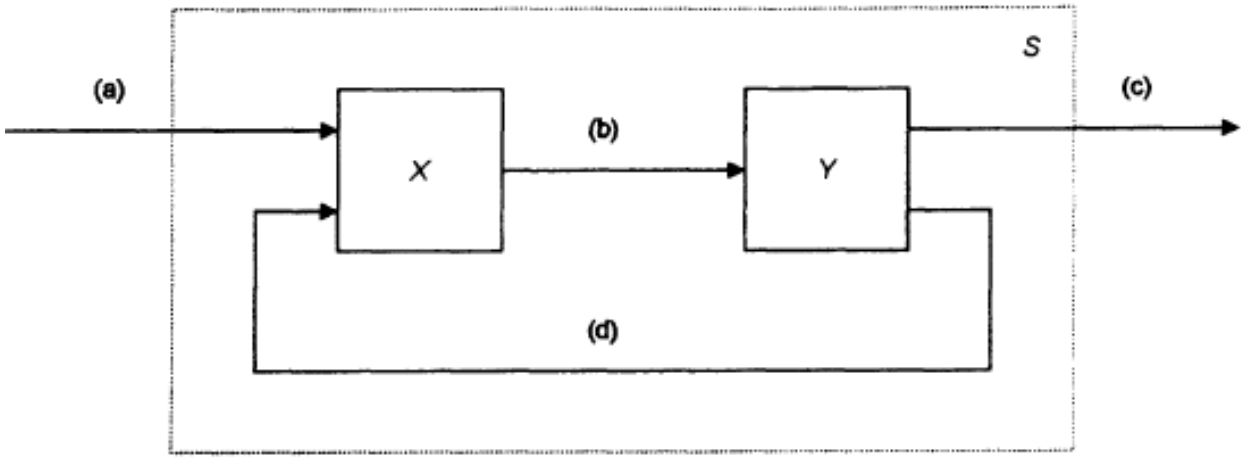
Harici Sinyaller ve Olaylar

İlhan AYDIN

İç ve Dış Sinyaller

- Genelde sistemler iç veya dış sinyaller ile bağlanan modüllerden oluşur.
- Dış sinyaller port olarak isimlendirilen giriş/çıkış'tan gelen veya giden sinyallerdir.
- İç sinyaller ise sadece sistem modüllerini bağlar ve tamamen sistem içinden beslenirler.
- Şekilde X iki giriş ve tek çıkışa sahiptir.
- Şekilde (a) dış sinyal (d) ve (b) iç sinyaldir.
- Diğer taraftan dış sinyaller (a) ve (c) sistemin giriş ve çıkışlardır.



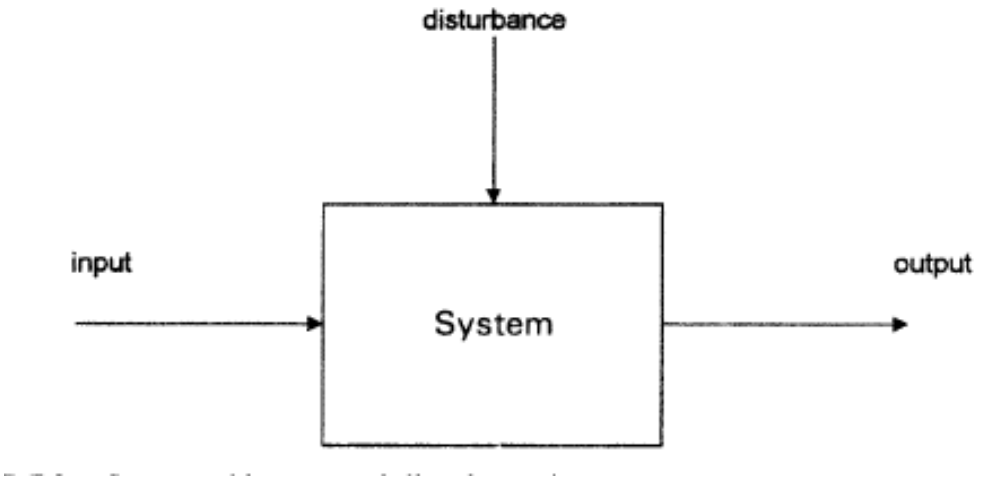
Şekil. X ve Y modülleri ile S sistemi

İç ve Dış Sinyaller

- İç sinyaller ileri beslemeli ve geri beslemeli olabilir.
- İleri beslemeli sinyaller girişten çıkışa ilerler.
- Böylece birden çok kez ziyaret edilen modül yoktur.
- Diğer taraftan geri beslemeli sinyaller döngüler oluşturur ve birden çok kez ziyaret edilir.
- Yukarıdaki Şekilde (b) sinyali ileri beslemeli ve (d) geri beslemelidir.
- İç girişler bazen açık sinyaller ve bozukluklara bölünebilir.
- Açık girişler arzu edilen sinyallerken bozukluklar basitçe ilgilenmesi gereken sıkıntılardır.
- Bir uçağın uçuşunda joystik bir pilotun daha hızlı veya daha yavaş gitmesini kontrol eder.

Bozuk Sinyaller

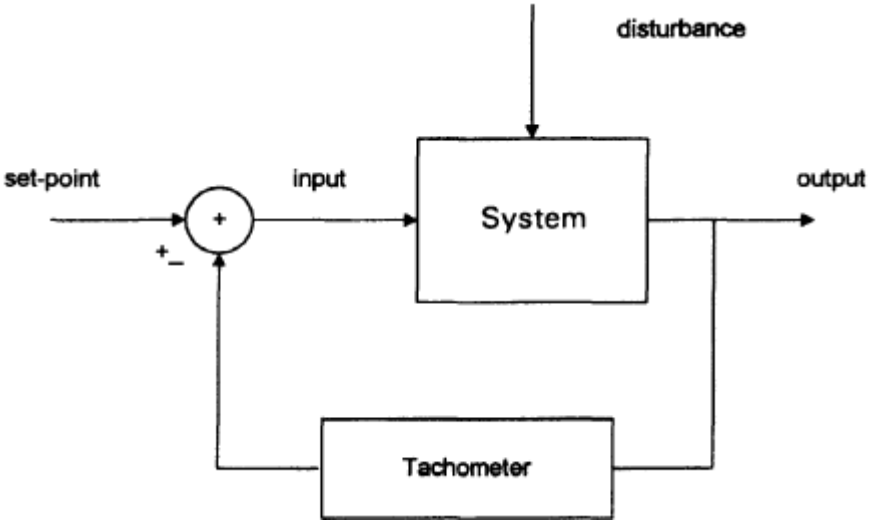
- Eğer durum verilen bir zamanda biliniyorsa model durumun herhangi bir anda durumun ne olduğunu tahmin edebilir.
- Genelde böyle modeller olasılıksaldır. Bunlar spesifik bir örnek için tam olmayan olası yanlış sonuçlar verir. Fakat bütün sistem için mükemmel sonuçlar verir.
- Olasılıklı modeller bir otonom sistem rastegele bir formda bozukluk olarak adlandırılan bir sinyal aldığında oluşur.
- Şekil 2'de bir sisteme hem bozuk hem de normal giriş sinyalleri verilmiştir.



Şekil 2. Bozuk sinyaller

Bozuk Sinyaller

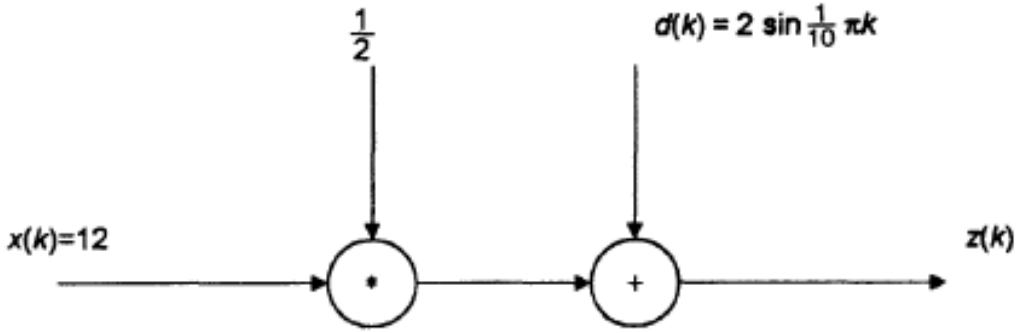
- Sabit yük ile bir motor sürücü sistemi düşünün.
- Sabit bir giriş gerilimi sabit bir hızda motoru döndürmek için uygulanır. Fakat yük birden değişirse sistem aniden yavaşlar. Çünkü giriş sinyali dengeyi sağlayamaz. Bu ekstra yük bozukluk olarak ifade edilir. Mühendisler bu problemi çözmek için geri beslemeli bir model önerirler.
- Çıkış hızı bir takometre ile ölçülür ve gerçek hıza karşılık gelen gerilim geri besleme olarak verilir. Sistem gelen bozukluğa göre kendini ayarlar. Bu durum Şekil 3'te verilmiştir.



Şekil 3. Kapalı devre hareket kontrol sistemi

Örnek:

- Şekil 4'te giriş sinyali *x(k)* = 12, ve transfer fonksiyonu ½ olarak verilmiştir. Sistemin deterministik bozukluk sinyali *d*(*k*) = 2 sin (1/10 π*k*). *Sistemi analiz edin ve bozukluğun etkisini azaltmak için bir geri besleme uygulayın.*



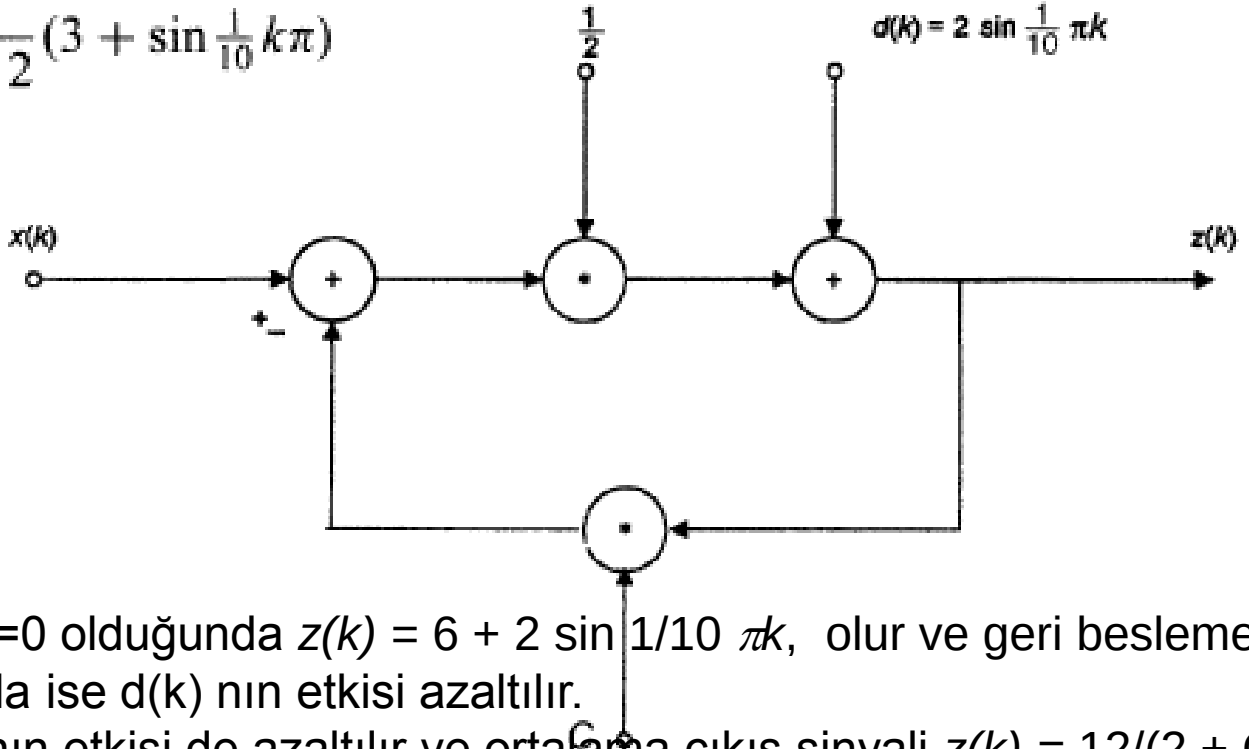
- *Eğer sistemde bozukluk yoksa z(k)= 6'dır. Bozukluk ±2'lik bir değişim uygular. z*(*k*) = 6+ 2sin(1/10 πk) sistemin çıkışı olur.

Örnek:

- Burada C kazancı ile bir geri besleme verilmelidir.
- Temel fikir d(k)'yı azaltmak için C yi bulmaktır.

$$z(k) = d(k) + \tfrac{1}{2}[12 - Cz(k)].$$

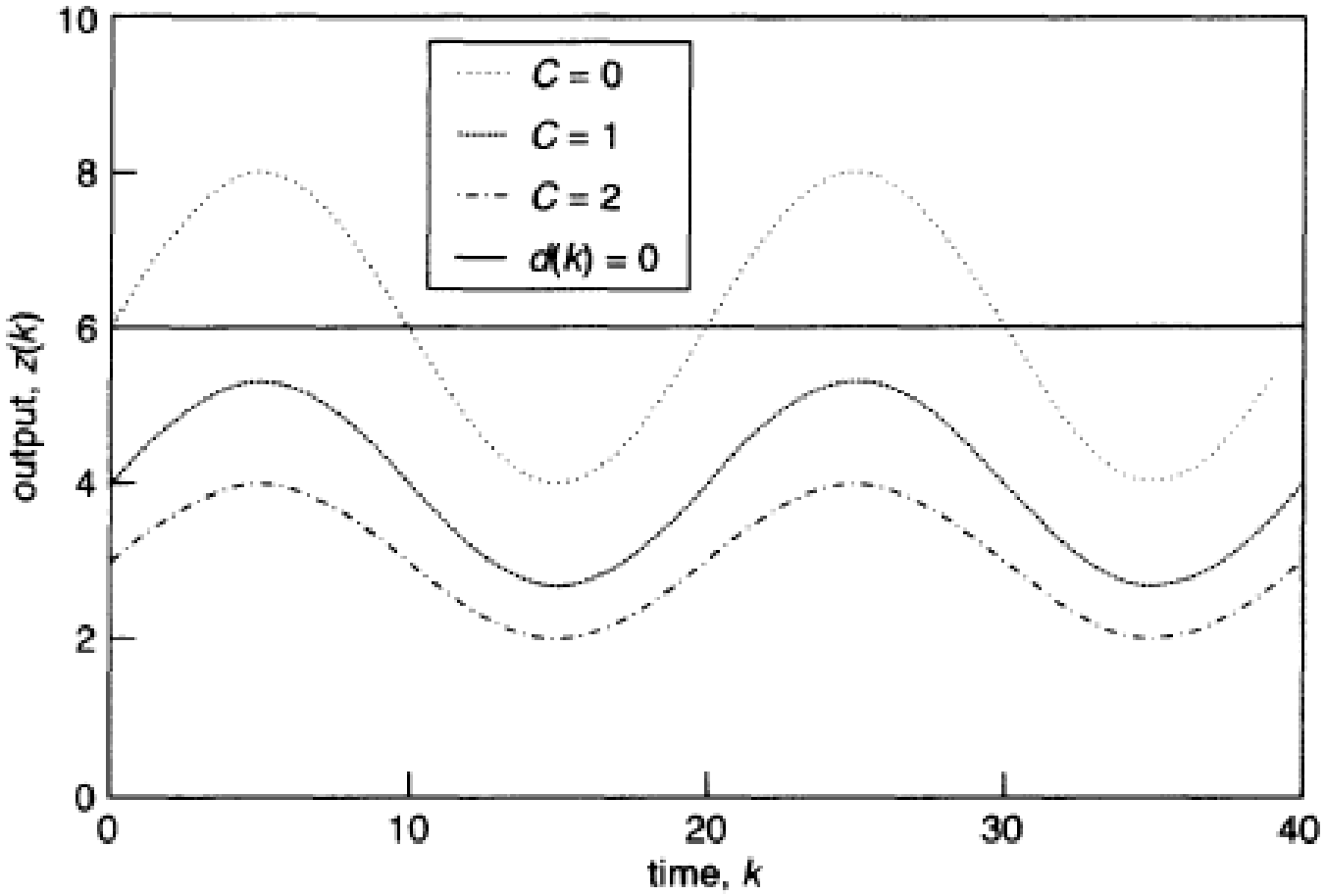
$$z(k) = \frac{4}{C+2}(3 + \sin \tfrac{1}{10} k\pi)$$



Denklemde C=0 olduğunda $z(k) = 6 + 2 \sin 1/10\ \pi k$, olur ve geri besleme olmaz.
C arttırıldığında ise d(k) nın etkisi azaltılır.
Fakat z(k)=6 nın etkisi de azaltılır ve ortalama çıkış sinyali $z(k) = 12/(2 + C)$. Sonuç artık hata

$$e(k) = 6 - \frac{12}{2+C} = \frac{6C}{2+C} \qquad e = \lim_{C\to\infty} 6C/(2+C) = 6$$

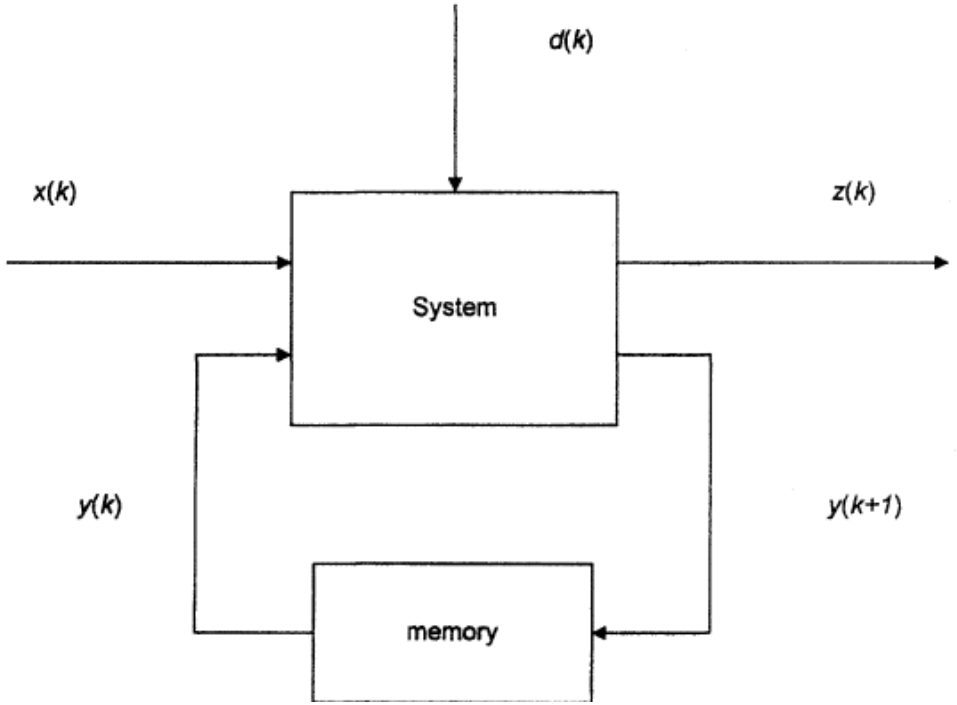
ve C büyük olur.



Şekil. Farklı geri besleme kazançları için sistemin cevabı

Durum Makinaları

- Modeller açık girişin sonlu bir X, bozuk sinyalin sonlu bir D ve durumların sayısı Y olduğu durumda ortaktır.
- Hem dış giriş kümesi hem de durum sayısı sonlu ise Z çıkışı da sonludur.
- Eğer $X = \{xi\}$, $D = \{di\}$, ve $Y = \{yi\}$, *o halde* $Z = \{Xi\} = \{(xi, di, zi)\}$ *X, D,* ve *Y' nin bir kümesidir*.



$y(k+1)=f(x(k), d(k), y(k))$,
$z(k) = g(x(k), d(k), y(k))$.

Şekil. Senkron sonlu durum makinası

Örnek:

- Düzenli bir saatin vuruşlarında A, B, C, D alfabelerinden ardışık olarak çalışan bir sayısal ardışıl sistem düşünelim.
- Bu devre dış d(k) giriş sinyaline göre aşağı veya yukarı birbirini izler.
- Başlangıç durumu $y(0)$ = A:
- Eğer giriş sinyali $d(k) = 1$, ardışıl sistem bulunduğu durumda kalır.
- Eğer giriş sinyali $d(k) = 2$, durum artan sırada listelenir: A, B, C, D, A, B, C, D, ...
- Eğer giriş sinyali $d(k) = 3$, durum azalan sırada listelenir: A, D, C, B, A, D, C, B, .

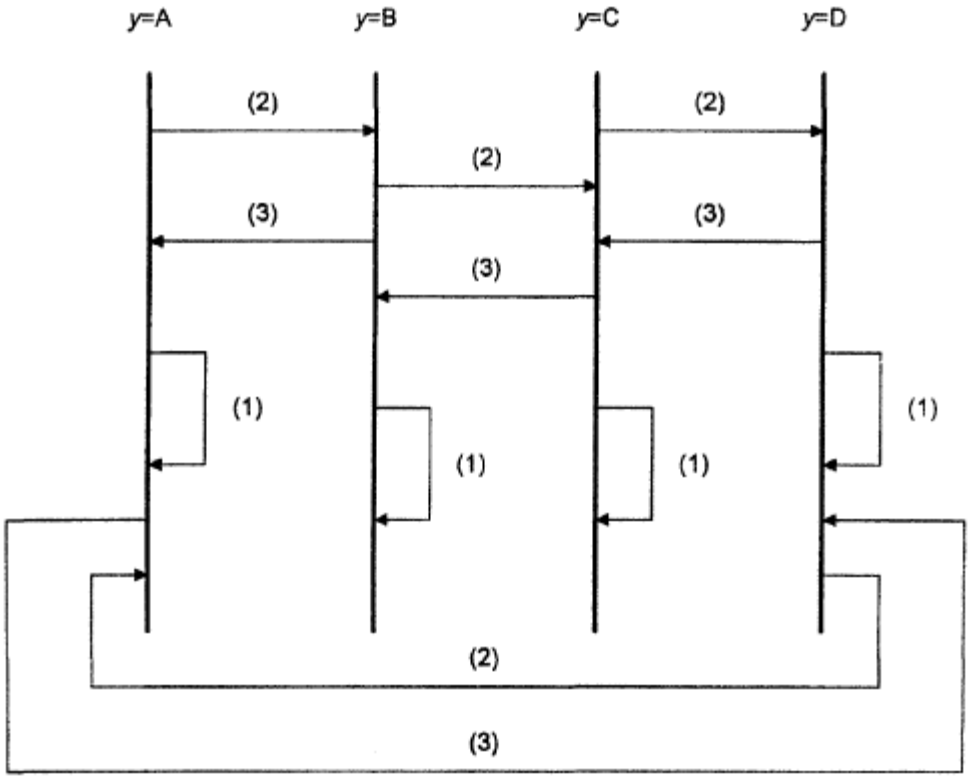
Çözüm

- Bu örnekte durum sadece saat vuruşu ile değişebildiğinden senkron bir sistemdir.
- Aynı zamanda durum ve girişlerin sonlu bir kümesi vardır.

$$X = \emptyset,$$
$$D = \{1, 2, 3\},$$
$$Y = \{A, B, C, D\}$$

Geçiş tablosu

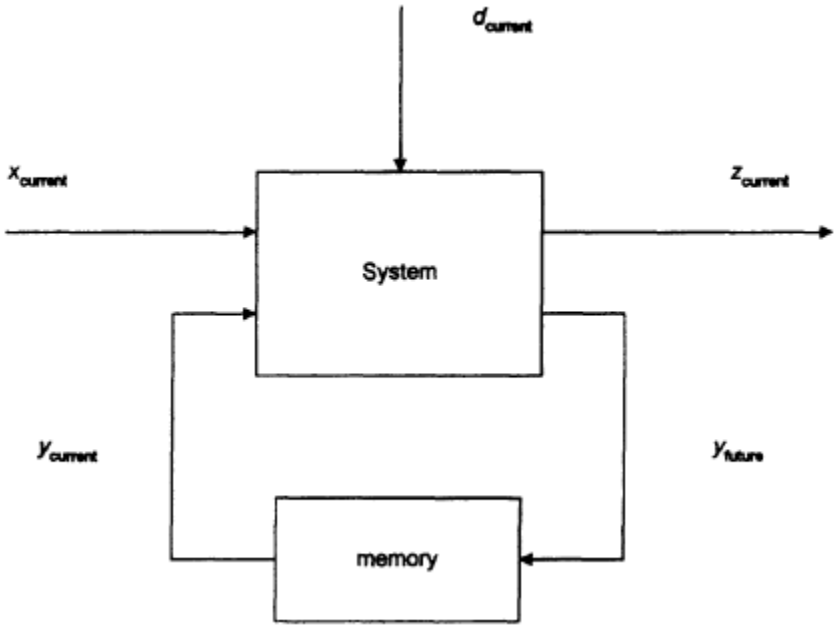
	$y(k) = A$	$y(k) = B$	$y(k) = C$	$y(k) = D$
$d(k) = 1$	A	B	C	D
$d(k) = 2$	B	C	D	A
$d(k) = 3$	D	A	B	C



Durum diyagramı

Durum makinaları

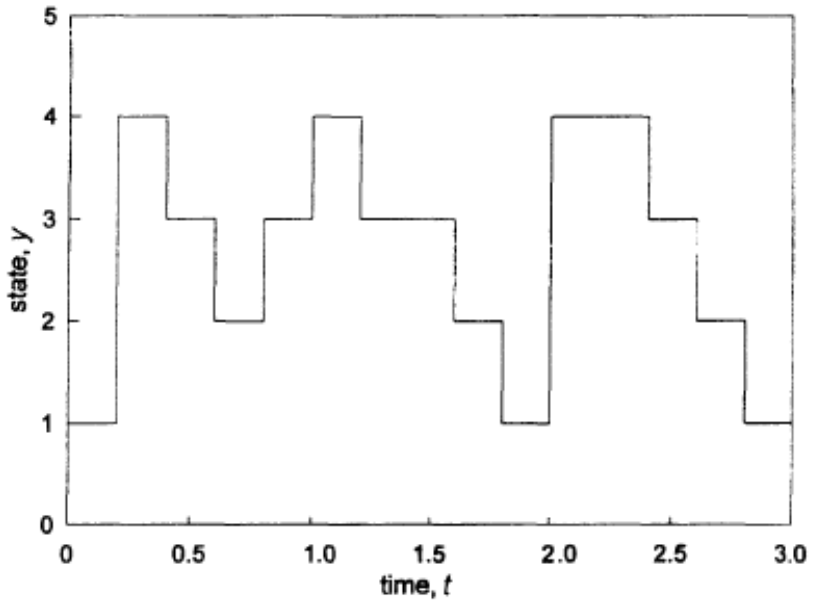
- Eğer olaylar düzensiz zamanlarda oluşuyorsa yani olaylar arası zaman değişken veya rastgele ise sistem asenkrondur.
- Bu durumda sistem blok diyagramı aşağıdaki gibidir.
- Senkron durumda tam bilinmesine rağmen, asenkron durumda önceki durum bilinmez.



Asenkron durum makinası

Durum Makinaları

```
k=0
t=t0
y=y0
print t, y
for k=1 to n
        t=t+δ
        d=INT(1+3*RND)
        call Φ(d, y, z)
        print t, y, z
next k
```




```
subroutine Φ(d, y, z)
      case y
       if y=A
            if d=1 then y1=A ; z1=A
            if d=2 then y1=B ; z1=B
            if d=3 then y1=D ; z1=D
       if y=B
            if d=2 then y1=B ; z1=B
            if d=2 then y1=C ; z1=C
            if d=3 then y1=A ; z1=A
       if y=C
            if d=2 then y1=C ; z1=C
            if d=2 then y1=D ; z1=D
            if d=3 then y1=B ; z1=B
       if y=D
            if d=2 then y1=D ; z1=D
            if d=2 then y1=A ; z1=A
            if d=3 then y1=C ; z1=C
       end case
            y=y1,
            z=z1
return
```

Durum Makinaları

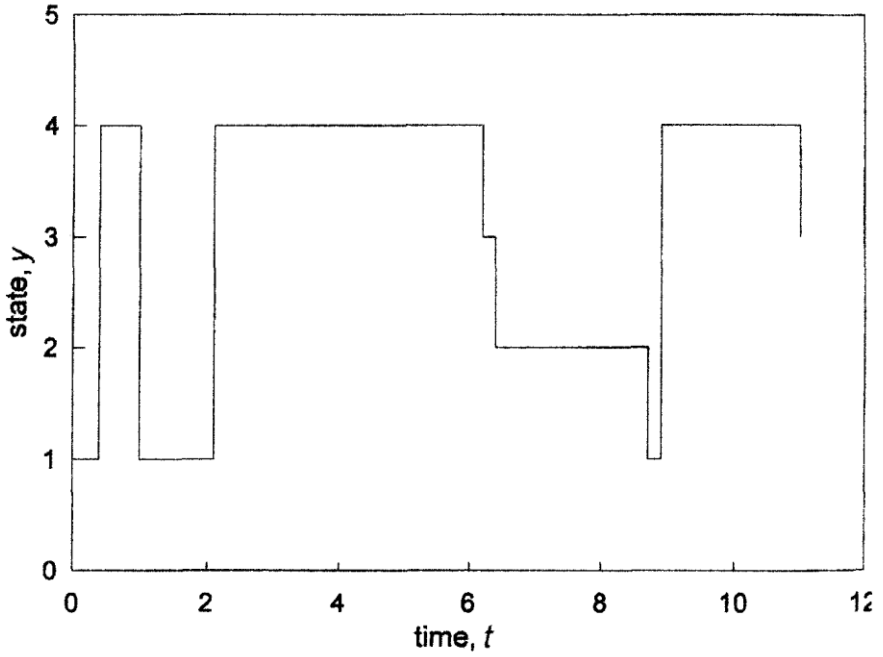
- Önceki slayttaki rastgele durum üretiminde olasılıklar yaklaşık olarak %25'tir.
- Çünkü her defasında rastgele sayı üretilerek gidilecek durum belirlenir.
- Eğer durumlar olasılığa bağlı bu durumda olasılık değerine göre giriş belirlenir.
- Örneğin Pr[D=1]=0.1, Pr[D=2]=0.3 ve Pr[D=3]=0.6 için aşağıdaki kod kullanılabilir.

```
r=10*RND
if r<1 then d=1
if 1 ⩽ r<4 then d=2
if r ⩾ .4 then d=3
```

Durum Makinaları

- Asenkron durum makinası için aşağıdaki kod yazılabilir.
- Burada zaman rastgele olup Poisson dağılımından alınmıştır.

```
      t=t0
      y=y0
      print t, y
[1]   t=t-μ*ln(RND)
      d=INT(1+3*RND)
      call Φ(d, y, z)
      print t, y, z
      if t<tmax then goto [1]
```



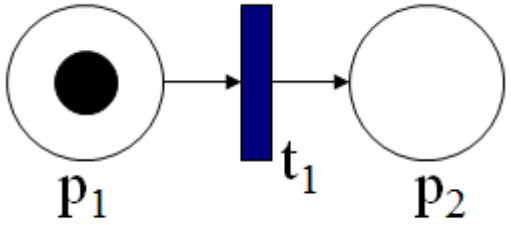
Asenkron durum makinası için sözde kod ve benzetim

Petri Ağları

N tane hücreye sahip bilgisayarda 2^n tane farklı hafıza değeri tanımlanabilmektedir. Çoğu bilgisayar ardışıl işlem yapmaktadır ve bir zamanda bir işlem gerçekleşmektedir. Bu karakteristik (sınırlı hafıza ve ardışıl işlem) geleneksel bilgisayarı sonlu durum makinesi haline getirmiştir. Böyle olması doğaldır çünkü bir anda sadece tek program işletilebilir ve programlar birbiri ardına işletilmektedir. Von neuman mimarisi olarak adlandırılan bu yapı bilgisayarın ilk çıktığı yıllardan beri temel dayanağı olmuştur. Ancak artık çok fazla strateji ve alternatif vardır. Çok işlemcili makineler aynı anda birden fazla işin paralel olarak işlenmesini sağlar. Burada eş zamanlı modellemeye ihtiyaç duyulur. Bu yapılar **petri ağları** olarak adlandırılır.

Temel Petri Ağ Elemanları

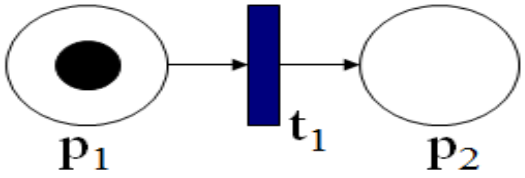
- Petri ağ iki düğümden oluşur: yerler ve geçişler. Arklar yanlızca bir yerden bir geçişe ya da bir geçişten bir yere olabilir.
- Bir yerde sıfır ya da daha fazla işaret bulunabilir.
- Grafiksel olarak,yerler, geçişler, arklar, ve işaretler şu sırada gösterilir: daireler, çizgiler, oklar ve noktalar.



Temel Petri Ağ Elemanları

Aşağıda iki yer ve bir geçişli petri ağ örneği vardır.

Giriş yerlerinin her birinde en az bir işaret varsa, geçiş düğümü iletime hazırdır.

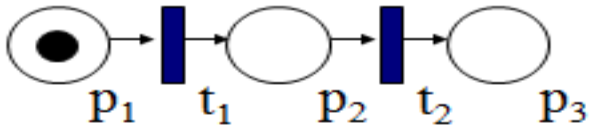


Durum geçişi (1,0)=>(0,1)

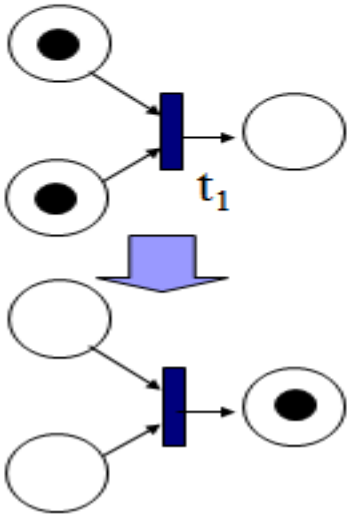
p_1 : giriş yeri p_2:çıkış yeri

Petri Ağ'ların Özellikleri

Sıralı Geçiş: t_1 iletiminden sonra t_2 iletimi gerçekleşir.



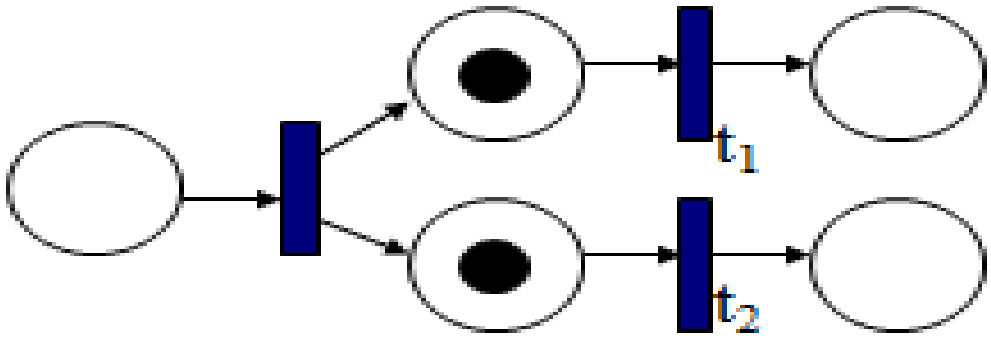
Sekronize Geçiş: Giriş yerlerinin her birinde en az bir işaret olduğunda t_1 etkin olacaktır.



Birleştirme: İşaretler bir çok yerden aynı geçiş hizmetine ulaştığında meydana gelir.

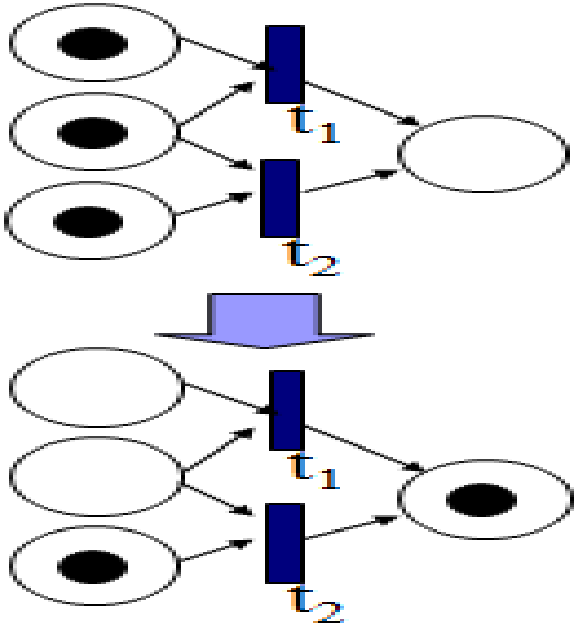
Petri Ağ'ların Özellikleri

Eş Zamanlılık



Petri Ağ'ların Özellikleri

Çakışma



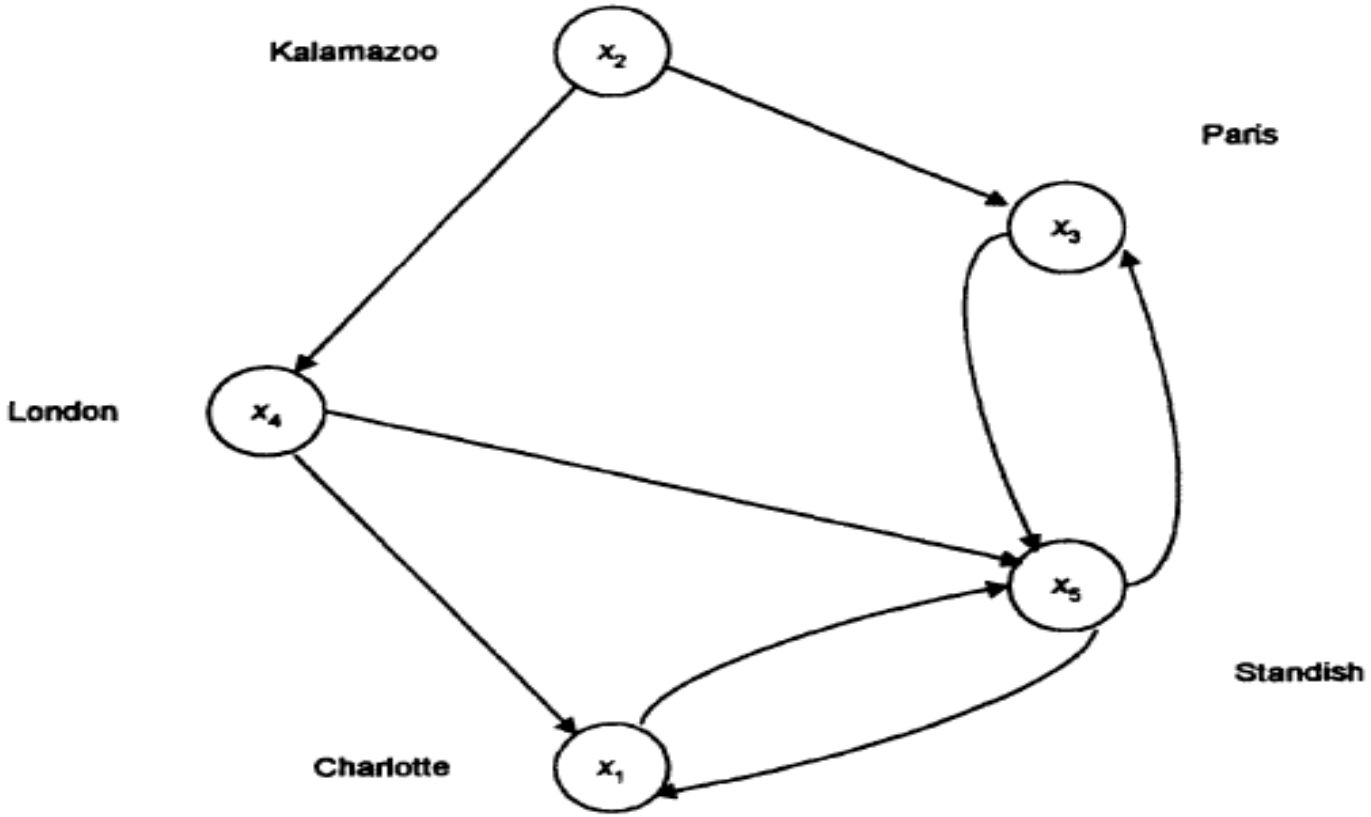
Petri Ağları

Şekil 1deki gibi mesaj merkezli bir ağ düşünün. Bağlantılar düğümlerle beraber bir arkla belirtilmiştir. Şekildeki her merkez i=1,2,3,4,5 için dairesel düğümlerle gösterilmiş ve ve hepsi başka bir şehre gitmek üzere kuyrukta beklemektedir. xi sorudaki belirli bir düğümü temsil ettiğinden yerel durum olarak adlandırılır. Bunun aksine, toplam sistemi karakterize etmek için gerekli olduğundan, sistem durumları (x=x1,x2,x3,x4,x5) yerel durumların vektörü şeklinde gösterilir. (Tablo 1) örneğin x vektörü [2, 0, 1, 3, 0] olduğunda charlott iletmek için 2, Paris 1 Londra 3 mesaja sahiptir. Kalamazo ve standish'in hiç mesajı bulunmamaktadır.

Tablo 1: Mesaj Merkezi Sistem Durumları

Şehir Merkezi	Düğüm	Yerel Durum
Charlotte	1	x_1
Kalamazoo	2	x_2
Paris	3	x_3
London	4	x_4
Standish	5	x_5

Petri Ağları



Şekil 1:Mesaj Merkezlerinin Ağı

Petri Ağları

Mesajların dağılımının süreyi aşmadan gerçekleşmesi tercih edilir. Şekil 1 kullanılarak başlangıç durumu x = [2, 0, 1, 3, 0] olan ve tüm geçişleri listeleyen bir tablo oluşturabiliriz.

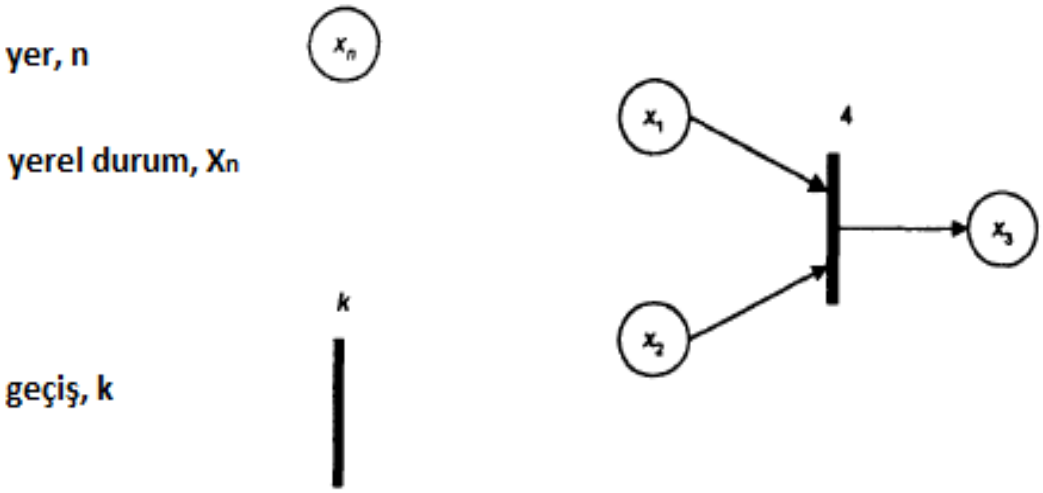
Tablo 2: X=[2,0,1,3,0] başlangıç durumundan bir sonraki durum olasılıkları

Geçiş	X(gelecek)
Charlotte→Standish	[1,0,1,3,1]
Kalamazoo→Paris	mümkün değil
Kalamazoo→London	mümkün değil
Paris→Standish	[2,0,0,3,1]
London→Standish	[2,0,1,2,1]
London→Charlotte	[3,0,1,2,0]
Standish→Charlotte	mümkün değil
Standish→Paris	mümkün değil
Standish→Kalamazoo	mümkün değil

Örneğin şekilden mümkün gibi görünse de Kalamazoo'dan Londra'ya gidiş imkansızdır. Çünkü Kalamazoo'nun bekleyen hiçbir mesajı yoktur. Geriye kalan 4 seçenekten biri gerçekleşecektir ve bu seçim rastgele gerçekleşir. Bu süreç kendini sürekli tekrarlar. Böylece zaman içerisindeki gerçekleşmesi mümkün tüm durumların örneği oluşturulur. Bunu yapmamızın sonucunda biz sadece gidilecek yerleri tespit edebiliyoruz. Ancak gidiş süresiyle ilgili bir bilgimiz olmayacaktır.

Petri Ağları

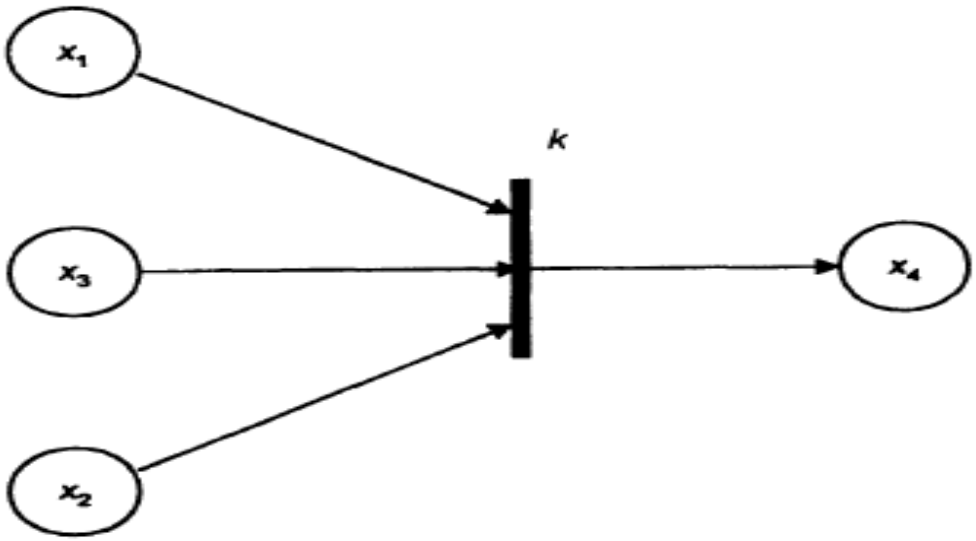
Petri ağlarında gidilecek yerler, mesaj sayıları gibi değerler tanımlanır. Olası geçişlerin kümesine eylem kümesi denir ve hareket için rastgele bir seçim yapılır. Bu tanım ve semboller Şekil 2'de verilmiştir. Sembolik olarak geçişler kalın çizgiyle gösterilir. Her geçişte benzeri olmayan bir numara verilir.



Şekil 2: Standart Petri Ağ Elemanları

Petri Ağları

Eylem kümesinde (S) tüm xi girişleri tanımlıysa geçiş pozitiftir. Bu şekil 3'te verilmiştir.

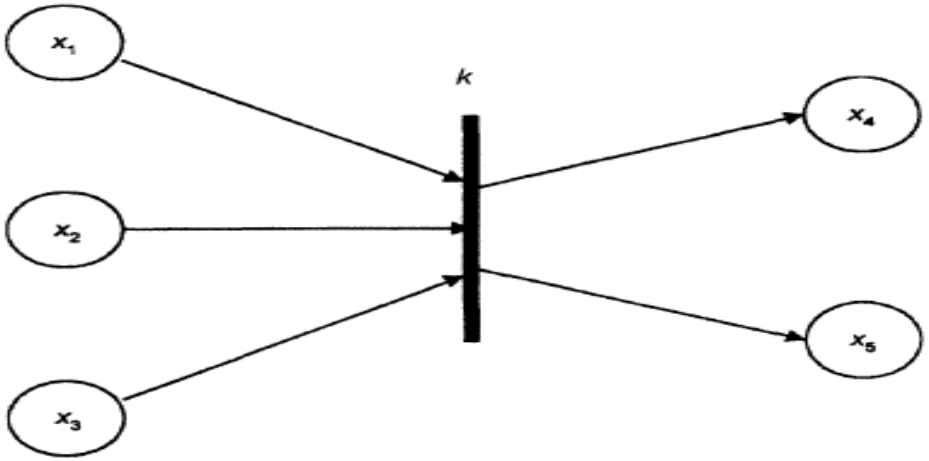


if $x_1>0$, $x_2>0$, $x_3>0$, then $S_{yeni} = S_{eski} \cup \{k\}$

Şekil 3: Eylem kümesinin güncellenmesi

Petri Ağları

Eylem kümesinde tüm kurallara karşılık gelen bir değer bulunmalıdır. S kümesinden rastgele bir eleman seçilir. Bu geçiş k olarak kabul edilir ve her bir durum geçişinde k artırılarak yeni durumlar tespit edilir,(Şekil 4)bu güncellemeyle yeni bir durum oluşur ve bu olay simülasyon bitene kadar devam eder.



k, S'nin bir elemanı ise, $x_1=x_1-1$
$x_2=x_2-1$
$x_3=x_3-1$
$x_4=x_4+1$
$x_5=x_5+1$

Şekil 4: Seçili düğümlerin güncellenmesi

Petri Ağları

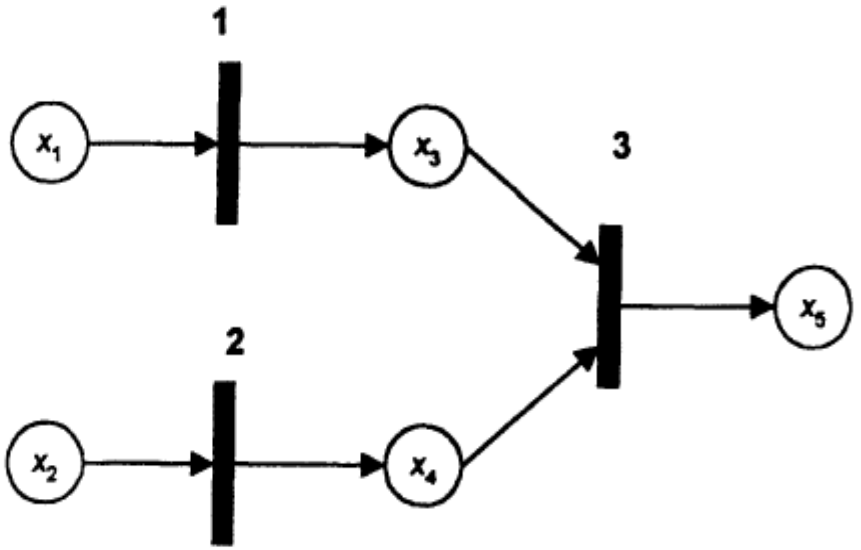
Petri ağları döngü içerisinde 3aşamalı olarak gerçekleşir. Başlangıç şartları girildikten sonra döngü sürekli tekrarlanır. Döngü içerisinde mümkün geçişler eylem kümesine eklenir. Eylem kümesinden rastgele bir hareket seçilir. Ve seçilen harekete göre geçişler güncellenir. Algoritması şu şekildedir;

```
Set initial place, token distribution
k=0
print k, state
for k=1 to n
        Establish all legal transitions; place in action set
        Randomly chose one transition from the action set
        Update only those nodes local to this transition
        print k, state
next k
```

Petri Ağları-Örnek

Şekilde gösterilen bir petri ağı düşünelim; 1. yer için 8 , 2. yer için 5 mesaj tanımlanmış, diğer yerlerin mesajı 0 olarak verilmiştir. Bu modelde petri ağ modeli kullanarak geçici ve uzun vadeli bir davranış simüle ediniz.

$\mathbf{x}(0) = [8,5,0,0,0]$



Petri Ağları-Çözüm

Bir petri ağ simülasyonu için üç aşama vardır. İlk olarak tüm geçişler tespit edilir ve işaretlenir. Bunlardan biri rastgele seçilir ve yürütülür. Verilen örnekte üç geçiş vardır. 1. ve 2. geçiş paraleldir fakat hangisinin önce başlayacağını bilemeyiz ve birini ilk başlaması için seçmeliyiz. x_3 ve x_4 mesajları yerine ulaşıncaya kadar 3 geçişi aktif değildir. 3 geçişi mesajını x_5'e ulaştırır.Ama bunlar gerçekleşene kadar x_1 ve x_2 çoktan gerçekleşmiş olacaktır. Aşağıdaki algoritma yukarıda anlatılan olayı özetlemektedir.

1. tüm yasal geçişlerin belirlenmesi ve işaretlenmesi:

- yasal geçişler bir eylem kümesi (S) oluşturularak yapılır.
- her geçişte yasal olarak bulunan davranışlar bu kümeye eklenir.
- boş bir eylem kümesi yasal hiç bir işlem yapılamayacağını gösterir.

```
S=∅
if x1>0 then S=S∪{1}
if x2>0 then S=S∪{2}
if x3>0 and x4>0 then S∪{3}
```

Petri Ağları-Çözüm

2. yasal geçişlerden birinin rastgele seçilmesi

Burada S boş küme değilse eylem kümesinden rastgele bir eleman seçilir. S boş ise 0 döndürülür.

$$\text{RAND}(\mathbf{S}) = \begin{cases} 0, & \mathbf{S} = \emptyset, \\ y \in \mathbf{S}, & \mathbf{S} \neq \emptyset. \end{cases}$$

S stringinden rastgele bir karakter döndüren algoritma şu şekildedir;

```
function RAND(S)
        r=RND
        for i=1 to LEN(S)
                if r<i/LEN(S) then
                        y=VAL(MID(S,i))
                        goto [1]
                end if
        next i
        y=0
[1]     RAND=y
        return
```

S,bir karakter string'idir.

MID(S,i) fonksiyonu S'den

tek bir karakter döndürür.

VAL, bu değeri sayıya dönüştürür.

LEN, karakter sayısını gösterir.

Böylelikle y=RAND(S) ile rastgele geçiş sağlanır.

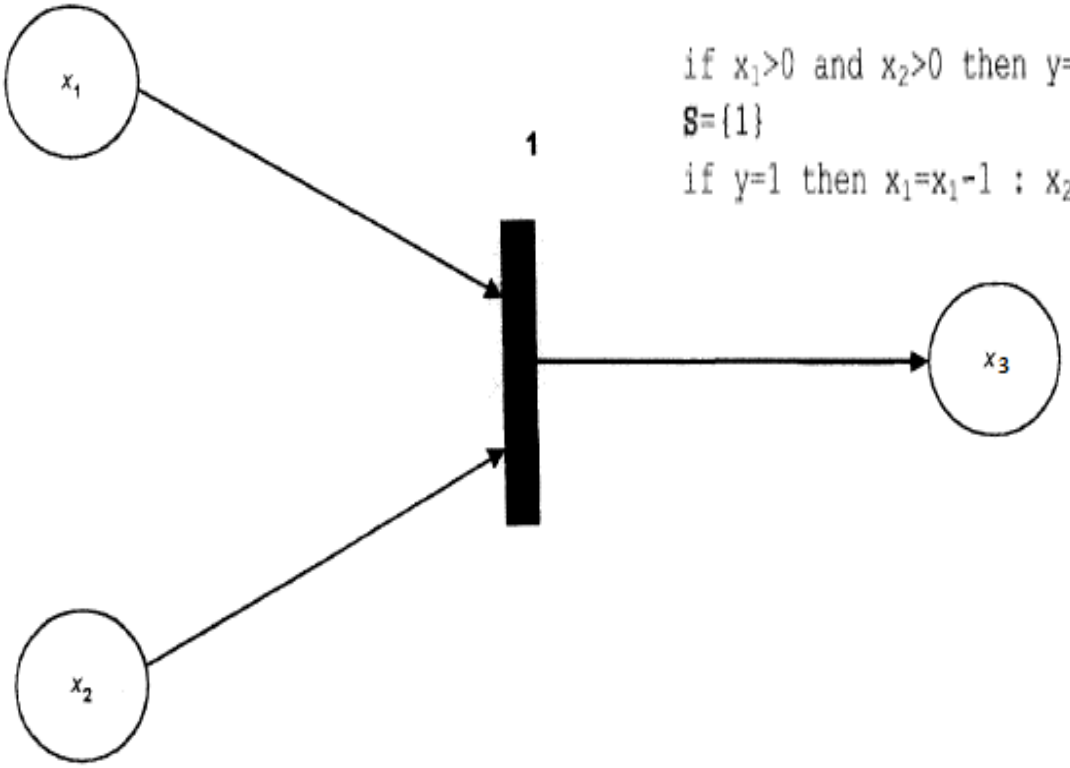
Petri Ağları-Çözüm

```
n=20
k=0
read  x1, x2, x3, x4, x5
print k, x1, x2, x3, x4, x5
for k=1 to n
        S=Ø
        if x1>0 then S=S∪{1}
        if x2>0 then S=S∪{2}
        if x3>0 and x4>0 then S=S∪{3}

        y=RAND(S)

        if y=0 then "deadlock"
        if y=1 then x1=x1-1 : x3=x3+1
        if y=2 then x2=x2-1 : x4=x4+1
        if y=3 then x3=x3-1 : x4=x4-1 : x5=x5+1
        print k, x1, x2, x3, x4, x5
next k
```

k	x_1	x_2	x_3	x_4	x_5
0	8	5	0	0	0
1	8	4	0	1	0
2	7	4	1	1	0
3	7	4	0	0	1
4	7	3	0	1	1
5	7	2	0	2	1
6	7	1	0	3	1

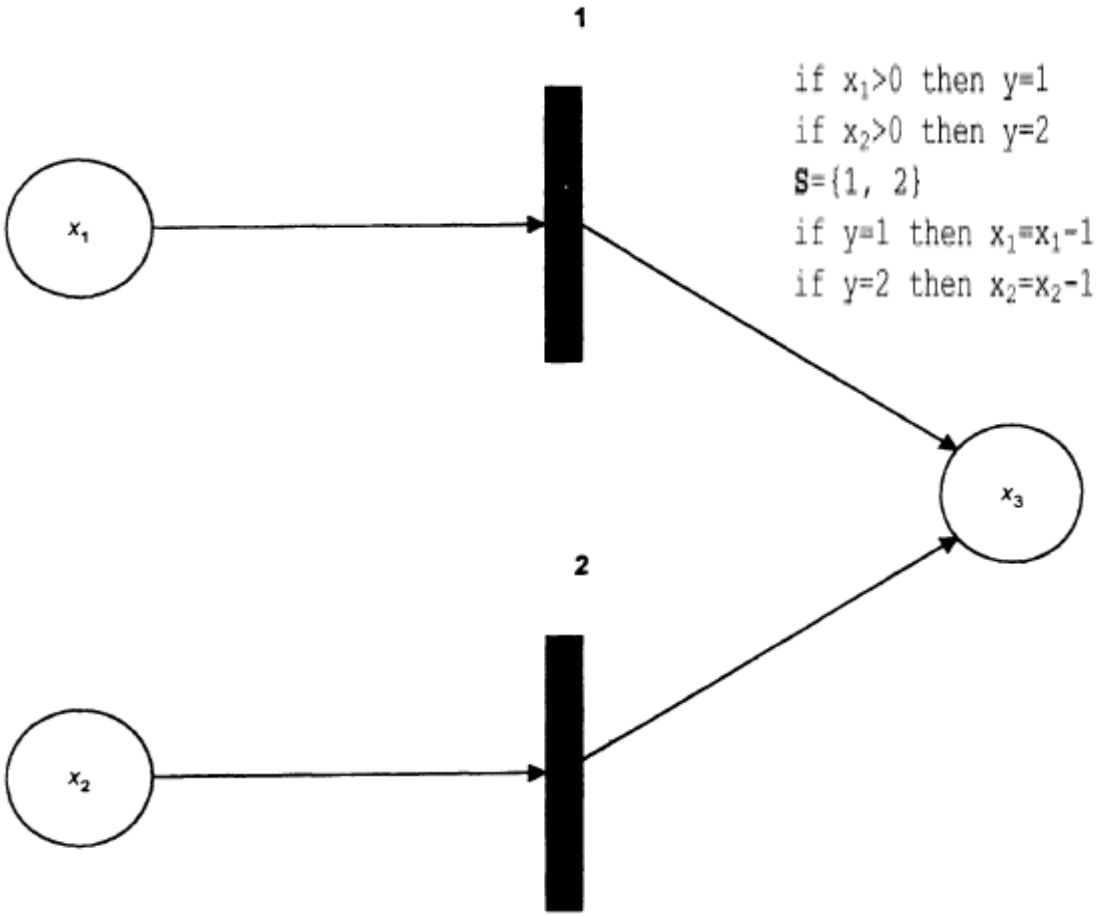
Petri Ağları

Kombinasyonel lojik devrelerdeki and ve or gibi petri ağlarının uygulanmasında kolaylık sağlayan birkaç standart yapı vardır.



Şekil 5: AND kapısı

Petri Ağları

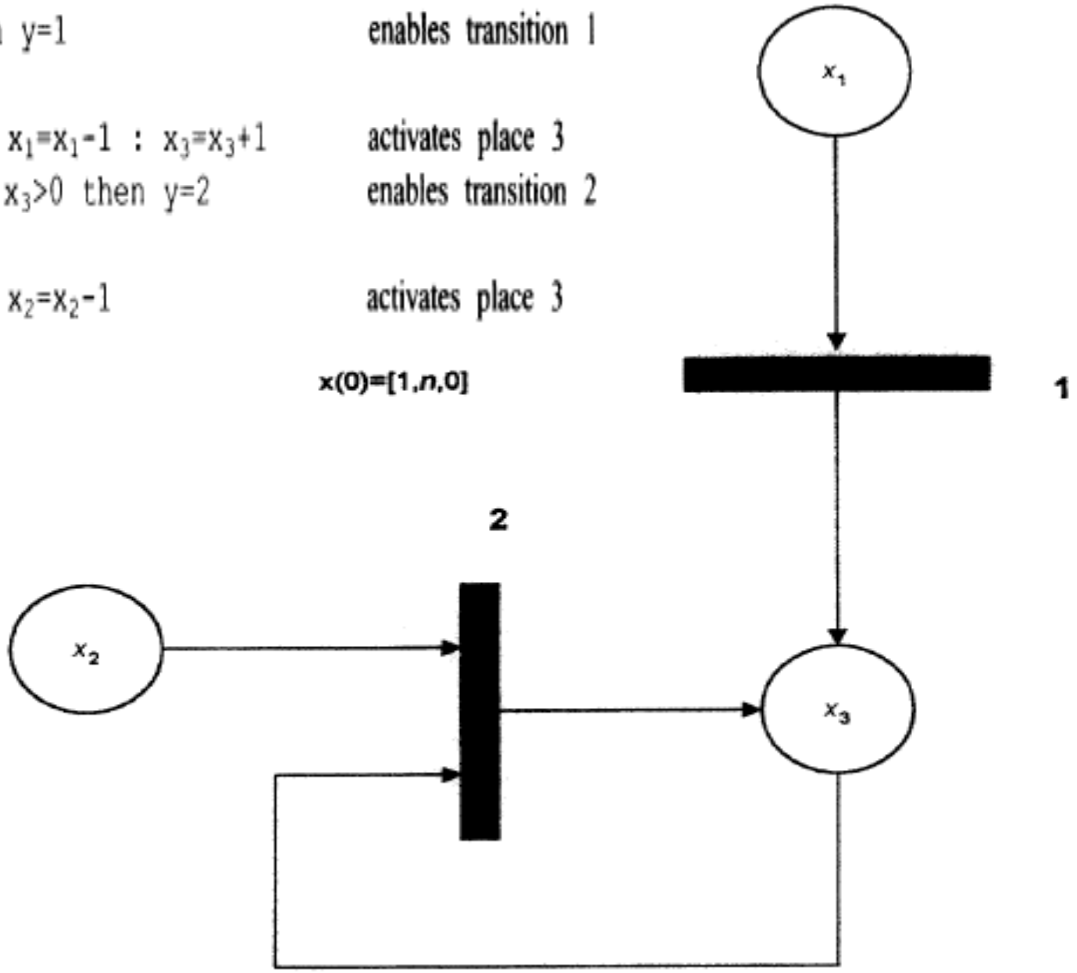


if $x_1>0$ then $y=1$	enables transition 1
if $x_2>0$ then $y=2$	enables transition 2
$S=\{1, 2\}$	activation set
if $y=1$ then $x_1=x_1-1$: $x_3=x_3+1$	activates place 3
if $y=2$ then $x_2=x_2-1$: $x_3=x_3+1$	activates place 3

Şekil 6: OR Kapısı

Petri Ağları

if $x_1>0$ then y=1	enables transition 1
S={1}	
if y=1 then $x_1=x_1-1$: $x_3=x_3+1$	activates place 3
if $x_2>0$ and $x_3>0$ then y=2	enables transition 2
S={2}	
if y=2 then $x_2=x_2-1$	activates place 3



Şekil 7: Hafıza

Örnek:

- Bir otomobil sistemi düşünelim. Bu sistemde araç bir anahtar ile çalıştırılır. Daha sonra boşta çalışma ve hareket etme eylemleri gerçekleştirilir. Bu diziyi basit Petri ağı kullanarak modelleyiniz

Çözüm

- Yerler ve geçişler için bazı tanımlamalar yaparak başlayacağız. Yerler sabit durumlarken geçişler bir durumdan diğerine geçişi sağlar. Liste aşağıdaki gibidir.
- **Yerler (Lokal durumlar):**

1: Anahtar ateşleme pozisyonunda, 2: Tanktaki benzin, 3: Akümülatörün şarjı,

4: Benzin karbiratörde, 5: Dağıtıcı güç alması, 6: Motor çalışması

7: Arabanın hareketi, 8: Arabanın çalıştırılması

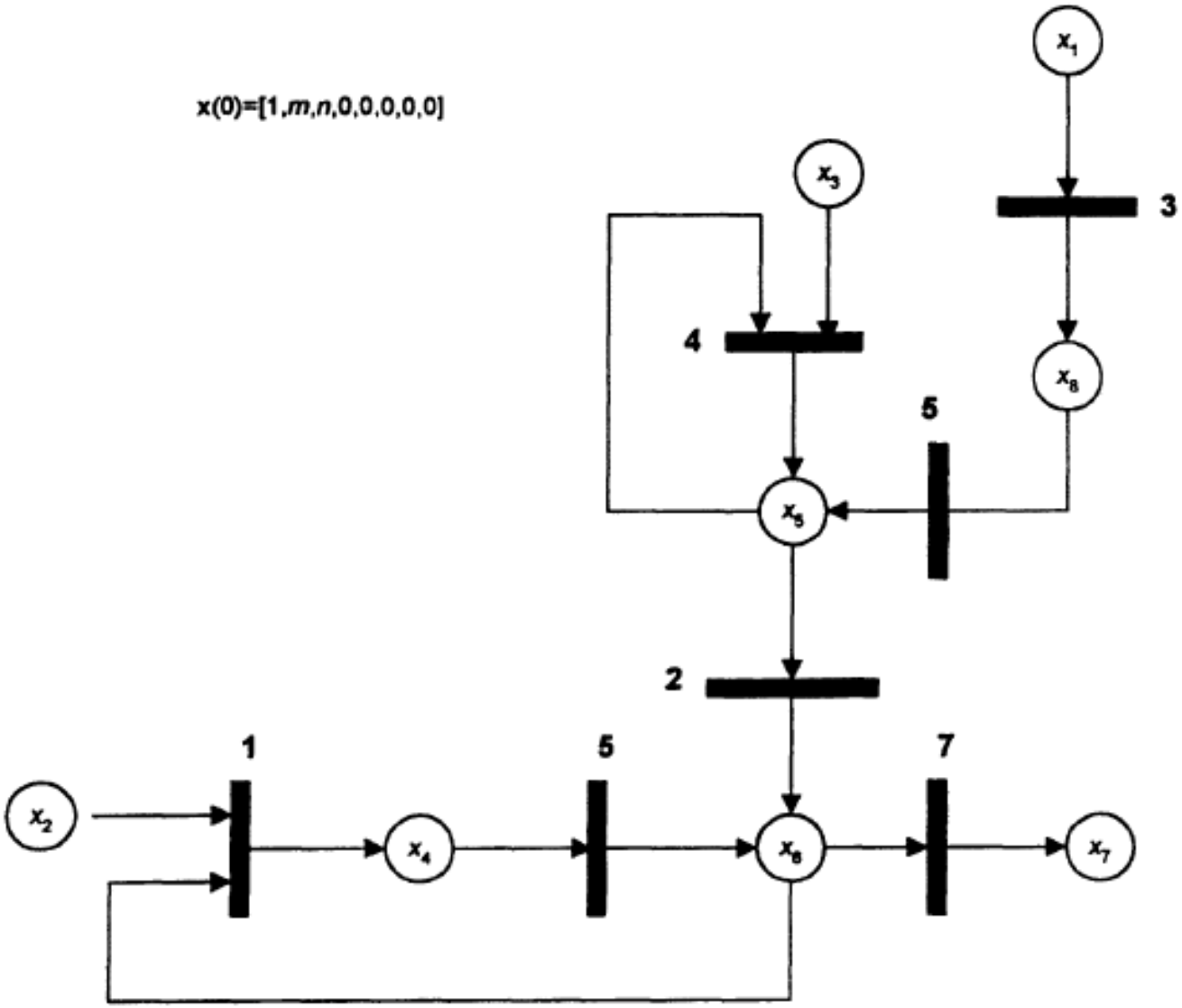
- **Geçişler**

1: Karbiratöre gazın hareketi, 2: Gazın silindire hareketi 3: Başlama pozisyonuna geçiş, 4: Distribütöre akım geçişi, 5. Kontakların kapatılması, 6: Silindirde kıvılcım oluşması, 7: Hızın azaltılması

Çözüm

- Başlangıç aşamasında ateşlemede bir anahtar olmalıdır. ($x1 = 1$), Depoda biraz benzin olmalıdır. ($x2 = m > 0$), ve akümülatörde şarj olmalıdır ($x3 = n > 0$), Burada m ve n daha önceden verilir.
- Diğer bütün lokal durumlar sıfırdır. Böylece başlangıç durum vektörü $x(0) = [1, m, n, 0, 0, 0, 0, 0]$.
- Bu liste kullanılarak aşağıdaki şekilde verilen petri ağı oluşturulabilir.
- İlk başlamadan sonra belirli bir zaman periyodunda benzin gücü ve bataryadan elektrik akımını korumak için iki hafıza elemanı kullanılır.

Örnek:



Otomobil başlatma modeli

Petri Ağların Analizi

- Petri ağları genel ayrık modelleri tanımlamak için genel bir araçtır.
- Dağıtık bir model bağımsız süreçlerin etrafında inşa edilir ve onların her biri sınırlı sayıdaki kaynağın kullanımı için rekabet eder.
- *Örneğin bir bilgisayar ağı 10 bilgisayar ve sadece 2 yazıcıya sahip olabilir.*
- *Eğer birkaç kullanıcı aynı zamanda kendi çalışmasını yazdırmak isterse yazıcı kaynağı için yarışma kaçınılmazdır.*
- *Eğer yazıcı bir yazıcı bekletici oluşturmak için ek disk sürücü kullanımı gerektirirse çoklu kaynak gerekir*.

Örnek:

- Sonsuz bir tampon ile bir üretici/tüketici sistem düşünelim.
- Bu sistem iki sürece sahiptir: Üretici süreci bir nesne oluşturur, tüketici süreci ise nesneyi kullanır. Tampon tüketici ve üretici arasında bir arayüz olarak nesne için geçici bir alan tutar.
- Bu sistemi petri ağlar ile modelleyiniz.
- *Çözüm:*
- Üretici iki olası geçişten birini alabilir: nesne üretme ve tampon kuyruğuna bir nesne ekleme
- Bu geçişler ile ilgili iki yer var vardır: nesne oluşturmaya hazır olma ve tampona nesne eklemeye hazır olma
- Tampon bir paylaşım kaynağıdır ve böylece sadece bir duruma sahiptir. Tampon nesneye sahiptir ve bu depolanan nesne sayısı ile karakterize edilir.

Örnek

- Diğer taraftan tüketicinin geçişleri tampondan bir nesne silme veya nesneyi tüketmeyi içerir
- Her biri bir yer ile ilişkilidir: bir nesneyi silmek için hazır olma ve bir nesneyi tüketmek için hazır olmak
- **Protokol**
- • Bir nesne üretmek:
  - Nesne üretmek
  - Tampona nesne eklemek
- Bir nesne tüketmek
  - Tampondan nesne silmek
  - Nesne tüketmek

Örnek

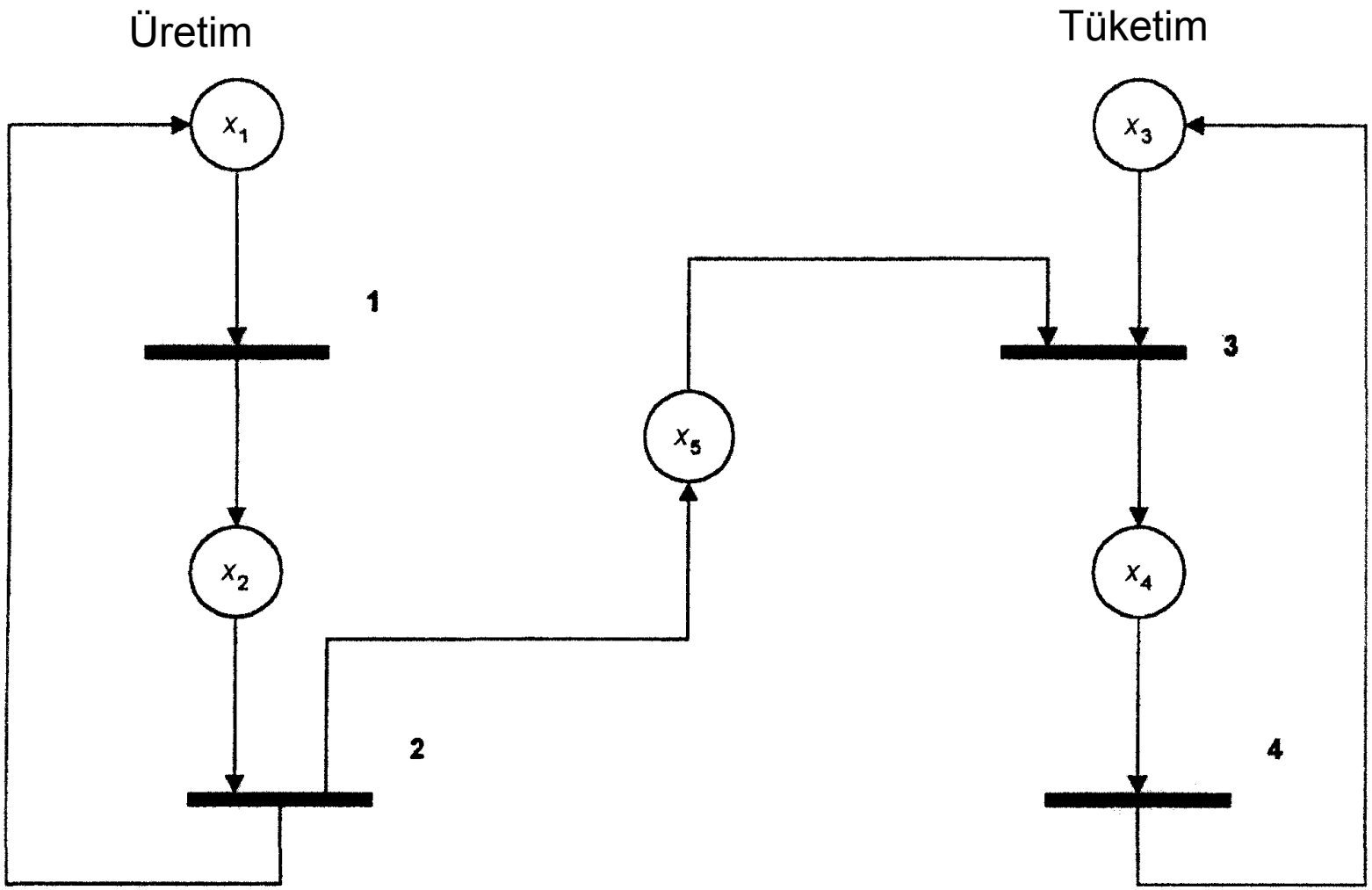
Yerler

Yer adı	Lokal durum
Nesne üretmeye hazır olma	X1
Nesne eklemeye hazır olma	X2
Nesne silmeye hazır olma	X3
Nesne tüketmeye hazır olma	X4
Tampon nesneye sahip	X5

Geçişler

Geçiş adı	Etiket
Nesne üretme	1
Nesne ekleme	2
Nesne silme	3
Nesne tüketme	4

Örnek:



Örnek:

- Durum vektörü bütün sistem durumları x = [*x*1, *x*2, *x*3, *x*4, *x*5] şeklinde beş kayıt ile gösterilir.
- Bu modelin test edilmesinde «üretime hazır olma» ve «silme için hazır olma» yerleri için bir değer ile başlamak yeterlidir.
- Diğer durumlar sıfır olarak alınır.
- Böylece x(0) = [1, 0, 1, 0, 0] olur.

Örnek:

Üretici/tüketici sisteminin benzetimi

```
read n
read x1, x2, x3, x4, x5
k=0
print k, x1, x2, x3, x4, x5
for k=1 to n
      S=∅
      if x1>0 then S=S∪{1}
      if x2>0 then S=S∪{2}
      if x5>0 and x3>0 then S=S∪{3}
      if x4>0 then S=S∪{4}

      Y=RAND(S)

      if y=0 then print "deadlock!" : stop
      if y=1 then x1=x1-1: x2=x2+1
      if y=2 then x2=x2-1: x1=x1+1: x5=x5+1
      if y=3 then x5=x5-1: x3=x3-1: x4=x4+1
      if y=4 then x4=x4-1: x3=x3+1

      print k, x1, x2, x3, x4, x5
next k
```

Örnek:

Üretici/tüketici benzetiminin sonuçları x(0) = [1, 0, 1, 0, 0]

k	x_1	x_2	x_3	x_4	x_5
0	1	0	1	0	0
1	0	1	1	0	0
2	1	0	1	0	1
3	1	0	0	1	0
4	0	1	0	1	0
5	1	0	0	1	1
6	1	0	1	0	1
7	0	1	1	0	1
8	0	1	0	1	0
9	0	1	1	0	0
10	1	0	1	0	1
20	0	1	0	1	0
30	0	1	0	1	1
40	0	1	0	1	0
50	0	1	0	1	1
60	1	0	1	0	1
70	0	1	0	1	5
80	1	0	1	0	6
90	1	0	1	0	3
100	1	0	1	0	6