İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

ZAMANLI AYRIK OLAY SİSTEMLERİ VE UYGULAMALARI

YÜKSEK LİSANS TEZİ

Elektrik Müh. İbrahim ÜNAL

Anabilim Dalı : ELEKTRİK MÜHENDİSLİĞİ

Programı : KONTROL VE OTOMASYON MÜHENDİSLİĞİ

OCAK 2008

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

ZAMANLI AYRIK OLAY SİSTEMLERİ VE UYGULAMALARI

YÜKSEK LİSANS TEZİ

Elektrik Müh. İbrahim ÜNAL
(504051137)

Tezin Enstitüye Verildiği Tarih : 18 Aralık 2007
Tezin Savunulduğu Tarih : 30 Ocak 2008

Tez Danışmanı : Doç.Dr. Salman KURTULAN
Diğer Jüri Üyeleri : Prof.Dr. Leyla GÖREN
Yrd.Doç.Dr. Osman Kaan EROL

OCAK 2008

ÖNSÖZ

Amacı, Zamanlı Ayrık Olay Sistemleri ve Uygulamalarının araştırılması olan bu çalışmada yardımlarını esirgemeyen danışmanım Doç. Dr. Salman Kurtulan'a ve herşeyimi borçlu olduğum aileme en içten teşekkürlerimi sunarım.

Ocak 2008 İbrahim ÜNAL

İÇİNDEKİLER

KISALTMALAR **iv**
TABLO LİSTESİ **v**
ŞEKİL LİSTESİ **vi**
SEMBOL LİSTESİ **vii**
ÖZET **viii**
SUMMARY **ix**

1. GİRİŞ **1**
1.1. Giriş ve Çalışmanın Amacı 1

2. ZAMANLI OTOMAT **3**
2.1. Zamanlı Otomat Tanımı 3
2.2. Saat Yapısı 4
2.3. Olay Zamanlama Dinamiği 9
2.4. Durum Uzay Modeli 13
2.5. Zamanlı Otomat Kuyruk Sistemleri 21
2.6. Olay Programlama Planı 24

3. ZAMANLI PETRİ AĞI **27**
3.1. Zamanlı Petri Ağ Tanımı 27
3.2. Zamanlı Petri Ağ Dinamiği 29
3.3. Zamanlı Petri Ağı Kuyruk Sistemleri 33

4. OTOYOL GİRİŞ DENETİMİ PETRİ AĞ UYGULAMASI **36**
4.1. Otoyol Giriş Denetimi 36
4.2. Zamanlı Petri Ağının Modellenmesi 38
4.2.1. Amaçlananlar 38
4.2.2. Kurallar 38
4.2.3. Amaçlanan Zamanlı Petri Ağının Tasarımı ve Analizi 40
4.3. Zamanlı Petri Ağ Modeli Bilinen Sistemin Simatic Manager Uygulaması 50

5. SONUÇ **54**

KAYNAKLAR **55**

EKLER **56**

ÖZGEÇMİŞ **62**

KISALTMALAR

DES	: Discrete Event Systems
DEDS	: Discrete Event Dynamic Systems
CVDS	: Continuous Variable Dynamic Systems
FIFO	: First In First Out

TABLO LİSTESİ

Sayfa No

Tablo 4.1: Yeşil ışık süresinini ayarlama kuralları..... 39
Tablo 4.2: Trafik Işığı için kurallara göre yerler..... 42
Tablo 4.3: Örnek başlangıç durumlarına göre L1'in ışık süreleri..... 43

ŞEKİL LİSTESİ

Sayfa No

Şekil 2.1	**:** E = {α} olay kümeli bir DES için olay-zaman diyagramı.....	4
Şekil 2.2	**:** E={α, β} ve olayları sürekli aktif olan bir DES için olay-zaman diyagramı.....	6
Şekil 2.3	**:** E = (α, β) ve α olayının her durumda aktif olmadığı olay-zaman diyagramı.....	7
Şekil 2.4	**:** DES'in zamanlı durum otomatı…….....	15
Şekil 2.5	**:** Örnek 2.1 zamanlı otomat için durum geçiş diyagramı.....	17
Şekil 2.6	**:** Örnek 2.1 için olay-zaman diyagramı…….....	17
Şekil 2.7	**:** Örnek 2.2 için süreç ve durum geçiş diyagramı.....	19
Şekil 2.8	**:** Örnek 2.2'deki süreç için olay-zaman diyagramı.....	20
Şekil 2.9	**:** Basit bir kuyruk sisteminin olay-zaman diyagramı….....	23
Şekil 2.10	**:** Olay programlama planı…….....	25
Şekil 3.1	**:** Örnek 3.1 için zamanlı Petri ağı…….....	29
Şekil 3.2	**:** Bir DES'in zamanlı Petri ağ modeli….....	30
Şekil 3.3	**:** Örnek 3.2 için zamanlı Petri ağ…….....	32
Şekil 3.4	**:** Basit bir kuyruk sisteminin zamanlı Petri ağ modeli.....	34
Şekil 4.1	**:** Denetim için algılayıcıların yerleri….....	37
Şekil 4.2	**:** Petri ağı uygulamasının senaryo şekli.....	38
Şekil 4.3	**:** Trafik ışığı için zamanlı Petri ağı…….....	41
Şekil 4.4	**:** Trafik ışığı için Petri ağındaki zamanlı geçişler.....	44
Şekil 4.5	**:** Trafik ışığı zamanlı Petri ağı için başlangıç durumu.....	45
Şekil 4.6	**:** Trafik ışığı zamanlı Petri ağı için ikinci aşama.....	46
Şekil 4.7	**:** Trafik ışığı Petri ağı için üçüncü aşama….....	47
Şekil 4.8	**:** Trafik ışığı Petri ağı için dördüncü aşama…….....	48
Şekil 4.9	**:** Trafik ışığı Petri ağı için son aşama….....	49
Şekil 4.10	**:** Data bloğun genel görünüşü…….....	50
Şekil 4.11	**:** YOL 1 ve YOL 3 bilgilerinin verilme durumu.....	51
Şekil 4.12	**:** Yol bilgisine göre L1 lambasının yeşil-kırmızı yanma süreleri	52
Şekil 4.13	**:** Son durumun oluşma hali.....	53
Şekil A.1	**:** Simulaworks ekran görüntüsü.....	56

SEMBOL LİSTESİ

e	**:** Olay
E	**:** Olay kümesi
T	**:** Sonlu geçiş kümesi
P	**:** Sonlu yerler kümesi
A	**:** Oklar kümesi
w	**:** Okların ağırlık kümesi
$I(t_j)$	**:** t_j geçişine olan giriş
$O(t_j)$	**:** t_j geçişinden olan çıkış
$I(p_i)$	**:** p_i yerine olan giriş
$O(p_i)$	**:** p_i yerine olan çıkış

ZAMANLI AYRIK OLAY SİSTEMLERİ VE UYGULAMALARI

ÖZET

Bu tez çalışmasının amacı ayrık olaylı sistemlerin zamanlı otomat ve zamanlı Petri ağı ile modellenmesine ilişkin kavramların tanıtılması ve analizine ilişkin yöntemlerin incelenmesidir. Ayrık olaylı sistemler (DES) olayların, düzensiz zaman aralıklarında meydana gelmesi ile zaman içinde gelişen ancak dinamik yapısındaki gelişimleri olaylara bağlı olan dinamik sistemlerdir. Ayrık olaylı sistem uygulamalarına günlük hayatta çokça rastlanır. Örnek olarak trafik kontrol sistemleri, esnek imalat sistemleri, bilgisayar haberleşme sistemleri, üretim bantları verilebilir. Tezde ilk önce olay kavramından bahsedilmiş, ayrık olay sistemleri hakkında bilgi verilmiştir. Sonrasında ayrık olaylı sistemler için zamanlama mekanizmalarını tanımlayarak otomat modellerini ve Petri ağlarını daha da genişletip, olay zamanı içeren ayrık olaylı sistemler için bir çerçeve yaratılmış ve bununla ilgili çeşitli örnekler verilmiştir. Daha sonra zamanlı modellere bir uygulama olarak 'Otoyol Giriş Denetimi' hakkında bilgi verilmiş, üstünlükleri ve sakıncalarından bahsedilmiştir. Otoyol giriş denetimi, otoyola konulan algılayıcılardan alınan bilgilerle araç akışının tespit edildiği ve araçların belli bir düzende otoyola alındığı bir denetleme sistemidir. Otoyol giriş denetimi ile ilgili olarak Simulaworks paket programı yardımıyla zamanlı Petri ağı uygulaması gerçekleştirilmiş, programdaki "jeton-oyun animasyonları" ile ağın işleyişi gözlemlenmiştir. En son olarak da Siemens Simatic Manager yazılım programı ile DES modeli olan zamanlı Petri ağı, yazılarak, bilgisayar ortamında gerçeklenmiştir.

TIMED DISCRETE EVENT SYSTEMS AND APPLICATIONS

SUMMARY

The objective of this study is introducing the timed automata and timed Petri net concepts related to modeling and examining the methods for analysing of the discrete event systems as timed models. Discrete event systems (DES) are dynamic systems which evolve in time by the occurrence of events at possibly irregular time intervals but evolution in dynamic structure of system is related to events. DES abound in real-world applications. Examples include traffic control systems, flexible manufacturing systems, computer-communication systems, production lines. Firstly we give definition about 'event' concept and discrete event systems. After that, we describe timing mechanisms for discrete event systems and directly extend automaton models and Petri nets to create a modeling framework for DES that includes event timing and give various examples about them. Then, we give information about the subject "Highway Entrance Control" and state the advantages and disadvantages about it and make a timed Petri net application. Highway Entrance Control is a control system that collects information about vehicle flow from the sensors located on the highway and controls the vehicle flow in a proper regulation. Then, by using Simulaworks, Petri net tool, timed Petri net application on "Highway Entrance Control" was made and running of these application were observed by "token-game animation" of the tool and the analysis of this petri net application. Finally, the timed Petri net we have its DES model was generated and realized with Siemens Simatic Manager software tool.

1. GİRİŞ

1.1 Giriş ve Çalışmanın Amacı

Fen ve teknolojideki hızlı gelişmeler alışılagelmiş diferansiyel veya fark eşitlikleriyle açıklanamayan birçok sistemi de beraberinde getirmiştir. Esnek imalat sistemleri, bilgisayar ağı sistemleri, çeşitli nakil (ulaşım) sistemleri ve diğerleri bu sistemlere örnek olarak verilebilir. Bu sistemlerin davranışları çoğunlukla, kendi içlerinde işlemekte olan ayrık (discrete) olaylarla belirlenir. Karakteristikleri, eş zamanlılık, asenkron işlemler, olay sürümlülük ve belirsizlik olan böyle sistemlere, ayrık olay sistemleri (Discrete Event Systems - DES) veya ayrık olay dinamik sistemleri (Discrete Event Dynamic Systems - DEDS) adı verilir. Ayrık olaylar meydana geldiklerinde sistemi bir durumdan diğer bir duruma geçirirler. Bu tür sistemler otomat veya Petri ağları ile modellenebilir. Ancak, üretim sistemlerinin denetiminde belirli olayların belirli gecikmelerle oluştuğu gözlenmektedir. Uygulamada bu türden gecikmeler Programlanabilir Lojik Kontrolörlerde (PLC) zamanlayıcı adı verilen fonksiyonlarla gerçeklenmektedir. Böylece sistemlerin zamana bağlı analiz veya tasarımı, zamanlı modellerin kullanımını gerektirmiştir.

Yapılan çalışmada formal otomat ve Petri ağ yöntemlerinin kapsamını daha genişletmek üzere olayların zamana bağlı ilişkisini tanımlayabilecek yöntemler araştırılmıştır. Bu doğrultuda olayların oluş zamanını da içeren yöntemler incelenmiştir. Bu araştırmalar sonucunda literatürde "Zamanlı Modeller-Timed Models" olarak adlandırılan model sınıfına ilişkin bilgiler verilecektir.

Olayların oluş zamanını içeren formal bir model yapısı oluştururken hangi olayın ne zaman olacağı bilgisinin önceden verilmiş olduğu varsayılmaktadır. Eğer bir ayrık olay sisteminin modellenmesi ile ilgileniliyorsa bu varsayım çok gerçekçi değildir; olayların oluş zamanının kesin olarak belirlenmesi çoğu zaman mümkün olamamaktadır. Bu nedenle ayrık olay sistem modellemesinde gerçekçi bir model yapısının rastlantısal

olması kaçınılmazdır. Ancak, bu tez çalışmasındaki amaç, sistem modellemesinden çok, tasarlanmış üst denetleyicilerin doğrudan gerçeklenmesini mümkün kılacak formal bir gösterilim elde etmektir.

Amaç bu gösterilimde kullanılacak tanımların üst denetleyicinin gerçekleneceği ortamla ilgili büyüklükler üzerine kurulu olması, böylelikle verilen otomat ve Petri ağ tanımının doğrudan gerçeklenebilir olmasıdır. Otomat ve Petri ağ tanımının zamanlama özelliğini de kapsayacak şekilde genişletilmesi yine bu amaç çerçevesinde gerçekleşecektir.

İlk bölümde, zamanlama mekanizması anlatılarak otomat ve Petri ağları genişletilip, olay zamanı içeren ayrık olay sistemleri için bir çerçeve yaratılmıştır. İkinci bölümde ise bunu kullanarak Simulaworks Petri ağı paket programı yardımıyla 'Otoyol Giriş Denetimi' uygulaması için zamanlı bir Petri ağı tasarlanmış ve bu ağın işleyişi, paket programdaki "jeton-oyun animasyonu" ile gözlemlenmiştir. Son bölümde ise tasarlanan zamanlı Petri ağ modeli Siemens Simatic Manager yazılımı ile bilgisayar ortamında gerçeklenmiştir.

2. ZAMANLI OTOMAT

2.1 Zamanlı Otomat Tanımı

Ayrık olay sistemlerinin (DES) gösterilimi için kullanılan sonlu durum makineleri ya da standart otomat gösterilimi kullanılarak modellenen sistemdeki olayların oluş sırasına ilişkin bilgi oluşturulabilir, ancak olayların oluş zamanları için bir bilgi içermez. Eğer modelde olayların olma zamanları ile ilgileniliyorsa modelin $\{e_0, e_1, \ldots\}$ şeklinde olay dizileri ve $\{x_0, x_1, \ldots\}$ şeklinde durum dizileriyle beraber $\{(x_0, t_0), (x_1, t_1), \ldots\}$ şeklinde durum ve durum geçişlerine ilişkin zamanları da içermesi gerekmektedir [1].

Zamanlı otomat tanımı için ilk aşamada

X sayılabilir durum uzayı

E sayılabilir olay dizisi

$f: X \times E \rightarrow X$ durum geçiş fonksiyonu

$\Gamma: X \rightarrow 2^E$ aktif olay kümesi

x_0 başlangıç durumu

olmak üzere zamanlı (X, E, f, Γ, x_0) otomat modeli temel alınacaktır. Standart otomat tanımından farklı olarak burada X ve E kümelerinin genel sayılabilir (sonlu olmak zorunda değil) olmasına izin verilmektedir. Ayrıca kilitlenme ile ilgilenilmediğinden işaretli durum tanımı da kullanılmamaktadır. Zamanlı otomat tanımını tamamlamak için olayların zamana bağlı davranışını tanımlayan "Saat Yapısı" kavramı tanıtılmalıdır. Bu kavram aşağıda basit örnekler üzerinden anlatılmıştır.

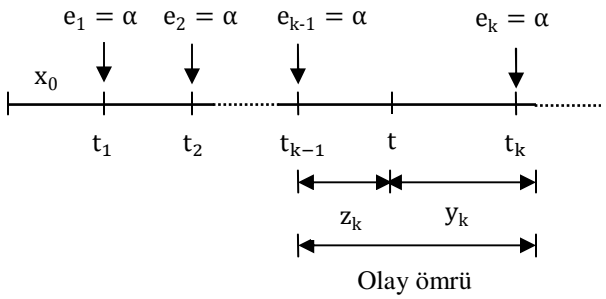
2.2 Saat Yapısı

Zamanlama mekanizmasını tanımlamak için gerekli kavramları açıklayabilmek üzere en basit DES örneği ele alınacaktır. Bir genellemeye varmak üzere daha karmaşık örneklerle devam edilecektir [1].

Tek olaylı bir DES: $E = \{\alpha\}$ ve her $x \in X$ için $\Gamma(x) = \{\alpha\}$ olsun. Bu durumda bu DES için olay-zaman diyagramı Şekil 2.1'deki gibi verilebilir. Bu olay-zaman ilişkin olay dizilimi $e_k = \alpha$, k=1, 2, … olmak üzere $\{e_1, e_2, ...,\}$ ile gösterilebilir. Olayların oluş anları ise t_k, k=1, 2, … ile gösterilmektedir. İki olay arasında geçen süreye olay ömrü adı verilmektedir. Buna göre olayın k. ömrü,

$$\nu_k = t_k - t_{k-1} \qquad k = 1,2,\ldots \qquad \textbf{(2.1)}$$

ile tanımlanır.



Şekil 2.1 : $E = \{\alpha\}$ olay kümeli bir DES için olay-zaman diyagramı

Bu sistemin zamana bağlı davranışı şu şekilde tanımlanabilir. t_{k-1} anında k. olay aktiftir ya da izinlidir denilir. Ve bu olay için ν_k olay ömrü verilmiş olur. Tam bu anda bu olaya ilişkin bir saat kurulur ve geri saymaya başlar. $t_k = t_{k-1} + \nu_k$ anında olay ömrü tamamlanır ve olay meydana gelir. Bu da bir durum değişikliğine neden olacak ve (k+1). Olay aktif hale gelecektir. (k+1). Olay aktifken sistem kendini aynı şekilde tekrar edecektir.

Bu noktada olayın oluşuyla aktif hale gelmesi arasındaki farkı vurgulamak gerekir. Olayın aktif olmasından kasıt otomatın mevcut durumunun aktif olay kümesi içerisinde olmasıdır. Bu olaya ilişkin saatin sıfıra ulaşması ile olay meydana gelir. Ve bir durum geçişine neden olur. Bu Petri ağlarındaki bir geçişi "yetkilendirme" ile onu "ateşleme" arasındaki ayrıma benzetilebilir. Şekil 2.1'deki α olayı her zaman aktiftir. Ancak sadece

$t_1, t_2, \ldots$ anlarında meydana gelmektedir. Burada α olayının her meydana gelişinde yeniden aktive olmaktadır, çünkü her durumda α olayı aktif olay kümesi içerisinde yer alır [1].

Daha ileri noktaların anlaşılması için, olay oluşumu ile bağıntısız olan herhangi bir t anı seçili olsun. $t_{k-1}<t< t_k$ olduğunu varsayılsın. Şekil 2.1'de t, $[t_{k-1}, t_k]$ aralığını iki parçaya böler.

Burada,

$$y_k = t_k - t \qquad \textbf{(2.2)}$$

k. olayın saati yada artık yaşam süresi ve

$$z_k = t - t_k \qquad \textbf{(2.3)}$$

k. olayın yaşı olarak adlandırılırlar.

$$v_k = z_k - y_k \qquad \textbf{(2.4)}$$

Açıktır ki olay ömrü dizisi $\{v_1, v_2, \ldots\}$ verildiğinde bu DES'in olay-zaman diyagramı tam olarak tanıtılmış olur. Bu diziye aynı zamanda α olayının saat dizisi adı da verilmektedir.

Sürekli olarak aktif iki olayı olan DES: Bu örnekte olay kümesi E={α,β} olan bir DES ele alınacaktır. Basitlik olması açısından her $x \in X$ için Γ(x)={α,β} olduğu, yani her iki olayın her an aktif olduğu varsayılacaktır. Her bir olaya ilişkin saat dizileri,

$$\mathbf{v_\alpha} = \{v_{\alpha,1}, v_{\alpha,2}, \ldots\} \qquad \mathbf{v_\beta} = \{v_{\beta,1}, v_{\beta,2}, \ldots\}$$

olarak verilmiş olsun. $v_{\alpha,1}$ ve $v_{\beta,1}$ değerlerinin karşılaştırılması ile bir t_0 anında bir sonraki olayın hangisinin gerçekleşeceği tespit edilebilir. Örneğin

$$v_{\alpha,1}<v_{\beta,1}$$

Koşulu sağlanıyor ise $t_1 = t_0 + v_{\alpha,1}$ anında olacak olay α'dır. t_1 anından sonra meydana gelecek ilk olayın tespiti içinse saat değerlerinin yeniden karşılaştırılması gerekecektir.

Ancak β olayı hala aktiftir ve yeni saat değeri aşağıdaki gibi hesaplanır.

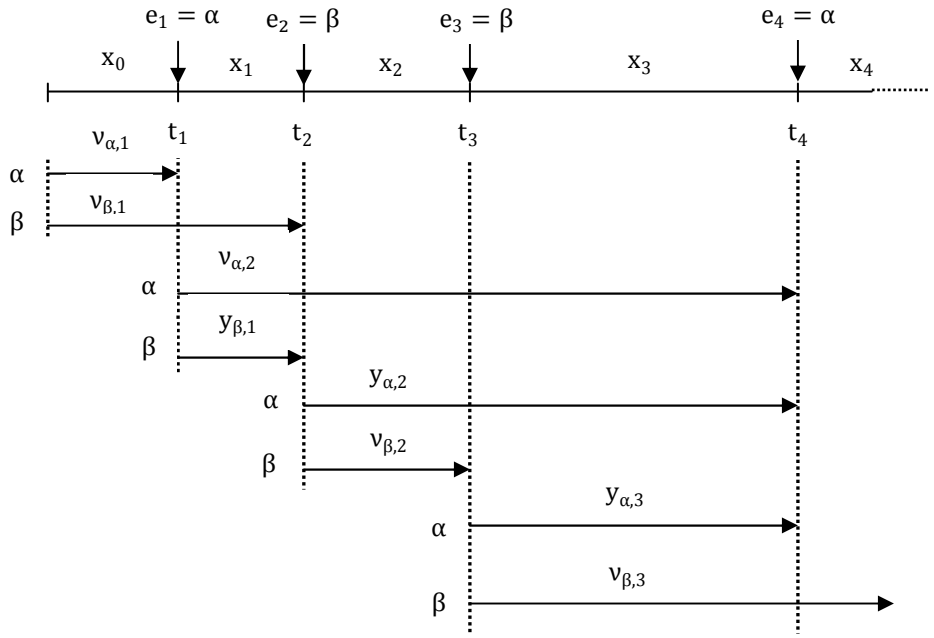
$$y_{\beta,1} = \nu_{\beta,1} - \nu_{\alpha,1}$$

Bu durumda bir sonraki olayın tespiti için $\nu_{\alpha,2}$ ile $y_{\beta,1}$'in karşılaştırılması yapılmalıdır. Eğer,

$$y_{\beta,1} < \nu_{\alpha,2}$$

ise $t_2 = t_1 + y_{\beta,1}$ anında meydana gelecek olay β'dır.

Olacak bir sonraki olayın tespiti sonuç olarak saat değerlerinin karşılaştırılması ve en küçük olanının seçilmesine dayanmaktadır. Eğer bir olay henüz olmuşsa saat değeri saat dizisindeki bir sonraki değerle güncellenir.

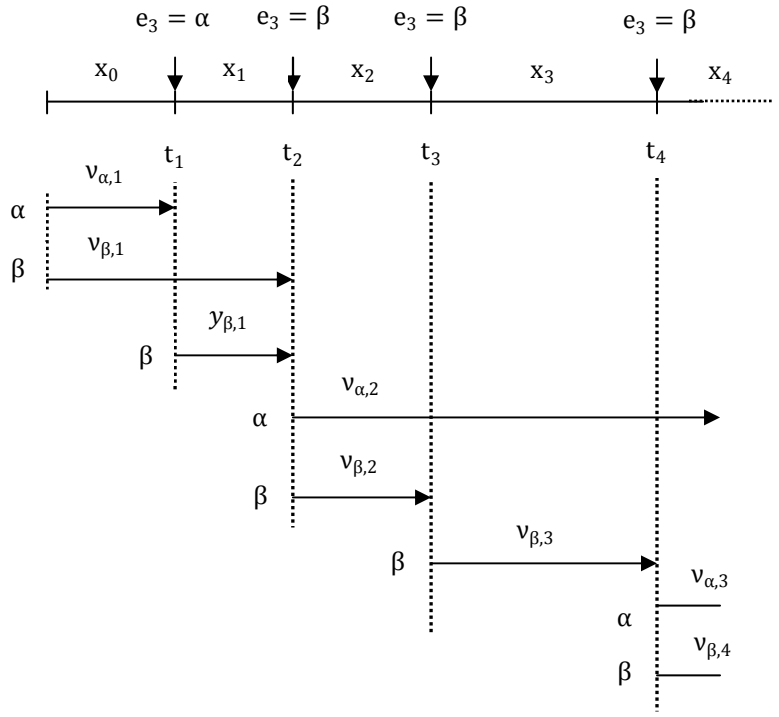


Şekil 2.2 : E={α, β} ve olayları sürekli aktif olan bir DES için olay-zaman diyagramı

Sürekli olarak aktif olmayan iki olaylı bir DES: Olay-zaman diyagramı Şekil 2.3'teki gibi olan bu örnekte olayların tümünün her durumda aktif olmadığı DES yapısı incelenecektir. Bir önceki örnekte bazı $x \in X$ için $\Gamma(x)=\{\alpha, \beta\}$ ve diğer $x \in X$ için $\Gamma(x)=\{\beta\}$ olduğu varsayılsın. x_0 ilk durumu için $\Gamma(x_0)=\{\alpha, \beta\}$ olsun. Bu durumda bir önceki örnekteki gibi ilk olacak olay α'dır.Yeni x_1 durumu için $\Gamma(x_1)=\{\beta\}$ olduğu kabul

edildiğinde aktif olan olay tek olduğu için saat değerlerinin karşılaştırılmasına gerek yoktur. t_2 anında β olayı olur ve x_2 durumuna geçilir [1].

Bu durumda her iki olayın da aktif olay kümesi içinde olduğunu düşünelim. Böylece her iki olay da yeni saat değerlerini alırlar ve $\nu_{\beta,2} < \nu_{\alpha,2}$ olduğundan bir sonraki olay β'dır. Geçilen yeni x_3 durumunda $\Gamma(x_3)=\{\beta\}$ olsun. t_4 anında ise β olayı gerçekleşir ve x_4 durumuna geçilmiş olunur. $\Gamma(x_4)=\{\alpha, \beta\}$ olduğu varsayılırsa her iki olay da aktif olur. Bu durumda t_3 anında α olayı deaktive edildiğinden $\nu_{\alpha,2}$ geçerliliğini kaybeder ve α için tamamen yeni bir $\nu_{\alpha,3}$ saat değeri atanır. Sonuç olarak sistem davranışı olayların aktif olmalarına ve güncellenen saat değerlerinin karşılaştırılmasına bağlı olarak belirlenmiş olur.



Şekil 2.3 : E = (α, β) ve α olayının her durumda aktif olmadığı olay-zaman diyagramı

Bu örnekte α, x_1 ve x_3 durumlarında gerçeklenebilir değildir. t_1'de, α meydana geldikten sonra tekrar aktive edilmemiştir. t_3 anında, β olayı meydana geldiği zaman, α açıkça etkinsizleştirilmiştir. Yani α'nın saat değeri bu noktada zaman diyagramından çıkartılmıştır. t_4 anında tekrar aktive edildiğinde ise, α'nın saatine yeni bir olay ömrü atanmıştır. Bir olay aktive edilirken kendi saat yapısına uygun yeni bir olay ömrü uygulanır. Fakat bir olay etkinsizleştirilirken saat yapısı zaman doğrusundan çıkartılır.

Sonuç olarak sıradaki olaya karar verilirken:

(a) mevcut durum olan x_k bilgisinin ki o $\Gamma(x_k)$ tarafından oluşturulur ile

(b) $i \in \Gamma(x_k)$ için tüm mevcut saat değerlerinin $y_{i,k}$, bilinmesi gerekmektedir.

"Sıradaki olayı" seçme mekanizması üç basit kurala dayandırılabilir:

Kural 1. Geçerli durumdaki tüm olayların saat değerleri karşılaştırılır ve en küçüğü seçilir.

Kural 2. Bir e olayının aktif olabilmesi için,

- e olayı henüz meydan gelmiş ve yeni durumda da gerçeklenebilir kalıyor (örneğin t_2 zamanında gerçekleşen β olayı gibi, Şekil 2.3) ya da,
- e olayı gerçeklenebilir değilken farklı bir olay meydana gelmiş ve bu da e olayının gerçeklenebilir olduğu yeni bir duruma geçiş teşkil ediyor (örneğin t_2 zamanında β olayı meydana geldiği zamanki α olayı gibi, Şekil 2.3) olması gerekmektedir.

Kural 3. Farklı bir olay, e olayının gerçeklenebilir olmadığı yeni bir duruma geçiş sağladığı zaman e etkisizleştirilmiştir denir (örneğin t_3 zamanında β olayı meydana geldiği zamanki α olayı gibi, Şekil 2.3).

Tanım: E olay kümesine ilişkin saat yapısı (ya da zamanlama yapısı)

$$V = \{\mathbf{v_i}: i \in E\}, \qquad \mathbf{v_i} = \{\nu_{i,1}, \nu_{i,2}, ...\}, \qquad i \in E, \nu_{i,k} \in R^+, \qquad k = 1, 2, ...$$

ile verilen olay ömrü dizilerinin bir kümesidir.

Saat yapısı, ele alınan DES için bir giriş olarak düşünülecektir. Saat yapısı bilgisiyle ele alınan DES'e ilişkin otomatın üreteceği olay dizisi elde edilir. Bu nedenle verilen bir DES için saat yapısı **V**'nin de tam olarak belirlenmiş olduğu varsayılacaktır [2].

Başka bir kullanışlı kavramda olay skorudur.

Tanım. Bir $i \in E$ olayına ilişkin $N_{i,k}$ ile gösterilen olay skoru, olay-zaman diyagramında k. durum geçişinden sonra $[t_0, t_k]$ aralığında i'nin kaç kez aktive olduğunu gösteren sayıdır.

Örneğin, Şekil 2.2'de t_0 anında her iki olay da aktive edildiğinden olay skorları $N_{\alpha,0} = N_{\beta,0} = 1$ olarak belirlenmiştir. t_1 anında $N_{\alpha,1} = 2$ ve $N_{\beta,1} = 1$ olur. t_2 anında ise olay skorları $N_{\alpha,2} = N_{\beta,2} = 2$ olarak belirlenir. Sonuç olarak her olay oluşunda olaylardan yeniden aktive olanların bulunup ilgili skorların 1 artırılması gerekir. Bir i olayının skoru, bir sonraki aktive oluşunda atanacak olay ömrünün belirlenmesi için $\mathbf{v_i}$ saat dizisine ilişkin bir işaretçi olarak kullanılır [2].

2.3 Olay Zamanlama Dinamiği

Değişkenleri (X, E, f, Γ, x_0) olan bir otomat için, olay kümesi E'nin sonlu ve m elemandan oluştuğu varsayılacaktır. Sistem dinamikleri durum geçiş denklemi tarafından sağlanır.

$$x_{k+1}= f(x_k, e_{k+1}), \quad k = 0, 1, ... \qquad \textbf{(2.5)}$$

Burada x_k şimdiki durum, x_{k+1}, e_{k+1} olayı olurken elde edilen sıradaki durumdur. Bu çerçevede $\{e_1, e_2 , ..., e_{k+1}, ...\}$ olaylarının verilen bir dizisi sisteme girişi oluşturur. "Durum" tanımı hatırlandığında görülür ki x_k'lı olay giriş bilgisi (2.5) denklemine kadar olan tüm gelecekteki durum değişiklerini tahmin etmede yeterlidir.

Şimdi değişkenleri (X, E, f, Γ, x_0) ve saat yapısı $V = \{\nu_i: i= 1, ..., m \}$ olan zamanlı otomatı oluştururken bu saat yapısı otomata giriş olarak verilmiş olsun. Olay dizisi $\{e_1, e_2 , ..., e_{k+1}, ...\}$ bilinmediği varsayılacaktır. Aslında, bu açıkça tanımlanacak bazı zamanlama mekanizmalarını temel alarak kararlaştırılmış olmak zorundadır. Diğer bir deyişle,

$$e_{k+1}= h(x_k, \nu_1, ..., \nu_m)$$

denkleminde bir ilişki aranırsa, böylece denklem (2.5)'i bir durum denklemi olan

$$x_{k+1}= f(x_k, \nu_1, ..., \nu_m)$$

ile değiştirilebilir.

DES'in dinamik davranışı tarif edilirken durum değişkeni olan x_k gereklidir. Bir önceki bölümde yapılan tartışma, gerçekte bir sonraki olayı belirlemenin, e_{k+1}, saat mukayeselerini, güncellemelerini ve ayrıca olayın puanını takip etmeyi içerdiğini öngörmektedir [3].

Bu nedenle amaçlanan, aşağıda belirtilen form tarafından yakalanan bir iç zamanlama mekanizması ile bir sonraki olayı belirlemektir.

$$e_{k+1} = h(x_k, \nu_1, ..., \nu_m, \bullet) \qquad \textbf{(2.6)}$$

Burada "." henüz belirlenmemiş olan ilave durum değişkenlerini temsil etmektedir. Bu ilişki tam olarak belirlendikten sonra (2.5) ve (2.6) denklemleri sistemin dinamiğini tarif edecektir.

Aşağıda şu ana kadar bahsedilen olay-zaman ilişkisi için formal bir tanım verilmeye çalışılacaktır. Gösterim kolaylığı açısından şu notasyon kullanılacaktır:

- x mevcut durum
- e x durumuna geçişe neden olan en son olay
- t e olayına karşılık gelen en son olay oluş anı

Bunlara ilaveten, her $i \in E$ olayına ilişkin iki değişken daha tanımlanacaktır:

- N_i i olayının şu anki skoru $N_i \in \{0, 1, ...\}$
- y_i i olayının şu anki saat değeri, $y_i \in R^+$

N_i ve y_i'nin tarifleri formal olmayan şekilde daha önce verilmişti, fakat aşağıda takip eden kısımda matematiksel olarak kesin hale getirilecektir [3].

Bir sonraki durum, olay, olay oluş anı, olay skoru ve saat değeri için (′) notasyonu kullanılacaktır. Buna göre x gösterimi x_k'ya, x' ise x_{k+1}'e karşılık gelmektedir ve benzer olarak daha önce tanımlanmış olan bütün diğer değişkenler içinde aynısı geçerlidir.

- e' sıradaki olaydır, "tetikleyici olay" olarak da anılır; açıktır ki $e' \in \Gamma(x)$
- t' sıradaki olay zamanıdır (e'olayına denk gelmektedir)
- x' sıradaki durumdur, $x' = f(x, e')$ ile verilmektedir

Benzer şekilde aşağıdaki belirleme yapılabilir:

N_i' i olayının sıradaki skorudur (e′olayının vuku bulmasından sonra)

y_i' i olayının sıradaki saat değeridir (e′olayının vuku bulmasından sonra)

Bir x durumu verildiğinde ilk belirlenmesi gereken e′ olayıdır. Şu adımlar izlenecektir.

Adım 1. x bilindiği için, Γ(x) gerçeklenebilir olay kümesi belirlenebilir.

Adım 2. Her i ∈ Γ(x) olayı ile ilgili olarak bir y_i saat değeri tanımlıdır. Bunlar arasında y^* ile gösterilen en küçük saat değeri aşağıdaki gibi belirlenir.

$$y^* = \min_{i \in \Gamma(x)} \{y_i\} \qquad \textbf{(2.7)}$$

Adım 3. y^* saat değerine karşılık gelen olay bulunur:

$$e' = \arg\min_{i \in \Gamma(x)} \{y_i\} \qquad \textbf{(2.8)}$$

Adım 4. e′ olayından faydalanarak bir sonraki durum belirlenir:

$$x' = f(x, e') \qquad \textbf{(2.9)}$$

Adım 5. Belirlenmiş olan y^* saat değerinden faydalanılarak bir sonraki olay anı bulunur:

$$t' = t + y^* \qquad \textbf{(2.10)}$$

x′, e′ ve t′ belirlendikten sonra bu işlem tekrarlanacaktır. Bununla birlikte Adım 2, yeni y_i' saat değerlerinin belirlenmiş olmasını gerektirir. Bu nedenle en az bir adıma daha gereksinim vardır.

Adım 6. Aktif olan yeni i ∈ Γ(x′) olayları için yeni saat değerleri bulunmalıdır:

$$y_i' = \begin{cases} y_i - y^* & i \neq e' \text{ ve } i \in \Gamma(x) \\ v_{i,N_i+1} & i = e' \text{ ve } i \notin \Gamma(x) \end{cases} \quad i \in \Gamma(x') \qquad \textbf{(2.11)}$$

Adım 6 olabilecek iki durum için saat değerinin belirlenmesini tanımlamaktadır:

(a) Yeni x' durumunda da aktif olarak kalan ve bu yeni duruma geçişe neden olan olaylardan farklı olayların saat değerleri y^* kadar azaltılmalıdır.

(b) Yeni duruma geçişe neden olan e' olayı ya da bir önceki durumda aktif olmayıp yeni durumda aktif olan olaylara yeni saat değerleri atanmalıdır.

(c) Yeni durumun aktif olay kümesinde bulunmayan olaylar için saat değerleri tanımsızdır. Bu nedenle i olayı aktifken e' olayının olmasıyla geçilen yeni durumda saat değerleri geçerliliğini yitirir.

Adım 6'nın uygulanabilmesi için N_i' yeni skor değerlerinin belirlenmesi gerekir. Bunun için son bir adıma daha gereksinim vardır:

Adım 7. $i \in \Gamma(x')$ için yeni skor değerleri belirlenmelidir:

$$N_i' = \begin{cases} N_{i+1} & i = e' \text{ veya } i \notin \Gamma(x) \\ N_i & \text{aksi halde} \end{cases} \quad i \in \Gamma(x') \qquad \textbf{(2.12)}$$

Burada i olayının skoru, olay aktif hale geldiğinde 1 artırılmaktadır. Bu $i \in \Gamma(x')$ ve i tetikleyen olay olduğu zaman olmakta veya i aktif halde olmayıp yeni duruma, x', girilmesi ile aktif hale gelince meydana gelmektedir [6].

En küçük olay saat değeri, y^*, Adım 2'de belirlenmektedir, bu ayrıca birbirini takip eden iki başarılı olay tarafından tanımlanan zaman aralığının uzunluğunu da belirlemektedir. Bu nedenle y^* için olaylar arası süresi terimi kullanılacaktır.

Yorum. i, $j \in \Gamma(x)$ gibi İki farklı olay i ve j için $y_i = y_j$ olabileceği düşünüldüğünde, olay kümesi E üzerinde bazı öncelik kurallarının uygulanması lazımdır. Eğer iki olayın meydana gelmesi eş zamanlı ise, olaylardan hangisinin ilk önce durumu etkiliyor olmasının belirlenmesi gerekir. Aksi belirtilmediği takdirde, olayların E'de önceliklerine göre sıralandığı varsayılacaktır. Örneğin i olayı, j olayından daha yüksek önceliğe sahip ise $j>i$ olduğu anlaşılır. Uygulamada, sürekli zaman içinde çalışırken, bu problem ortaya çıkmaz, bunun oluşabilmesinde iki veya daha fazla olayın eş zamanlı olarak meydana gelmeleri için zorlayıcı kontrol kurallarının olması lazımdır [6].

Şimdi artık zamanlı otomata resmi bir tanım verebiliriz.

Tanım. V = $\{\mathbf{v_i} : i \in E\}$ saat yapısı ve (X, E, f, Γ, x_0) bir otomat olmak üzere Zamanlı Otomat,

$$(X, E, f, \Gamma, x_0, \mathbf{V})$$

altılısı ile tanımlanır. Otomat,

$$e' = \arg\min_{i \in \Gamma(x)} \{y_i\}$$

kuralı ile üretilen $\{e_1, e_2, \ldots\}$ olay dizisi ile sürülen $x'=f(x, e')$ durum dizilerini üretir. Olaylar değerleri her adımda güncellenen,

$$y_i' = \begin{cases} y_i - y^* & i \neq e' \text{ ve } i \in \Gamma(x) \\ v_{i,N_i+1} & i = e' \text{ ve } i \notin \Gamma(x) \end{cases} \qquad i \in \Gamma(x')$$

saat değerlerine bağlı olarak üretilir. Burada y^* olaylar arası süresi,

$$y^* = \min_{i \in \Gamma(x)} \{y_i\}$$

ve olay skorları N_i, $i \in E$,

$$N_i' = \begin{cases} N_{i+1} & i = e' \text{ veya } i \notin \Gamma(x) \\ N_i & \text{aksi halde} \end{cases} \qquad i \in \Gamma(x')$$

ile tanımlanır. İlk değerler ise $i \in \Gamma(x_0)$ için $y_i=v_{i,1}$ ve $N_i=1$; $i \notin \Gamma(x_0)$ için y_i tanımsız ve $N_i=0$ olarak belirlenir.

2.4 Durum Uzay Modeli

Burada amaçlanan, esasında zamanlı otomat tanımının dinamik bir sistem için, bir durum uzay modelinin tarifi olduğunu vurgulamaktır [2].

(2.9) denkleminde tarif edilen otomat durum geçiş mekanizması ile başlanırsa:

$$x' = f(x, e')$$

burada görüldüğü gibi e′, (2.8) denklemi aracılığı ile belirlenmelidir.

$$e' = \underset{i \in \Gamma(x)}{\arg\min} \{y_i\}$$

y_i tekrarlanan (2.11) denklemi ile tanımlandığından, bu modelin durum değişkeni olarak alınmalıdır. Buna ilaveten, (2.11) denklemi N_i'ye bağlıdır.

N_i tekrarlanan bir şekilde (2.12) denklemi aracılığı ile tanımlanmaktadır ve buda bir durum değişkenidir. Şimdi bütünüyle durum uzay modeli tanınmıştır, yani x, y_1, …, y_m, N_1, …, N_m durum değişkenleridir ve ν_1, …, ν_m sistemi yönlendiren giriş dizileridir. Modelin daha kısa bir tanımını sağlayabilmek için, şu kıstas getirilmelidir:

f_i^x (.) : (2.9) denkleminde sıradaki durum fonksiyonu

f_i^y (.) : (2.11) denkleminde i. sıradaki saat fonksiyonudur, $i \in \Gamma(x')$

f_i^N (.) : (2.12) denkleminde i. sıradaki skor fonksiyonudur, $i \in \Gamma(x')$

Bu durumda y ve N'nin mevcut saat ve skor (sütun) vektörleri olmalarına izin verilir.

$$y = [y_1, \ldots, y_m]^T, \qquad N = [N_1, \ldots, N_m]^T$$

Burada belirtilmesinde fayda olan husus, durum değişkenlerinin N_1, …, N_m sadece kendilerini güncellemek ve olayların vuku bulmalarının sayılmasına yardımcı olmak için kullanıldığıdır. Bununla birlikte bunlar, x otomat durumu veya y_1, …, y_m saat değerlerini etkilemezler [3].

Şimdi model için gerekli olan durum denklemleri aşağıdaki gibi yazılabilir:

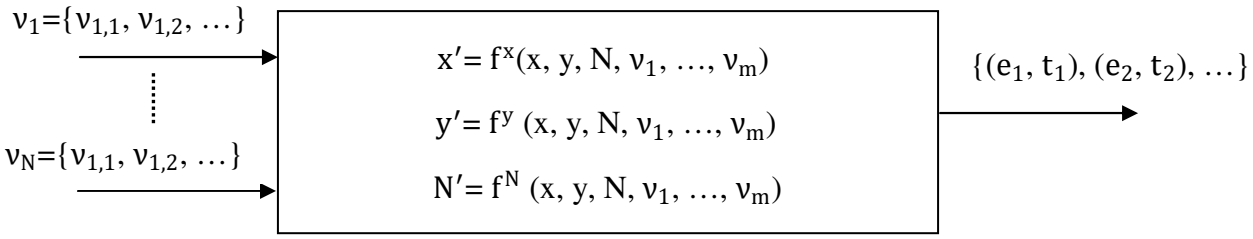
$$x' = f^x(x, y, N, \nu_1, \ldots, \nu_m) \qquad \textbf{(2.13)}$$

$$y' = f^y(x, y, N, \nu_1, \ldots, \nu_m) \qquad \textbf{(2.14)}$$

$$N' = f^N(x, y, N, \nu_1, \ldots, \nu_m) \qquad \textbf{(2.15)}$$

Model, Şekil 2.4'te özetlenmiştir. Giriş, $\mathbf{V}=\{\nu_i: i=1, ..., m\}$ saat yapısından oluşmaktadır. Çıkış ise zamanlı olay dizisi olarak dikkate alınmakta ve DES'in olay-zaman

diyagramını tanımlamaktadır. Burada kaydedilmesi gereken husus, gerçek zaman değişkeninin, (2.10) denklemi aracılığı ile güncellendiği ve modeldeki dinamiklerin başlıca bir kısmı olmadığıdır. Bununla birlikte, $\{(e_1, t_1), (e_2, t_2), \ldots\}$ çıkış serisini tanımlamakta kolaylık sağlamaktadır.



Şekil 2.4 : DES'in zamanlı durum otomatı

Şekil 2.4'teki zamanlı durum otomatı üç durum denklemi (2.9), (2.11) ve (2.12) ile verilmiştir ve zaman $t' = t + y^*$ ile güncellenmektedir. Burada y^* denklem (2.7) aracılığı ile elde edilmektedir.

Modelin başlangıç koşulları bir önceki bölümde zamanlı otomatın tanımı içinde belirtildiği gibidir. x_0 bilindiği zaman, başlangıçta aktif olan bütün $i \in \Gamma(x_0)$ olayları için $N_{i,0} = 1$ olarak ayarlanır ve eğer $i \notin \Gamma(x_0)$ ise o zaman $N_{i,0} = 0$ olarak ayarlanır. Ayrıca bütün $i \in \Gamma(x_0)$ olayları için, $y_{i,0} = \nu_{i,1}$ olarak bir belirleme yaparken $i \notin \Gamma(x_0)$ için $y_{i,0}$ tanımsız olmaktadır.

Yorum. Bu model ile ilgili olarak, "durum" teriminin kullanılmasında bir zorluk mevcuttur. Sistemin "durumu" $(x, y_1, \ldots, y_m, N_1, \ldots, N_m)$ değişkenlerinden oluşmaktadır. Bütün bu değişkenler, saat yapısı V ile birlikte, DES'in gelecekteki davranışını öngörmek için gereklidir [1].

Otomat durumu x bazen sistemin fiziksel veya harici durumu olarak da anılmakta ve bu sistemin iç durumunu oluşturan diğer değişkenler $(y_1, \ldots, y_m, N_1, \ldots, N_m)$ ile zıtlık oluşturmaktadır. Bu farkın kesin olarak belirlenmesinin gerektiği hallerde, x süreç durumu, $(x, y_1, \ldots, y_m, N_1, \ldots, N_m)$ ise sistem durumu olacaktır [2].

Bunun sonucu olarak gerçek durum uzayı, sistem için şu şekilde verilmektedir: $X+R^{+m}$ x Z^{+m}, ve burada $x \in X$, $y \in R^{+m}$ ve $N \in Z^{+m}$,dir. Burada Z^{+} negatif olmayan tamsayı serisidir.

Alternatif bir durum uzay modeli, y_i, i=1, …, m olay saatleri yerine, z_i, i = 1, …, m olay yaşlarının kullanılmasına dayanmaktadır. Bu modeli elde etmek (2.4) denkleminin kullanımı ile kolaydır.

Böylece durum denklemi (2.11) şu şekilde değiştirilir.

$$z_i' = \begin{cases} z_i + y^* & i \neq e' \text{ ve } i \in \Gamma(x) \\ 0 & i = e' \text{ ve } i \notin \Gamma(x) \end{cases} \quad i \in \Gamma(x') \qquad \textbf{(2.16)}$$

$$y^* = \min_{i \in \Gamma(x)} \{(\nu_i, N_i - z_i)\} \qquad \textbf{(2.17)}$$

$$e' = \arg\min_{i \in \Gamma(x)} \{(\nu_i, N_i - z_i)\} \qquad \textbf{(2.18)}$$

Geriye kalan durum denklemleri (2.9) ve (2.12) değiştirilmez. (2.16) da, tetikleyen olay e' haricinde, yeni durum x' içerisinde aktif olarak kalan olayların yaşı artırılmalıdır. Ayrıca yeni aktif hale getirilmiş olayların, tetikleyen olay e' dahil olmak üzere, yaşları 0'a ayarlanmalıdır, $e' \in \Gamma(x')$.

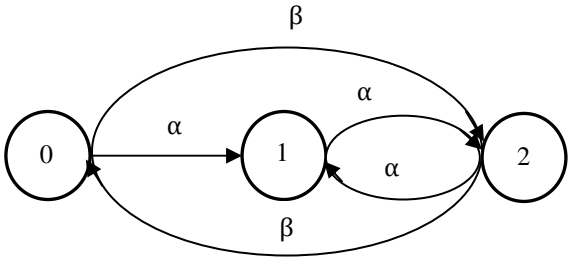
Bu modelde başlangıç koşulları, x_0 bir kere belli olduktan sonra kolay bir şekilde belirlenebilmektedir. Daha önce olduğu gibi başlangıç anında aktif olan bütün $i \in \Gamma(x_0)$ olayları için $N_{i,0}= 1$ ve $i \notin \Gamma(x_0)$ için $N_{i,0} = 0$ olarak belirleniyor. Bunlara ilaveten, bütün $i \in \Gamma(x_0)$ olayları için $z_{i,0}= 1$ olarak bir ayarlama yaparken, $z_{i,0}$, $i \notin \Gamma(x_0)$ olayları için belirsizdir [3].

Örnek 2.1

Şekil 2.5'te bir durum geçiş diyagramı gösterilen otomat için $E = \{\alpha, \beta\}$ ve $X = \{0, 1, 2\}$ bilgileri verilmektedir. Burada eğer dikkat edilirse $\Gamma(0) = \Gamma(2) = \Gamma\{\alpha, \beta\}$ fakat $\Gamma(1) = \{\alpha\}$ olduğu görülür. Başlangıç durumunun $t_0= 0.0$ anında $x_0=0$ olduğu varsayılacaktır.

Bir saat yapısı **V** aşağıda belirtildiği şekilde temin edilmektedir (sadece ilk birkaç olay ömrü gösterilmektedir):

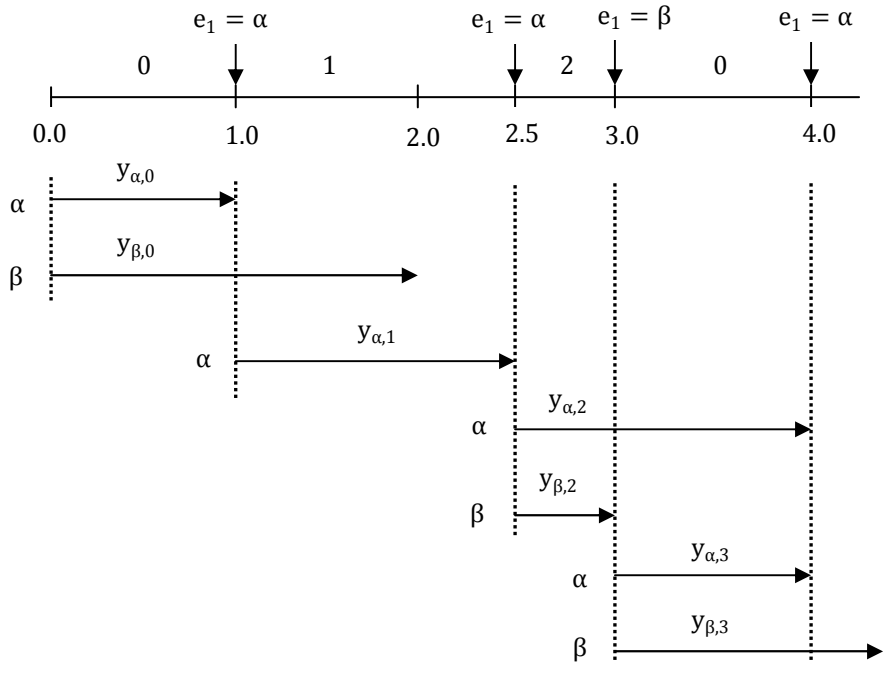
$v_\alpha = \{1.0, 1.5, 1.5, \ldots\}$ $v_\beta = \{2.0, 0.5, 1.5, ...\}$



Şekil 2.5 : Örnek 2.1 zamanlı otomat için durum geçiş diyagramı

Bu zamanlı ayrık olay sisteminin örnek yolu, ilk dört olay için, Şekil 2.6'da gösterilmiştir. Başlangıçta, her iki olayda aktiftir ve bunların saat ile skor değerleri aşağıdaki şekilde ayarlanmıştır.

$y_{\alpha,0} = v_{\alpha,1} = 1.0 \rightarrow N_{\alpha,0}=1,$ $y_{\beta,0} = v_{\beta,1} = 2.0 \rightarrow N_{\beta,0}=1$



Şekil 2.6 : Örnek 2.1 için olay-zaman diyagramı

Şimdi artık diyagram, (2.7)'den (2.12)'ye kadar olan model denklemlerinden herhangi birine gerek kalmadan belirlenebilir. Öncelikle ilk olay ömürleri karşılaştırılır.

$$y_0^* = \min\{y_{\alpha,0},\ y_{\beta,0}\} = y_{\alpha,0} \quad \rightarrow \quad e_1 = \alpha$$

Böylece durum geçiş diyagramına bakmak sureti ile bir sonraki durum aşağıdaki gibidir.

$$x_1 = f^x(0,\alpha) = 1$$

Ayrıca olay zamanı basit bir şekilde $t_1 = y_0^* = 1.0$ olarak verilmektedir.

Bu işlemi $\Gamma(1) = \{\alpha\}$ ile tekrar ederken kaydedilmesi gereken nokta β'nın saat değerinin konu ile ilgisinin olmadığıdır, çünkü β bu durumda gerçeklenebilir değildir.

α tetikleyici olay olması nedeni ile, skorun 1 artırılması ile yeniden aktif hale getirilir. Bu durumda yani saat değeri aşağıdaki gibi olur.

$$y_{\alpha,1} = v_{\alpha,2} = 1.5$$

Burada açıkça belli olan bir sonraki tetikleyici olayın $e_2 = \alpha$ olmasıdır. Öyleyse, sıradaki durum,

$$x_2 = f^x(1,\alpha) = 2$$

ve zaman ileriye doğru $t_2 = t_1 + y_1^* = 2.5$ kadar hareket ettirilir

Bundan sonra $\Gamma(2) = \{\alpha, \beta\}$ ve her iki olayda yeniden aktif hale getirilmiştir. Ayrıca bunların skorları da artırılmıştır. Aşağıdaki şekilde belirtilen ayarlama yapılırsa,

$$y_{\alpha,2} = v_{\alpha,3} = 1.5 \qquad y_{\beta,2} = v_{\beta,2} = 0.5$$

$$y_2^* = \min\{\ y_{\alpha,2}, y_{\beta,2}\ \} = y_{\beta,2} \quad \rightarrow \quad e_3 = \beta$$

Böylece bir sonraki durum aşağıdaki gibi olur.

$$x_3 = f^x(2, \beta) = 0$$

Olayın zamanı ise şekildedir; $t_3 = t_2 + y_2^* = 2.5 + y_{\beta,2} = 3.0$

Şekil 2.6'da gösterilmekte olan en son geçiş 0 durumunda meydana gelir. α olayı hala aktif haldedir ve β tetikleyici olaydır. Sadece β olayının skoru artırılır. Saat değerleri aşağıda belirtilen şekilde güncellenirse;

$$y_{\alpha,3} = y_{\alpha,2} - y_2^* = 1.0, \qquad y_{\beta,3} = v_{\beta,3} = 1.5$$

$$y_3^* = \min\{y_{\alpha,3}, y_{\beta,3}\} = y_{\alpha,3} \quad \rightarrow \quad e_4 = \alpha$$

En son durum aşağıdaki gibi bulunur.

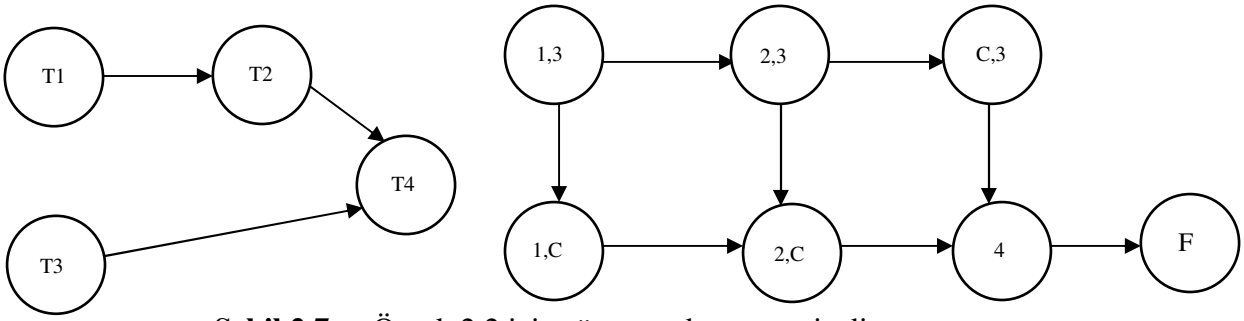
$$x_4 = f^x(0, \alpha) = 1$$

Olayın zamanı ise $t_4 = t_3 + y_3^* = 3.0 + y_{\alpha,3} = 4.0$ olarak belirlenir.

Örnek 2.2

Birçok işlem seri halinde veya aynı zamanda veya her ikisinin bir karışımı olarak gerçekleştirilmesi gereken görevlerden oluşur. Her bir görev gerçekleştirilmek için belli miktarda bir zamana gerek duyar.

Sistemin muhtelif olay ömürleri aracılığı ile modeli çıkartılabilir. Bu, bir ürünün imal edilmesinde, bir algoritmanın bilgisayarda yürütülmesinde veya sadece basit şekilde bir "projenin" gerçekleştirilmesinde de geçerlidir. Bu örnekte ise şu şekilde bir olay örgüsü izlenecektir: Motoru belli bir sıcaklığa gelmesi için çalıştır; bu gerçekleşirken benzin tankını kontrol et; eğer belli bir seviyenin altında ise doldur; daha sonra yağ basıncına bak; eğer çok düşük ise, motoru durdur ve görevden "çık"; aksi takdirde motor ısısını kontrol et ve eğer istenen seviyede ise "ilerle", aksi takdirde söz konusu seviyeye gelene kadar bekle ve sonra "ilerle".

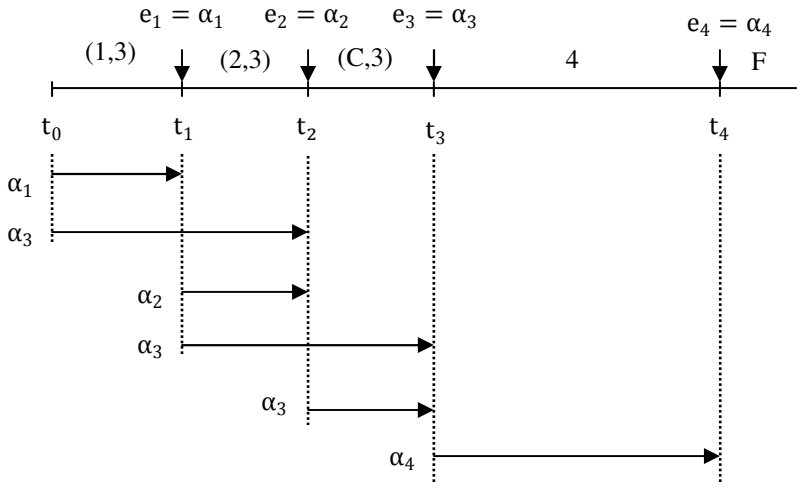


Şekil 2.7 : Örnek 2.2 için süreç ve durum geçiş diyagramı

Şekil 2.7'de birbirini takip eden iki görevden, T1 ve T2, oluşan basit bir süreçten bahsedilmektedir. Bunlar üçüncü bir göreve, T3, paralel olarak gerçekleştirilebiliyorlar. İki paralel işlemcide gerçekleştirilen bilgisayar işlemleri olarak düşünülebilir. T1 ve T3 görevlerinin çıkışı T4'ün tek bir işlemci üzerinde başlamasından önce birleştirilmelidir.

Süreç, durum geçiş diyagramı Şekil 2.7'de gösterilen bir otomat aracılığı ile tarif edilebilir. E = $\{\alpha_i : i=1, \ldots, 4\}$ gibi süreci yönlendiren dört tane olay mevcuttur ve α_i, i görevinin tamamlanmasını göstermektedir. Durum uzayı X, gösterildiği gibi yedi durumdan oluşmaktadır. Durum (1,3), T1 ve T3'ün mevcut durumda aynı anda devam ettiğini göstermektedir. Benzer şekilde durum (2,3) için de aynısı söylenebilir. Diğer taraftan ise (1,C), T1'in devam etmekte olduğunu ve bu zaman aralığında T3'ün tamamlandığını göstermektedir. Benzer olarak (2,C) için de aynısı söylenebilir. Durum (C,3), T3'ün devam etmekte olduğunu fakat bu esnada T1 ve T2 görevlerinin tamamlandığını göstermektedir. Son olarak durum 4, T4'ün devam etmekte olduğunu ve durum F ise sürecin bittiğini gösterirler.

Bu örnekte her bir örnek yol durum (1,3) ile başlamakta ve durum F ile sonuçlanmaktadır. Ayrıca her olay sadece bir kere meydana gelebilmekte ve bu nedenle saat yapısı sadece dört sayılı bir kümeden oluşmaktadır. Olay-zaman diyagramı Şekil 2.8'de gösterilen sistemde, zaman aralığı $[t_2, t_3]$, T2 ve T3 çıkışının ikisinin de mevcut olması gereksiniminden dolayı T4'ün başlamasında bir zaman gecikmesi göstermektedir.



Şekil 2.8: Örnek 2.2'deki süreç için olay-zaman diyagramı

Sonuç olarak söylenebilir ki otomat (X, E, f, Γ, x_0) ve saat yapısı $\mathbf{V}=\{\mathbf{v_i}\ ;\ i \in E\}$ birbirlerinden tamamen ayrıdırlar. Bununla belirtilmek istenen, otomat (X, E, f, Γ, x_0), yürütülmesi için bir olay serisine ihtiyacı olan basit bir sistem ve **V** ise bağımsız olarak oluşturulmuş bir dizi kümesi olduğudur. Böylece **V**, farklı zamanlı otomatlara giriş olarak kullanılabilir. Otomat ve onun saat yapısı arasındaki bağlantı, kendini gerçeklenebilir bir olay seti Γ(x) aracılığı ile gösterir. Buda x'in sadece Γ(x) ile göründüğü (2.11) ve (2.12) denklemlerinde açıkça belirtilmiştir. Bu şekilde, zamanlama mekanizması aşağıdaki forma sahip bir ilişki tarafından oluşturulur.

$$e' = h[y,N,\Gamma(x)]$$

Bu denklemde y ve N yedek durum değişkenleridir ve bunlar aşağıda belirtilen denklem aracılığı ile otomat durumunda güncelleme yapabilmek için gereklidir.

$$x' = f^x(x, e')$$

2.5 Zamanlı Otomat Kuyruk Sistemleri

Basit bir kuyruk sistemi için geliştirilmiş otomat modeli şu şekildedir:

$E = \{a,d\}, \qquad X = \{0,1,2,\ldots\}$

$\Gamma(x) = \{a,d\}$ bütün $x > 0$ için, $\qquad \Gamma(0) = \{a\}$ **(2.19)**

$$f(x,e') = \begin{cases} x+1 & \text{eğer } e' = a \\ x-1 & \text{eğer } e' = d \text{ ve } x > 0 \end{cases}$$

Burada a bir müşteri gelmesi olayıdır ve d ise müşterinin ayrılması olayıdır. Otomat durumu x sıranın uzunluğunu temsil eder (hizmet verilmekte olan müşteri dahil olmak üzere). Burada kaydedilmesi gereken husus $d \notin \Gamma(0)$ olması sebebi ile $f(x,e')$'nin $x = 0$, $e' = d$ için tanımlanmamış olmasıdır [1].

Saat yapısı V verildiğinde, tetikleyici olay, her zaman a ve d olaylarının saat değerlerini mukayese etmek sureti ile elde edilmektedir. Bu duruma tek istisna, sadece bir a olayının gerçeklenebilir olduğu $x = 0$ koşuludur.

Böylece aşağıda belirtilenler geçerli olur.

$$e' = \begin{cases} d & \text{eğer } y_d < y_a \text{ ve } x > 0 \\ a & \text{aksi halde} \end{cases}$$

Bir kere e' belirlendikten sonra olay saati ve skoru durum denklemleri (2.11) ve (2.12) aşağıda belirtildiği şekilde yazılabilir:

$$y_a' = \begin{cases} y_a - y_d & \text{eğer } e' = d \\ v_a, N_a + 1 & \text{aksi halde} \end{cases} \quad \textbf{(2.20)}$$

$$y_d' = \begin{cases} y_d - y_a & \text{eğer } e' = a \text{ ve } x > 0 \\ v_d, N_d + 1 & \text{aksi halde} \end{cases} \quad \textbf{(2.21)}$$

$$N_a' = \begin{cases} N_a + 1 & \text{eğer } e' = a \\ N_a & \text{aksi halde} \end{cases} \quad \textbf{(2.22)}$$

$$N_d' = \begin{cases} N_d + 1 & \text{eğer } [e' = a \text{ ve } x = 0] \text{veya } [e' = d \text{ ve } x > 1] \\ N_d & \text{aksi halde} \end{cases} \quad \textbf{(2.23)}$$

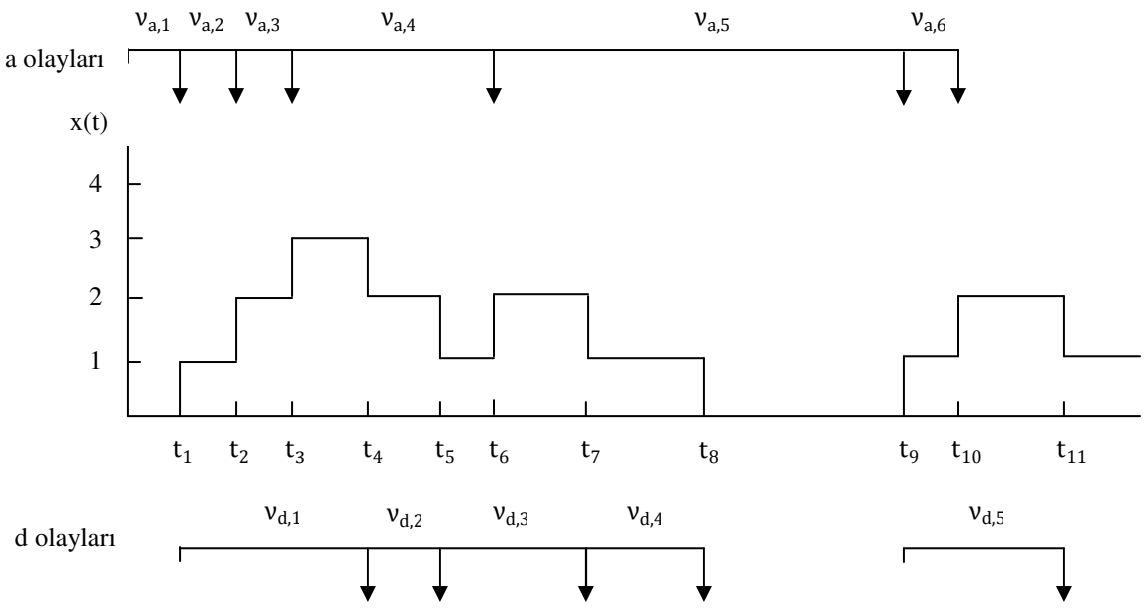
Bu modelin özel yapısı nedeniyle, örnek yollarını, zaman içinde kuyruk uzunluğunun değişimi olarak temsil etmek geleneksel hale gelmiştir. Bir örnek Şekil 2.9'da gösterilmektedir, burada iki belli saat serisi, a ve d verilmektedir. Bütün a olayları pozitif bir birim atlamasına ve bütün d olayları negatif bir birim atlamasına neden olmaktadır. Burada belirtmemiz gereken husus ilk d olayının sadece t_1 zamanında aktif hale getirilmiş olduğudur. Bunun nedeni başlangıçta x = 0 ve $\Gamma(0) = \{a\}$ olmasıdır. Benzer bir gözlem t_{11}'deki beşinci d olayına uygulanır, bu olay t_9'da, boş kuyruk zaman aralığının $[t_8,t_9]$ sonunu takip ederek aktif hale gelmiştir [1].

Burada bir kere daha, aynı kuyruk sistemi için olayın yönlendirdiği (2.19)'dan (2.23)'e kadar olan model ile zamanın yönlendirdiği model arasındaki farkların üzerinde durulması faydalı olabilir. İlk önce, CVDS geleneksel fark denklem formülasyonunu kullanarak, bu sistem için sürekli zaman modeli oluşturmanın zor olduğu gözlemlenebilir.

Bunun sebebi x(t)'nin negatif olmayan ayrık tam sayılar kümesi üzerinden tanımlanmış olmasıdır ve bu nedenle türev $\dot{x}(t)$, kelimenin tam anlamıyla mevcut değildir. Bununla birlikte aşağıda belirtildiği şekilde ayrık bir zaman modeli oluşturulabilir [2].

Burada k = 1, 2,… zaman indeksi olsun. Zaman doğrusu τ, 2τ,…, $k\tau$,… şeklinde zaman bölümlerine ayrılır, öyle ki en fazla bir varış ve bir ayrılma olayı zaman bölümü τ uzunluğu içinde gerçekleşebilir. Böylece, aşağıda belirtildiği şekilde bir giriş serisi a(k) tanımlanabilir:

$$a(k)=\begin{cases} 1 & \text{eğer k. zaman bölümünde bir varış gerçekleşirse} \\ 0 & \text{aksi halde} \end{cases} \qquad k = 1, 2, \ldots$$



Şekil 2.9: Basit bir kuyruk sisteminin olay-zaman diyagramı

Sıra uzunluğu x(t) her varış (a olayı) ile 1 artmakta ve her ayrılış (d olayı) ile 1 azalmaktadır. a olayının aktivasyonları her a meydana gelmesinde olmaktadır. d olayının aktivasyonları ise sadece d meydana geldiğinde ve sıra boş kalmamışsa veya eğer sıra boş ise a olayının meydana gelmesi ile olmaktadır [6].

x(k) için durum denklemi aşağıda belirtilen formda olacaktır.

$$x(k+1)=\begin{cases} x(k)+1 & \text{eğer } a(k)=1 \text{ ve k. zamanda bir ayrılış gerçekleşmemişse} \\ x(k)-1 & \text{eğer } a(k)=0 \text{ ve k. zamanda bir ayrılış gerçekleşmişse} \\ x(k) & \text{aksi halde} \end{cases}$$

Burada "k. zaman bölümünde bir ayrılış olmaktadır" koşulu (2.21) altındaki saat durum değişkeni y_d ile bir benzeşim yapılmasını gerektirmektedir. Bu benzeşimin oluşturulması, ortaya konulmak istenen hususun farklı olması nedeniyle, burada göz ardı edilen ince detayları içermektedir. Özellikle, eğer k. zaman bölümünde herhangi bir olayın vuku bulmuyorsa zaman ile yönlendirilen model değişmemek için x(k)'ya ihtiyaç duyar. Eğer zaman bölümleri, uzunlukları kısa olacak şekilde seçilmiş iseler o zaman bu bölümlerden çoğunluğu bir olayı ihtiva etmeyecektir. Bu potansiyel olarak verimli bir model değildir çünkü sistemde değişikliğin oluşmaması halinde zaman sık sık güncellenmektedir [1].

Bunlara ilaveten a(k)'nın gerçek varış saat serisi v_a ile yaklaşıklık oluşturduğu gözlemlenebilir. Bu nedenle, aynı zaman dilimi içerisinde iki varışın olmaması gereksinimini sağlamak için çok küçük değerde τ kullanmak istenebilir. Bu yakınlaştırmayı daha iyi hale getirebilir, fakat aynı zamanda yukarıda belirtilen verimsizliği de artırır.

2.6 Olay Programlama Planı

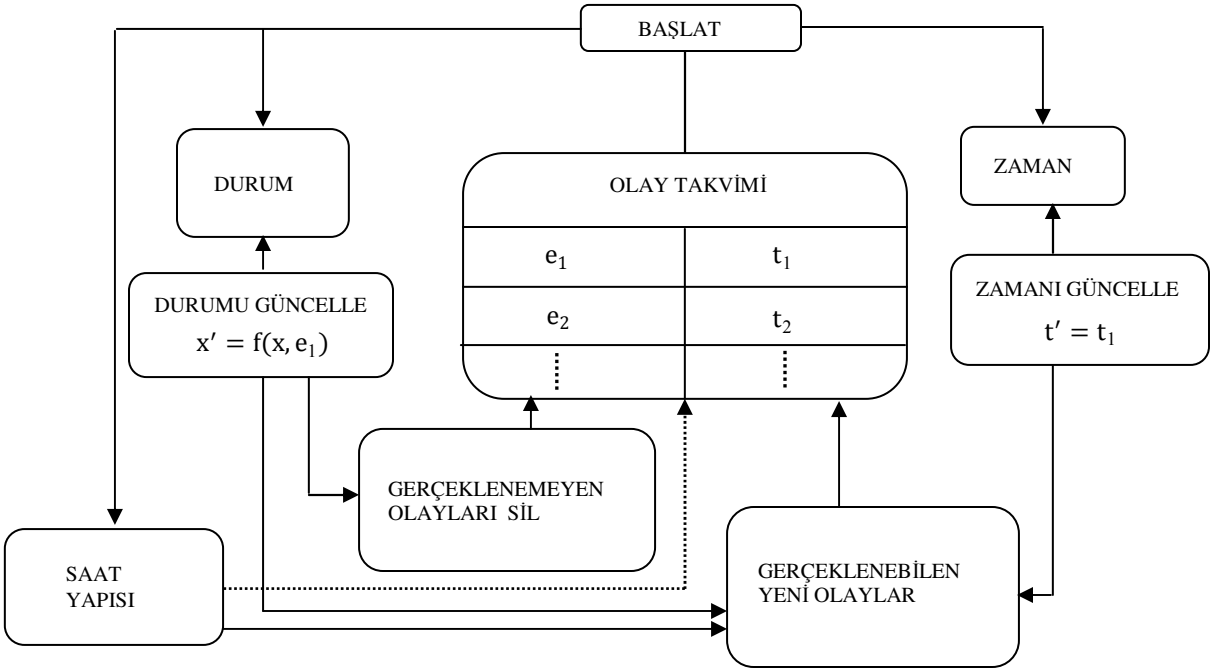
Bu bölümde, bilgisayar uygulaması açısından daha basit olan Şekil 2.4'te özetlenmiş otomat modeline bir varyasyon getirilecektir.

Ne zaman bir i olayı, belli bir t_n olay zamanında, N_i puanı ile aktif hale getirilirse bunun bir sonraki meydana gelmesi (t_n+v_{i,N_i+1}) zamanı olarak programlanmaktadır. Bu şekilde saat değeri y_i yerine, programlanmış bir olay listesi tutulmaktadır.

$$L = \{(e_k, t_k)\}, \qquad k = 1, 2, \ldots, m_L$$

Burada $m_L \leq m$ mevcut durumda geçerli olayların sayısıdır. Buna ilaveten, programlanmış olay listesi her zaman en küçük planlanmış zaman önceliği bazında

sıralanır. Bu da, listedeki ilk olayın e_1 her zaman tetikleyici olay olacağını ima etmektedir.



Şekil 2.10 : Olay programlama planı

Programlanmış olay listesi mevcut durumda ki bütün geçerli olayları içerir ve bunlar en küçük zaman önceliği bazında sıralanmışlardır. Bir sonraki olay her zaman e_1 dir ve bu t_1 anında meydana gelir. Böylece, durum ve zamanda güncellemeler yapılmış olur. Daha sonra durumun yeni değeri bazında bazı olaylar aktif hale getirilirken bazıları da aktif olmaktan çıkarlar. Aktif hale getirilen olaylar, planlanmış meydana gelme zamanları ile birlikte programlanmış olaylar listesine girilir ve bu esnada doğru sıralama düzeni korunur [2].

Şekil 2.4'teki model ile mukayese edildiğinde kavramsal olarak yeni veya farklı hiç bir şey mevcut değildir. Bununla birlikte, programlanmış olay listesi (bazen olay takvimi olarak da değinilir), şimdiye kadar ki hususların merkezinde yer alan olay zamanlama mekanizmasının esasının yakalanabilmesine yardımcı olmaktadır.

Olay programlama planı, verilen saat yapısı ile yönlendirilen modele dayalı örnek yolların oluşturulması için bir prosedür olarak düşünülmelidir. Bu Şekil 2.10'da gösterilmiştir. Durum, verilen bir x_0 değeriyle başlatılmıştır ve zaman genellikle sıfıra

ayarlanmaktadır. Her olay için bir tane olmak üzere saat yapısı ise saat dizilerinin bir kümesidir, Programlanmış olay listesi x_0 içinde geçerli olan bütün olayları ihtiva etmek üzere başlatılmaktadır ve planlanan zamanlar uygun saat dizisi ile temin edilmektedir. Bu ayrıca planlı zamanların artan düzeni içinde de sınıflandırılmıştır. Prosedür aslında bölüm 2.3'de tarif edilen yedi basamak ile benzerdir. Özellikleri ise:

Basamak 1: İlk giriş olan (e_1, t_1) programlanmış olay listesinden çıkartılır.

Basamak 2: Zaman, yeni olay zamanı t_1'e güncellenir.

Basamak 3: Durum, durum geçiş fonksiyonuna göre güncellenir, $x' = f(x,e_1)$.

Basamak 4: Yeni durumda geçerli olmayan olaylara tekabül eden herhangi bir giriş programlanmış olay listesinden çıkartılır, yani tüm $(e_k,t_1) \in L$'yi silinir öyle ki $e_k \notin \Gamma(x')$ olsun.

Basamak 5: Programlanmış olay listesine hali hazırda planlanmamış herhangi bir geçerli olay eklenir. Planlanan olay zamanı, bu tür i için, saat yapısındaki i serisinden yeni bir olay ömrünün mevcut zamana ilave edilmesi ile bulunur.

Basamak 6: Güncel hale getirilmiş programlanmış olay listesi, en küçük planlanmış zaman önceliği bazında yeniden düzenlenir.

Prosedür, yeni düzenlenmiş liste için Basamak 1'den başlamak sureti ile tekrarlar [3].

3. ZAMANLI PETRİ AĞI

3.1 Zamanlı Petri Ağ Tanımı

Zamanlı olmayan ayrık olaylı sistem modellerinde, Petri ağları çok genel bir çerçeve temin eder. Söz konusu bu çerçeve, Petri ağının bir saat yapısı ile donatıldığında zamanlı modellere de genişletilebilir ve zamanlı Petri ağı haline dönüşür. O halde zamanlı otomat her daim zamanlı Petri ağlarından elde edilebilir [1].

(P,T,A,ω,x) işaretli bir Petri ağı olsun. Burada T geçişler (olaylar) kümesi, P ise yerler kümesidir (muhtelif olayların mümkün olması için gerekli koşullar). A kümesi bütün arkları içermektedir. (p_i,t_j), öyle ki p_i, t_j'nin bir giriş yeridir ve (t_j, p_i) için p_i, t_j'nin bir çıkış yeridir. Her arkın pozitif tam sayıdan oluşan bir ağırlığı $\omega(p_i,t_j)$ veya $\omega(t_j, p_i)$ mevcuttur. Ayrıca, t_j geçişi ile ilgili olarak giriş ve çıkış yerleri kümelerini göstermek üzere sırasıyla $I(t_j)$ ve $O(t_j)$ notasyonları kullanılmaktadır. x ise, Petri ağının işaretleme fonksiyonudur ve bir işaretleme, ağ içinde her bir yerde mevcut olan jeton sayısını temsil etmektedir.

Otomat için tarif edilene benzer bir saat yapısı incelenecektir. Buradaki tek fark saat serisi v_j'nin t_j geçişi ile ilişkili olmasıdır. Pozitif gerçek bir sayı olan $v_{j,k}$'nın t_j'e atanmasının anlamı şöyledir: k. zaman içinde t_j geçişinin mümkün kılınması ile hemen ateşleme yapılmaz fakat $v_{j,k}$ tarafından verilen bir gecikme ile yaşanır, söz konusu gecikme esnasında, jetonlar t_j'nin giriş yerlerinde tutulurlar [8].

Ateşleme gecikmelerinin oluşması için bütün geçişlere ihtiyaç yoktur. Bazı geçişler mümkün kılınmalarını müteakip derhal ateşlenirler. Bu nedenle T, T_0 ve T_D olmak suretiyle alt kümelere ayrılır, şöyle ki $T = T_0 \cup T_D$. T_0 her zaman sıfır ateşleme gecikmeleri oluşturan geçişler kümesidir ve T_D ise genel olarak belli bazı ateşleme gecikmeleri yaratan geçişler kümesidir. T_D, zamanlı geçişler olarak adlandırılır.

Tanım. Saat yapısı (veya zamanlama yapısı) işaretli bir Perti ağın (P,T,A,ω,x) zamanlı geçişler kümesi $T_D \leq T$ ile ilgilidir ve bu aynı zamanda olay ömürleri dizilerinin bir kümesidir.

$$\mathbf{V} = \{ \mathbf{v_j} : t_j \in T_D \}$$

$$\mathbf{v_j} = \{v_{j,1}, v_{j,2}, \ldots\},\ t_j \in T_D,\quad v_{j,k} \in \mathbb{R}^+, k = 1,2,\ldots$$

Grafik olarak geçişler, eğer ateşleme gecikmesi yok ise çubuklarla ve zamanlı geçişler ise dikdörtgenler ile temsil edilmektedir. Zamanlı geçiş ile ilgili saat dizisi ise normal olarak dikdörtgenin yanına yazılır.

Tanım. Zamanlı bir Petri ağ altılı veri kümesinden oluşur.

$$(P,T,A,\omega,x,\mathbf{V})$$

Burada (P,T,A,ω,x) işaretli bir Petri ağı, $\mathbf{V} = \{ \mathbf{v_j} : t_j \in T_D \}$ ise bir saat yapısıdır.

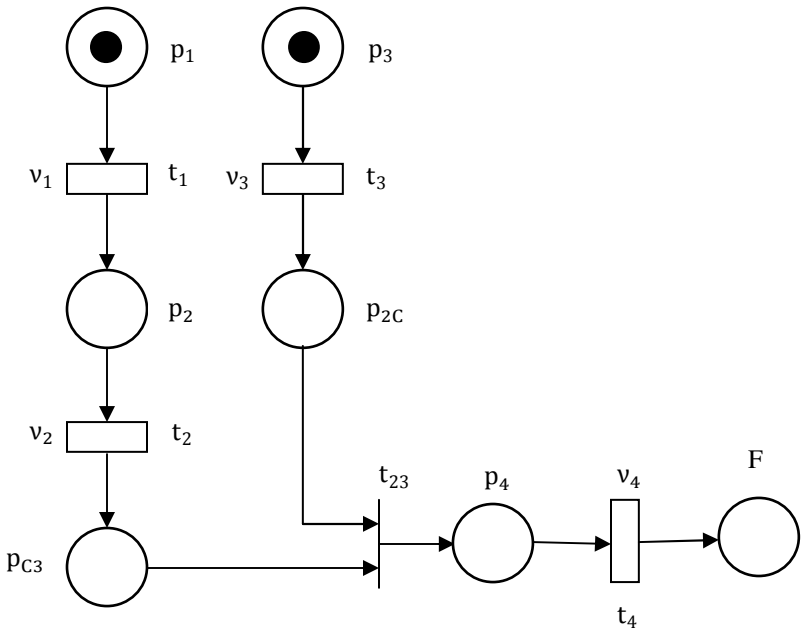
Yorum. Burada bir kere daha bir geçişi mümkün hale getirmek ile ateşlemek arasındaki farkın önemi belirtilmelidir. Bir zamanlı geçiş bütün giriş yerleri yeterli sayıda jetona sahip olduğu zaman hemen mümkün kılınır; fakat bir ateşleme gecikmesinden sonra ateşleme yapılır (bu da saat kümesinin bir elemanı tarafından temin edilmektedir) [7].

Örnek 3.1

Örnek 2.2'deki sürece geri dönülürse, söz konusu örnekte T_1 ve T_2 görevleri üçüncü bir görev olan T_3 ile birlikte aynı zamanda gerçekleştirilmekte ve daha sonra dördüncü bir görev T_4, T_2 ve T_3'ün çıkışını birleştirmek sureti ile gerçekleştirilmekteydi. Bu modelin olası bir Petri ağı Şekil 3.1'de gösterilmektedir.

Zamanlı geçişler $t_1,\ldots, t_4$, görev tamamlama olaylarına tekabül etmektedirler ve bunlar dikdörtgenler ile gösterilirken ateşleme gecikmeleri $v_1,\ldots, v_4$'te bunlara eşlik etmektedir. Görev 2 yapıldığı zaman bu p_{C3} yerine bir jeton ilave etmektedir (yani T_2'nin bittiğini fakat T_3'ün hala devam etmekte olabileceğini belirtmektedir). Benzer şekilde, p_{2C} içinde aynısı söylenebilir. Bu şekilde zamanlı olmayan geçiş t_{23} hem görev 2 ve hem de görev 3 tamamlandığı zaman mümkün kılınmaktadır. Burada dikkat

edilmesi gereken husus t_{23} ve p_4 göz ardı edildiğinde p_{C3} ve p_{2C} doğrudan t_4 geçisini mümkün kılabilirler. Süreç, F'e bir jeton ilave edildiğinde sona erer. Şekil 3.1'de Petri ağı başlangıç konumunda görevler 1 ve 3 devam ederken gösterilmektedir.



Şekil 3.1: Örnek 3.1 için zamanlı Petri ağı

3.2 Zamanlı Petri Ağ Dinamiği

(2.7)'den (2.12)'ye kadar olan modelin geliştirilmesinde görüldüğü şekilde ilerlemek sureti ile Petri ağın geçiş ateşleme zaman dinamiği üzerine dayanan bir model elde edilebilir. Burada olaylar, geçişler ile yer değiştirecektir. Bununla birlikte modeli, bireysel geçişlerin (olayların) dinamiklerine ayrıştırmak sureti ile Petri ağ yapısı etkin olarak kullanılabilir. Diğer bir deyişle, durum denklemleri, söz konusu türde bir modelde aşağıda belirtilen formun geçiş ateşleme kümesini oluştururlar [1].

$$\{\tau_{j,1}, \tau_{j,2}, \ldots\}, \qquad j = 1, \ldots, m$$

Burada $\tau_{j,k}$, t_j geçişinin k = 1,2,… için k. ateşleme zamanıdır. Bu Şekil 3.2'de gösterilmektedir. m geçişi ateşleme kümesi çıkışı, Şekil 2.4 altında verilen zamanlı otomat modeline karşılık gelmektedir, burada çıkışa tek zamanlı olay serisi olarak bakılmaktadır $\{(e_1,t_1), (e_2,t_2), \ldots.\}$.

Şekil 3.2: Bir DES'in zamanlı Petri ağ modeli

Zamanlı olmayan Petri ağlarda geçiş ateşleme zamanları için genel amaçlı durum denklemleri elde etme oldukça karmaşık bir hal alabilir. Fakat belirli Petri ağ sınıflarında (mesela, bir sonraki bölümde dikkate alınan basit kuyruk sistemi gibi) geçiş ateşleme zaman dinamiği için bir model elde etmek oldukça basit olabilmektedir [7].

Takip eden özelliklere sahip Petri ağları dikkate alındığında her bir yer için sadece bir giriş ve bir çıkış geçişi mevcuttur. Buna ilaveten bütün ark ağırlıkları 1'dir. Bu tür bir Petri ağ işaretli graf olarak bilinmektedir. İşaretlenmiş graf tanımının bazı etkileri analiz edilmelidir. Bunun için ilk önce yer p_i'nin sadece bir giriş geçişi t_r'a sahip olduğu üzerine odaklanılmalıdır [8].

$\pi_{i,k}$: p_i yerinin, k. jetonunu aldığı andaki zaman, k = 1,2,…

Başlangıçta $x(p_i) = 0$ olduğunu varsayarak k. zamanda bir jeton p_i içine konulmaktadır ve bu da $\tau_{r,k}$ ile gösterilen t_r'nin k. ateşleme zamanıdır. Diğer taraftan p_i başlangıçta x_{i0} kadar jetona sahip ise, bu takdirde t_r'nin k. ateşleme zamanı p_i'nin $(x_{i0}+ k)$. jetonu aldığı zamandır. Bu nedenle aşağıdaki ilişki verilebilir:

$$\pi_{i,k+x_{i0}} = \tau_{r,k} \quad p_i \in O(t_r), \quad k= 1,2,\ldots \qquad \textbf{(3.1)}$$

Burada x_{i0}, p_i yerinin başlangıç işaretidir. Ayrıca bütün k = 1, …, x_{i0} için, $\pi_{i,k} = 0$'dır.

$$\pi_{i,k} = \tau_{r,k-x_{i0}} \quad p_i \in O(t_r), \quad k = x_{i0}+1, x_{i0}+2,\ldots \qquad \textbf{(3.2)}$$

Bunu takiben bir p_i yerinin sadece bir çıkış geçişi t_j olduğu gerçeği üzerine odaklanılmalıdır. t_j'nin sadece bir giriş yeri p_i olduğunu varsayarak eğer t_j zamanlı değil ise o zaman t_j'nin k. ateşleme zamanı kesin olarak p_i'ın k. jetonunu aldığı zamandır ve bu nedenle t_j'i mümkün kılar.

Bu şekilde aşağıda belirtilen basit ilişkiye sahip olunur:

$$\tau_{j,k} = \pi_{i,k} \qquad k = 1,2,\ldots$$

Diğer taraftan eğer t_j bir saat serisi v_j ile zamanlandırıldı ise o zaman bu ilişki aşağıda belirtildiği gibi olur.

$$\tau_{j,k} = \pi_{i,k} + v_{j,k} \qquad k = 1,2,\ldots$$

Bu durumda eğer p_i, t_j in tek giriş yeri değil ise o takdirde t_j, set içerisinde son giriş yeri $I(t_j)$ k. jetonunu aldığı zaman, k. kere mümkün kılınır, yani bazen $\pi_{s,k}$, $p_s \in I(t_j)$ aşağıdaki şekilde olur.

$$\pi_{s,k} \geq \pi_{i,k} \qquad \text{bütün } p_i \in I(t_j) \text{ için}$$

Bu gerçek aşağıda belirtildiği şekilde ifade edilebilir:

$$\tau_{j,k} = \max_{p_i \in I(t_j)} \{\pi_{i,k}\} + v_{j,k} \quad k = 1, 2, \ldots \qquad \textbf{(3.3)}$$

Özet olarak (3.1)'ten (3.3)'ya kadar olan denklemler, bu sınıftaki Petri ağların geçiş ateşleme zamanlarını belirlemek için bir set tekrarlanan denklem temin etmektedir.

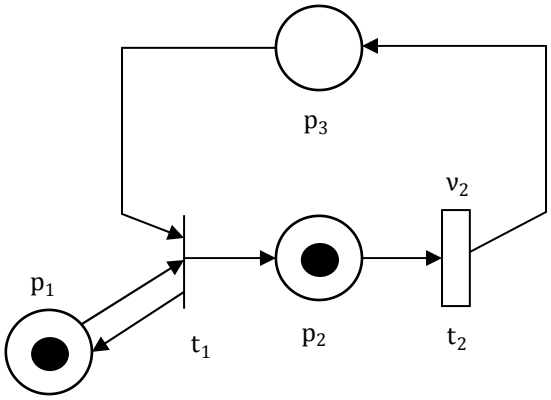
Örnek 3.2

Şekil 3.3'deki Petri ağı dikkate alındığında, bunun işaretli bir graf olduğu görülür. Çünkü her bir yer, bir giriş ve bir çıkış ark ağırlığına sahiptir. Başlangıç durumu [1,1,0]'dir. Bu nedenle (3.1)'ten (3.3)'ya kadar bir set tekrarlanan denklem elde edilebilir.

İnceleme ile birlikte aşağıda belirtilen denklemler $\tau_{1,k}$ ve $\tau_{2,k}$, k = 1,2,… için yazılabilir:

$$\tau_{1,k} = \max\{\pi_{1,k}, \pi_{3,k}\} \qquad \textbf{(3.4)}$$

$$\tau_{2,k} = \pi_{2,k} + v_{2,k} \qquad \textbf{(3.5)}$$



Şekil 3.3: Örnek 3.2 için zamanlı Petri ağ

Geçiş t_1 zamanlı olmadığından son giriş yerleri $\{p_1, p_3\}$ bir jeton aldığında hemen ateşlenir. Geçiş t_2 zamanlıdır böylece p_2 bir jeton aldıktan sonra $v_{2,k}$ gecikmesini takiben ateşlenir. Ağ incelendiğinde şunlar görülür:

$$\pi_{1,k} = \tau_{1,k-1}\ , \quad k = 2, 3,\ldots \qquad \pi_{1,1} = 0 \tag{3.6}$$

p_1 başlangıçta bir jeton içerir ve bunu bir daha elde edebilmek için t_1 in bir sonraki ateşlemesini beklemesi gerekmektedir.

Benzer şekilde,

$$\pi_{2,k} = \tau_{1,k-1}, \quad k = 2, 3,\ldots \qquad \pi_{2,1} = 0 \tag{3.7}$$

Nihai olarak p_3 ne zaman t_2 ateşlese bir jeton almaktadır ve başladığında hiç bir jetonu olmaması sebebi ile aşağıda belirtilen denklem geçerli olur:

$$\pi_{3,k} = \tau_{2,k}, \qquad k = 1, 2,\ldots \tag{3.8}$$

(3.4)'den (3.8)'e kadar olan denklemleri birleştirmek sureti ile $\pi_{1,k}$, $\pi_{2,k}$, $\pi_{3,k}$ göz ardı edilebilir ve k = 1,2,… için aşağıda belirtilenler elde edilir:

$$\tau_{1,k} = \max\{\tau_{1,k-1}, \tau_{1,k-1} + v_{2,k}\}$$

$$= \tau_{1,k-1} + v_{2,k} \tag{3.9}$$

$$\tau_{2,k} = \tau_{1,k-1} + v_{2,k} \tag{3.10}$$

Burada her iki geçişte tekrarlayan bir şekilde t_2'deki ateşleme gecikmesini beklemektedirler. Ayrıca t_1, t_2'nin ateşlenmesi ile ateşlendiği için bütün k = 1,2,… için $\tau_{1,k} = \tau_{2,k}$ olur.

Genel olan Petri ağlar için (3.1)'ten (3.3)'ya kadar olan formda, geçiş ateşleme zamanları ve yer etkin kılma zamanları için denklemler elde etmek mümkündür.

3.3 Zamanlı Petri Ağı Kuyruk Sistemleri

Yerler kümesi P = {Q, I, B} olan basit bir kuyruk sisteminin Petri ağ modelininin zamanlı versiyonu durum x = [0,1,0] için Şekil 3.4'te verilmektedir. Burada ilk olarak sıra boş ve sunucu hareketsizdir. Bu durumda, zamanlı geçiş seti T_D= {a, d} ve bu sunucuya müşteri varışlarına ve sunucudan müşteri ayrılışlarına tekabül etmektedir. Diğer taraftan ise geçiş s, hiç bir ateşleme gecikmesi oluşturmamaktadır: hizmet, sunucunun boş hale gelmesi ve sırada bir müşteri olması ile başlamaktadır [1].

Bu model için saat yapısı şu şekildedir $v_a = \{v_{a,1}, v_{a,2},\ldots\}$ ve $v_d = \{v_{d,1}, v_{d,2},\ldots\}$. Bu daha önce verilmiş olan zamanlı otomat modeli için olan saat yapısı ile aynıdır.

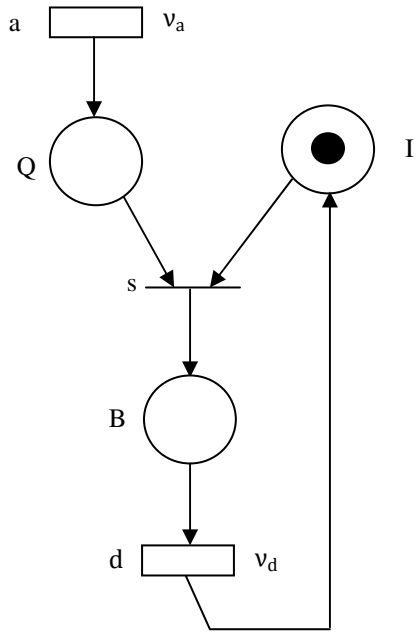
Bunlara ilaveten, eğer aşağıdaki mümkün kılınırsa,

$$x(t) = x(Q) + x(B)$$

bu şekilde Q ve B yerlerinde t zamanında bulunan toplam jetonların sayısı göstermiş olunur.

Şekil 3.4'e bakarak, modelimizin işaretli bir grafın gereksinimlerini yerine getirdiğini görebiliriz. Bu nedenle, (3.1)'ten (3.3)'ya kadar olan formda derhal denklemler elde edebilir ve bu şekilde basit bir kuyruk sisteminin geçiş ateşleme dinamiğini tarif edebiliriz. Aşağıdakileri başlangıç durumu $\pi_{I,1} = 0$ için varsayalım.

a_k : k. varış zamanı
d_k : k. ayrılma zamanı
s_k : k. hizmet başlama zamanı
$\pi_{Q,k}$: Q, k. jetonunu aldığı zaman
$\pi_{I,k}$: I, k. jetonunu aldığı zaman
$\pi_{B,k}$: B, k. jetonunu aldığı zaman



Şekil 3.4: Basit bir kuyruk sisteminin zamanlı Petri ağ modeli

(3.3)'den veya doğrudan yapılan bir inceleme ile aşağıdakilere ulaşılır:

$$a_k = a_{k-1} + v_{a,k}, \qquad k = 1,2,\ldots \qquad a_0 = 0 \qquad \textbf{(3.11)}$$

$$s_k = \max\{\pi_{Q,k} + \pi_{I,k}\}, \qquad k = 1,2,\ldots \qquad \textbf{(3.12)}$$

$$d_k = \pi_{B,k} + v_{d,k}, \qquad k = 1,2,\ldots \qquad \textbf{(3.13)}$$

$$\pi_{Q,k} = a_k, \qquad k = 1,2,\ldots \qquad \textbf{(3.14)}$$

$$\pi_{I,k} = d_{k-1}, \qquad k = 2,3,\ldots \qquad \pi_{I,1} = 0 \qquad \textbf{(3.15)}$$

$$\pi_{B,k} = s_k, \qquad k = 1,2,\ldots \qquad \textbf{(3.16)}$$

(3.12)'den (3.16)'a kadar olan denklemleri birleştirmek sureti ile $\pi_{Q,k}$, $\pi_{I,k}$ ve $\pi_{B,k}$ ortadan kaldırılabilir ve bu durumda aşağıda belirtilen denklemler elde edilir [7].

$$s_k = \max\{a_k, d_{k-1}\}, \qquad k = 1,2,\ldots \qquad d_0 = 0 \qquad \textbf{(3.17)}$$

$$d_k = s_k + v_{d,k}, \qquad k = 1,2,\ldots \qquad \textbf{(3.18)}$$

s_k (3.18)'den çıkartılabilir ve bu şekilde söz konusu basit kuyruk sistemi için, aşağıda belirtildiği şekilde önemli temel bir ilişki elde edilir.

$$d_k = \max\{ a_k, d_{k-1} \} + v_{d,k} , \qquad k = 1,2,\ldots \qquad d_0 = 0 \qquad \textbf{(3.19)}$$

Bu basit bir tekrarlayan ilişkidir ve müşteri ayrılma vakitlerini karakterize etmektedir. Bahse konu olan ilişki k. ayrılmasının $v_{d,k}$ zaman biriminde (k-1). ayrılmasından sonra meydana geldiği gerçeğini dikkate almaktadır, bu duruma $a_k > d_{k-1}$ hali istisna teşkil etmektedir. Sonraki durum, d_{k-1} ayrılanın sırayı boş bırakması ile meydana gelmektedir; bunu müteakip sunucunun a_k deki bir sonraki varışı beklemesi ve bir sonraki ayrılışı ise $a_k + v_{d,k}$ oluşturması gerekmektedir [8].

(3.11) ve (3.19) denklemlerini yeniden yazmak sureti ile k = 2,3,… için aşağıda belirtilenler elde edilir:

$$a_k = a_{k-1} + v_{a,k} , \qquad a_0 = 0 \qquad \textbf{(3.20)}$$

$$d_k = \max\{a_{k-1} + v_{a,k}, d_{k-1} \} + v_{d,k} , \qquad d_0 = 0 \qquad \textbf{(3.21)}$$

Denklemleri yazılarak Şekil 3.2'nin çerçevesinde durum uzay modeli elde edilir. Kuyruk sistem modeli, saat serileri v_a ve v_d tarafından yönlendirilmektedir ve çıkışı, varış ve ayrılma zaman dizilerinden $\{ a_1, a_2, \ldots\}$ ve $\{d_1, d_2, \ldots\}$ oluşmaktadır. Ayrıca bunlar durum denklemleri (3.11) ve (3.19) ile oluşturulmaktadır [3].

4. OTOYOL GİRİŞ DENETİMİ ZAMANLI PETRİ AĞI UYGULAMASI

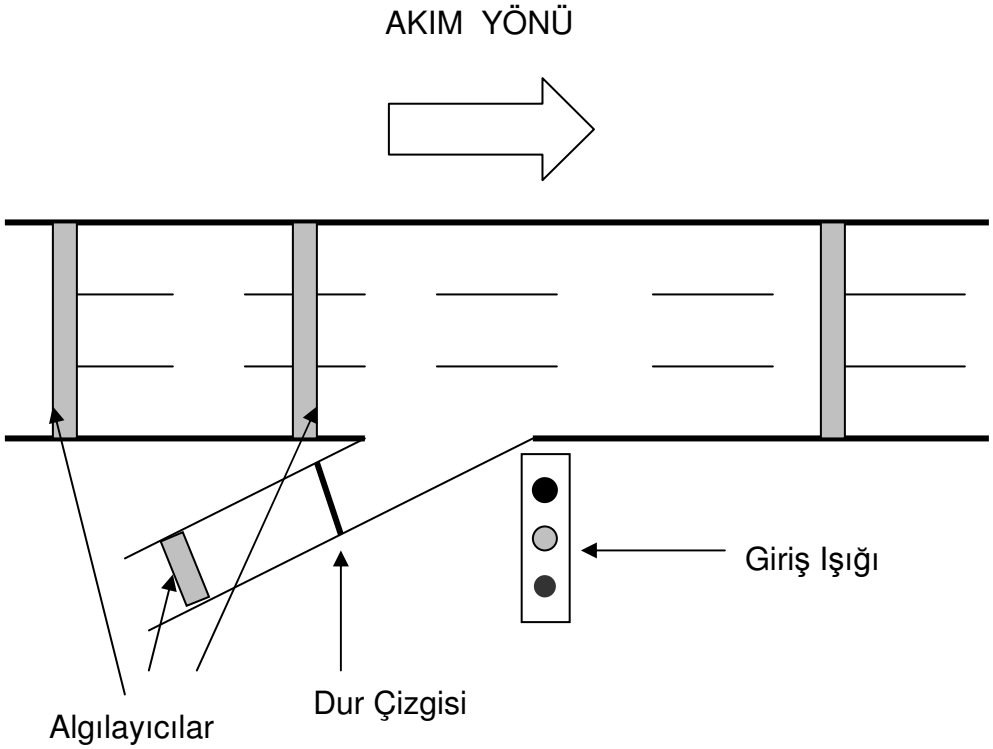
4.1 Otoyol Giriş Denetimi

Şehir içi caddelerde trafik akımı (trafik ışıkları vasıtasıyla) gruplandırılarak yönetilir. Şehir içi otoyollarda ise gruplanma istenmez. Çünkü oluşacak hız farkları altyapı kapasitesinin kullanımını azaltır. Altyapı kapasitesini en uygun şekilde kullanacağından otoyol akımında kararlı akım şartları hüküm sürmelidir.

Giriş denetimi yeni bir trafik yönetim kavramı değildir; kırk yıldır kullanılan etkin bir denetim aracıdır. Giriş denetiminin basit özeti, otoyol girişine yerleştirilecek trafik ışığı sayesinde girişteki araçların otoyola kontrollü katılmalarının sağlanmasıdır. Otoyol giriş denetimi, kapasite paylaşım yöntemlerinden birisidir. Uygulama, otoyol girişlerinde araçların otoyola erişimini denetleyerek akıcı olan otoyola giren araçları sınırlayarak, giren trafik hacmini denetleyerek ve giren taşıtları aralıklara bölerek otoyolun kapasitesinin verimli kullanılmasını sağlar. Giriş denetimi uygulayarak, dur kalk gecikmelerinin azaltılması, planlı güzergah değişimlerinin özendirilmesi ve altyapının istenen hizmet düzeyine ulaştırılması amaçlanır. Uygulamanın başarısı, anayol akımının hızını düşürmeden, akıma katılabilecek en yüksek hacimden sonra otoyol girişi hacimlerini sınırlandırmasına, düzenlemesine ve taşıtların uygun zamanlamayla katılmalarına izin vermesine bağlıdır [4].

Giriş denetim uygulaması, “ ilk gelen ilk hizmet alır” (FIFO: first in first out) prensibine dayanarak otoyolun artık kapasitesini girişteki araçlara ayırmaktır. Girişe gelen araçlar belirli bir süre veya anayol akımı belli bir hacim değerine düşünceye kadar bekletilirler. Artık kapasite anayoldaki araçların çıkışlardan anayolu terk etmeleri ile gerçekleşir.

Giriş denetimi için otoyol girişinde trafik ışığı, giriş yolu ile otoyolda algılayıcılar ve denetim donanımı gereklidir (Şekil 4.1). Otoyol algılayıcılarından gelen bilgi, denetim donanımı ile işlenir, otoyolun artık kapasitesi bulunur, trafik ışığı ve girişteki algılayıcı sayesinde artık kapasite varsa, giriş yolundan aracın anayola katılmasına izin verilir [4].



Şekil 4.1 : Denetim için algılayıcıların yerleri

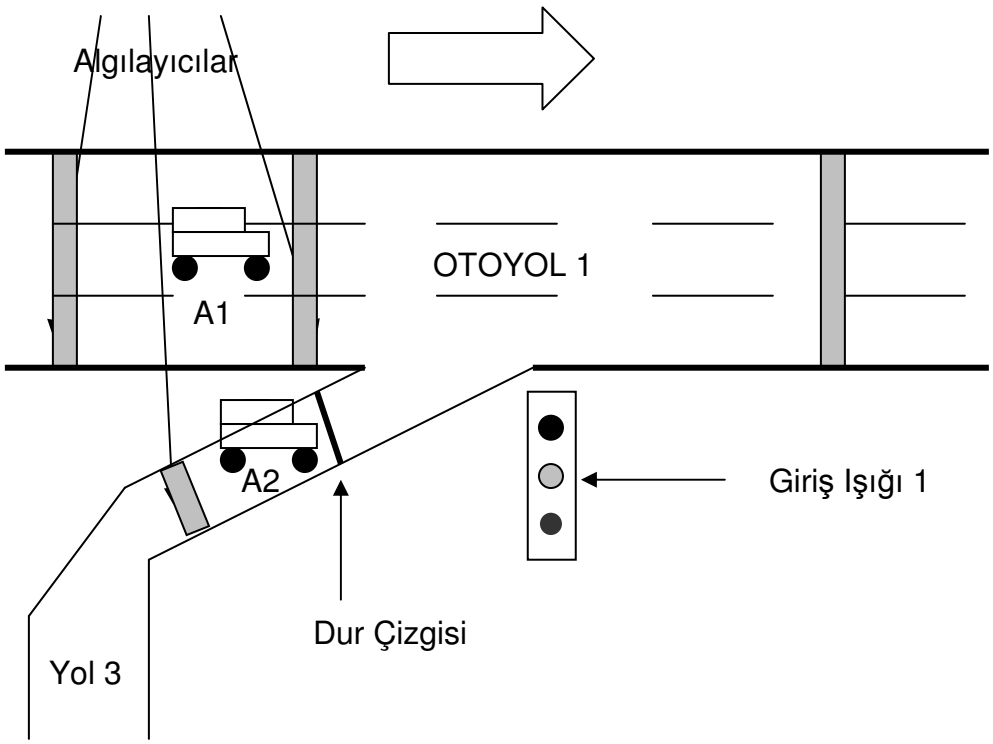
Basit bir giriş denetimi için üç adet algılayıcı gereklidir: Anayol sağ şeritte aralık algılayıcısı, giriş yolunda talep algılayıcısı ve otoyol ana algılayıcısı. Aralık algılayıcısı, anayol en sağ şeridinde taşıtlar arasındaki aralıkları ölçen ve otoyol trafik ışığı için gerekli bilgiyi aktaran ana algılayıcıdır. Talep algılayıcısı, giriş yolunda dur çizgisi arkasında bekleyen araç sayısını belirleyen algılayıcıdır [4].

Öncelikle hesaplanmış otoyol kapasitesi ile anayol algılayıcısı sayesinde ölçülen yukarı akım hacminin farkı (artı kapasite) bulunur. Artık kapasite aynı zamanda otoyola katılabilecek araç sayısıdır ve belirlenecek zaman aralıkları boyunca güncellenir. Aralık algılayıcısı sayesinde araçlar arası uygun boşluk bulunur.

Bu boşluk için hesaplanan araç sayısına uygun yeşil ışık girişte yakılır ve denetim hacminin (otoyol girişinde denetimli erişimi sağlanan akımın) akıma katılması sağlanır.

4.2 Zamanlı Petri Ağının Modellenmesi

4.2.1 Amaçlananlar



Şekil 4.2 : Petri ağı uygulamasının senaryo şekli

Şekil 4.2'de görüldüğü üzere Otoyol 1 ve yan yol olan Yol 3 akan araçlar için zamanlı bir petri ağı uygulaması gerçekleştirilmiştir. Bu uygulamada amaçlananlar şudur: Giriş Işığı 1'in yeşil yanma durumunu Otoyol 1 ve Yol 3'ün yoğunluğuna göre ayarlayarak A2 araçlarının otoyola girişini zamanlı Petri ağı ile denetlemek [4].

4.2.2 Kurallar

Yeşil ışığın yanma süresinin hesaplanması için algılayıcılar Otoyol 1'in ve Yol 3'ün akış yoğunluğunu kontrol merkezine bildirdiğini ve sayısal sınır değerlerine göre üç ayrı 'sözel değer'in ortaya çıktığını varsayıyoruz. Bu varsayımla birlikte burada bir tür bulanık mantık kullanmış oluyoruz [5].

Buna göre ;

Akış Yoğunluğu → '**Az**', '**Orta**', ya da '**Çok**' Yoğunluktadır.

Buna karşılık Giriş Işığında üç ayrı yeşil yanma süresi olduğunu varsayıyoruz. Bunlar :

Yeşil ışık süresi→ '**Az**' =(Örneğin 30sn),

'**Orta**'=(Örneğin 60sn.),

'**Çok**'=(Örn.90sn.)

Yukarıdaki varsayımları 1. ve 2. şıklarda tasarlanmasını amaçlanan Petri ağında kullanabilmek için Tablo 4.1 ve 4.2'deki kurallar geliştirilmiştir. Buradaki bulanık mantık içeren kurallar tahmini olarak uzmana danışılmadan örnekte kullanılması için yazılmıştır. Otoyolların, akışın durumuna göre uzman bilgisi dahilinde değiştirilebilir. (Örn. "Y1=A ve Y3 =O ise Ç" iken "Y1=A ve Y3=O ise O" kuralı yazılabilir) [5]

Tablo 4.1 : Yeşil ışık süresini ayarlama kuralları

Y_1 VE	Y_3 İSE	I_1
A	A	O
A	O	Ç
A	Ç	Ç
O	A	A
O	O	O
O	Ç	Ç
Ç	A	A
Ç	O	O
Ç	Ç	O

Tablo 4.1'deki sembollerin anlamları şunlardır:

Y_1 = Otoyol1'in araç akış yoğunluğu,

Y_3 = Yol3'ün akış yoğunluğu,

I_1 = Giriş ışığı 1'in yeşil ışık yanma durumu

A = Az süreli (ışık için), Az yoğun (yol için)

O = Orta süreli (ışık için), Orta yoğun (yol için)

Ç = Çok süreli (ışık için), Çok yoğun (yol için)

Örneğin Tablo 4.1'deki ilk satırı açıklarsak:

Y_1 = A ve Y_3= A ise I_1 = O'dur. Yani, Otoyol 1'in akış yoğunluğu AZ iken Yol 3'ün akış yoğunluğu da AZ ise o halde Giriş Işığı 1'in yeşil ışık Yanma süresi ORTA (=60sn) dir [4].

4.2.3 Amaçlanan Petri Ağının Tasarımı ve Analizi

4.2.1'deki amaçlara uygun olarak 4.2.2'deki kurallar kullanılarak trafik ışığı ilgili olmak üzere bir adet petri ağı tasarlanmıştır. Bu petri ağının tasarımında ve analizinde ;

- Simulaworks *

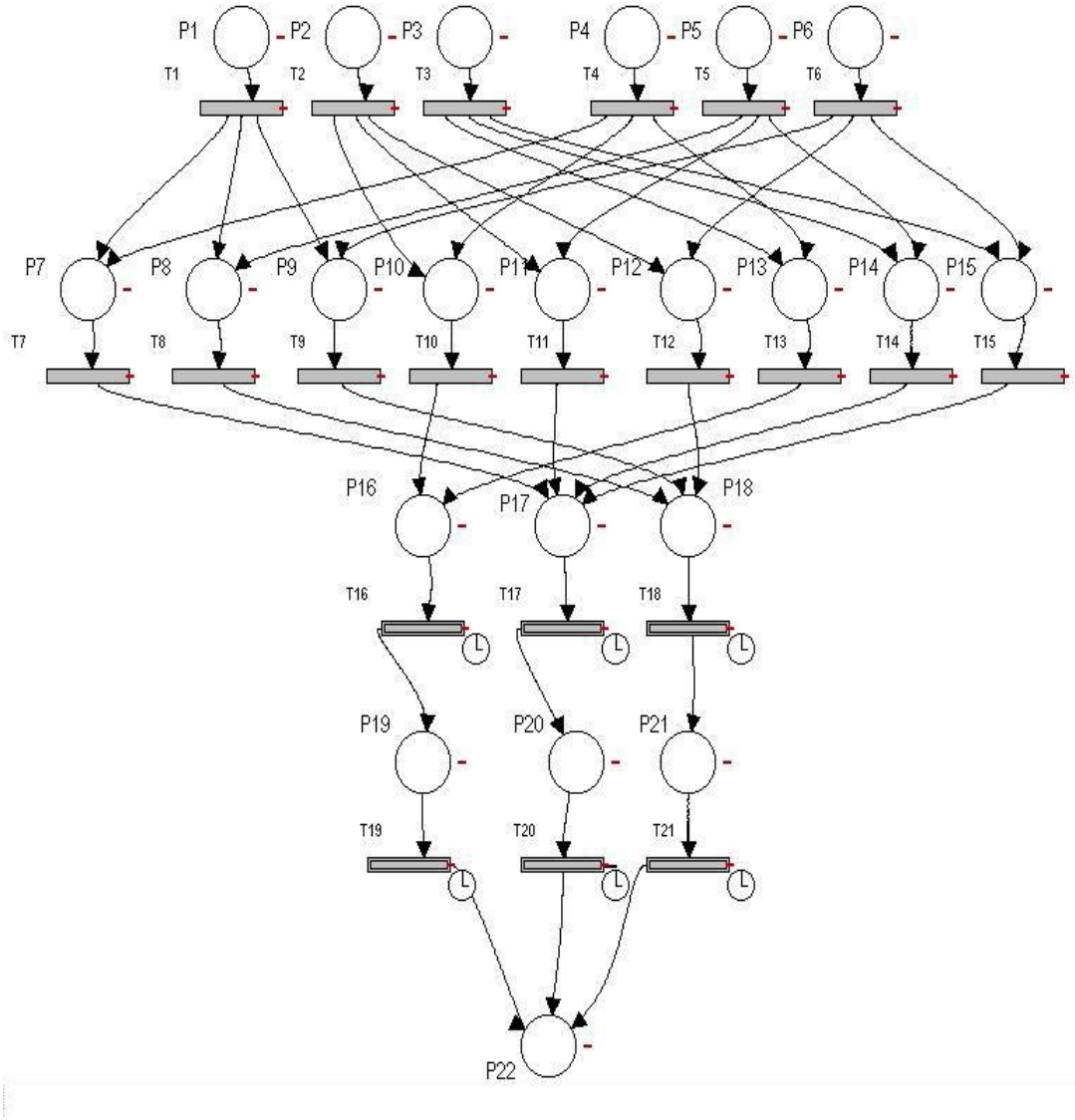
programı kullanılmıştır.

Trafik Işığı Petri Ağı

Tablo 4.1'deki kurallara göre Simulaworks programında hazırlanan Petri ağı modeli Şekil 4.3'de gösterilmiştir. Bu petri ağında toplam 22 tane (P1,...,P22) yer ve 21 tane geçiş (T1,...,T21) vardır.

Bu Petri ağında P1, P2, P3 yerleri Yol 1'in sırasıyla AZ, ORTA, ÇOK yoğunlukta olduğu değerleri gösteren yerler, P4, P5, P6 'da sırasıyla Yol 3' ün AZ, ORTA, ÇOK yoğunlukta olduğu değerleri gösteren yerlerdir. Tablo 4.1'deki kurallara göre sırasıyla P7, P8, P9, P10, P11, P12, P12, P14, P15 yerleri oluşturulur. Bu yerlerden çıkan okların ağırlığı 2'dir. Yani bu yerler, P1, P2, P3'den birinden bir jeton P4, P5, P6'dan birinden bir jeton olmak üzere iki jeton geldiğinde ilgili geçişlerini ateşleyebilirler [4].

İlgili geçişler ateşlendiğinde keza başlangıç durumuna göre her seferinde sadece tek bir yer iki jeton kazanabileceği için P16, P17, P18 yerlerinden yalnız biri jeton kazanır.



Şekil 4.3 : Trafik ışığı için zamanlı Petri ağı

P16, P17, P18 yerleri trafik ışığının yeşil yanmasının sırasıyla AZ, ORTA ya da ÇOK olarak yanması olaylarına karşılık gelir. P16, P17, P18 yerlerinden birisi (keza her seferinde sadece biri jeton kazanabilir) jeton kazandığında, örneğin başlangıç durumunda sadece P1 ve P4 yerlerinde jeton var ise ilgili geçişlerin ateşlenmesi sonucunda P17 yeri jeton kazandığında, Petri ağının zamanlı Petri ağı kısmına geçilmiş olur.

Burada T16, T17, T18, T19, T20, T21 yerleri zamanlı geçişlerdir yani bu geçişler oklardan kendilerine yeterli miktar jeton geldiğinde anında ateşlenmezler ayarlanan belli bir süre sonrasında otomatik olarak ateşlenirler.

Tablo 4.2 : Trafik Işığı için kurallara göre yerler

	P1 veya P2 veya P3	P4 veya P5 veya P6		P16 veya P17 veya P18
P7→	A	A		O
P8→	A	O		Ç
P9→	A	Ç		Ç
P10→	O	A		A
P11→	O	O		O
P12→	O	Ç		Ç
P13→	Ç	A		A
P14→	Ç	O		O
P15→	Ç	Ç		O

Burada yeşil ışık süresini 90 saniyelik bir periyot olarak düşünmekteyiz. Aynı zamanda Yol 1 ve Yol 3'ün akış yoğunluğu değerlerinin her 90 saniyede bir yenilendiğini ve de P16–P21 arasındaki geçişlerdeki gecikmeler haricinde Petri ağında hiçbir gecikme olmadığını varsaymaktayız.

Yeşil ışığın ardından sarı ışık olmaksızın direk kırmızı ışığın yandığını ve ilk olarak yeşil ışığın yandığını varsayarsak 120 saniyelik örnek bir periyot (bu periyot otoyol akış yoğunluğuna göre değiştirilebilir) için aşağıdaki kuralları oluşturabiliriz [4].

EĞER (120 sn.lik bir periyotta)

Yeşil Işık AZ süreli ise → 30 sn yeşil ışık ,ardından 90 sn. kırmızı ışık yanar.
Yeşil Işık ORTA süreli ise → 60 sn yeşil ışık, ardından 60 sn. kırmızı ışık yanar.
Yeşil Işık ÇOK süreli ise → 90 sn.yeşil ışık, ardından 30 sn. kırmızı ışık yanar.

Tüm bunlara bakarak 360 saniyelik bir periyottaki yeşil ışık durumları örnek başlangıç durumlarına göre Tablo 4.3'de gösterilmiştir.

Tablo 4.3 : Örnek başlangıç durumlarına göre L1'in ışık süreleri

T (sn)	Yol 1 Akış Yoğ.	Yol 3 Akış Yoğ.	Yeşil Işığın 'Sözel' Yanma Süresi	Yeşil Işık Süresi (sn)	Kırmızı Işık Süresi (sn)
1-120	A	A	O	60	60
120-270	A	O	Ç	90	30
270-360	O	O	O	60	60

Simulaworks programında zamanlı geçişlerin zamanını saniye cinsinden değil de geçişlerin ateşlenme hızları olarak verebildiğimiz için yukarıdaki kurallara ve Tablo 4.3'e dayanarak Simulaworks programında hazırlanan TEZ CD'sindeki "isik.sim" dosyasındaki Petri ağında zamanlı geçişler ;

30 sn'lik gecikme için 3 birim hızlı
60 sn'lik gecikme için 1,5 birim hızlı
90 sn'lik gecikme için 1 birim hızlı

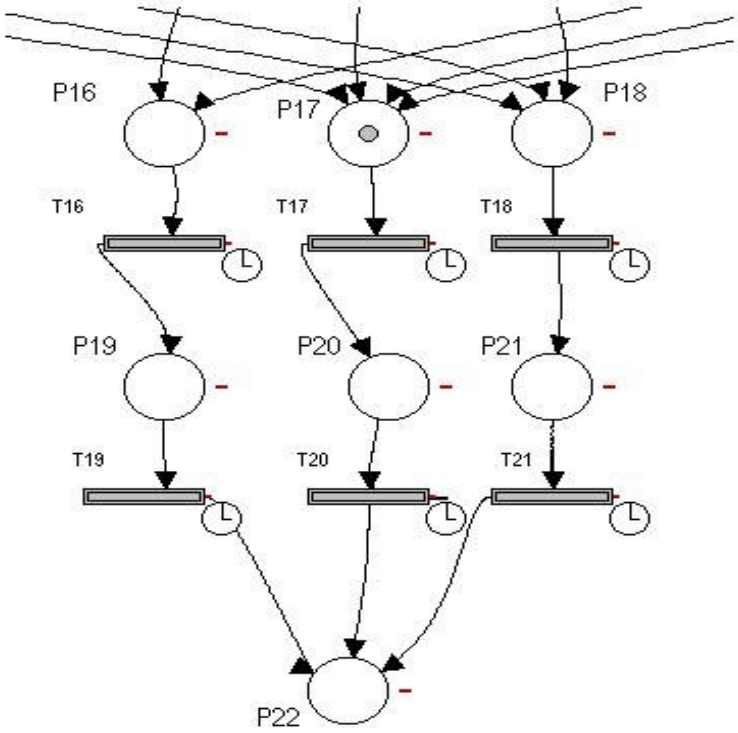
olarak tasarlanmıştır. Buna göre Petri ağındaki zamanlı geçişlerin hızları ;

V (T16) = 3 br. V (T17) = 1,5 br V (T18) = 1br.
V (T19) = 1 br. V (T20) = 1,5 br. V (T21) = 3 br. 'dir.

O halde örnek olarak P1 = A, P4= A başlangıç durumu için P17 yerine jeton geldiğinde T17 geçişi başlangıç konumuna gelir 1,5 br hızlıdır yani temsili olarak 60 sn bekler ateşlenir ve P20 yeri jeton kazanır.

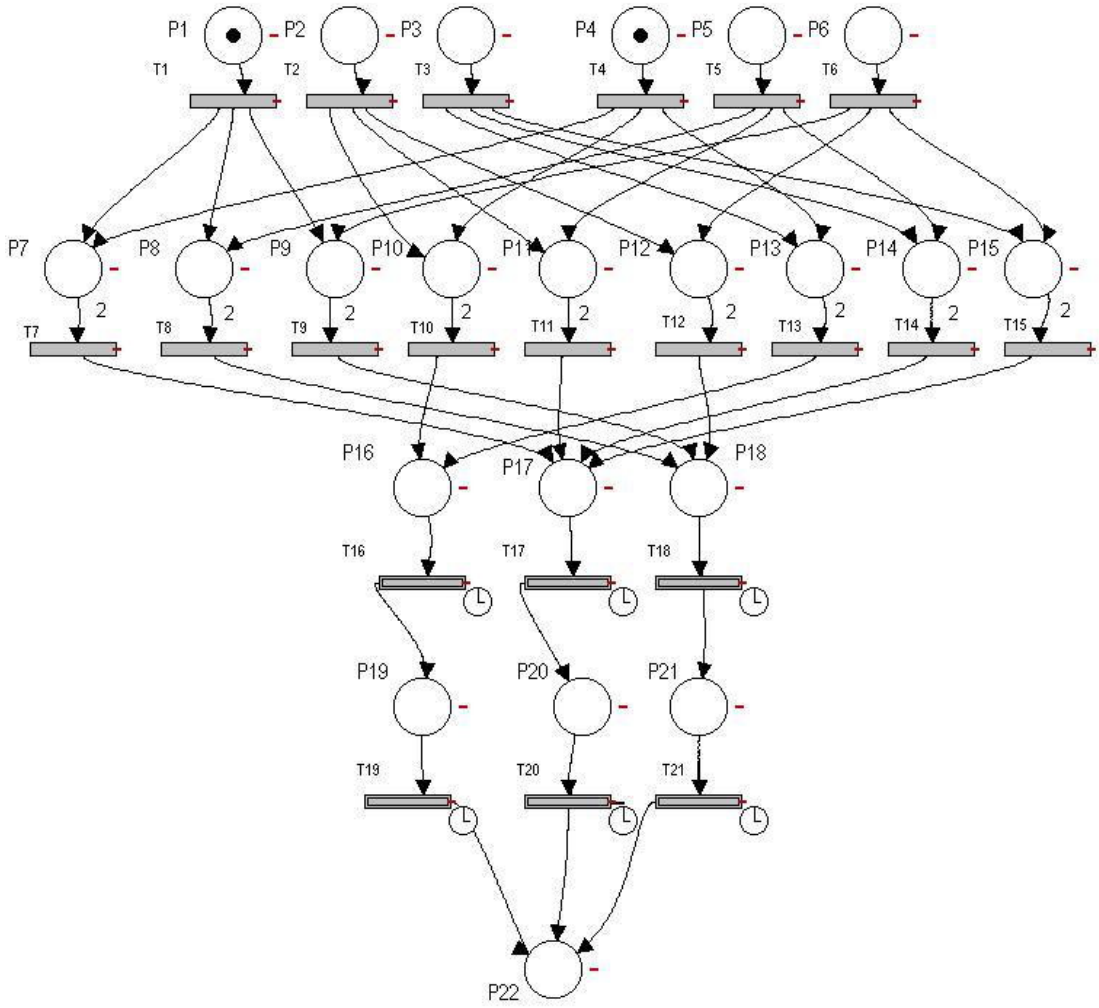
P20 jetonu kaybettiğinde T20 başlangıç konumuna gelir ve 1,5 br hızlıdır yani temsili olarak 60 sn. bekler ve sonrasında ateşlenir ve P22 jeton kazanmış olur. Petri ağı ilk 120 saniyelik periyot için sonlanmış olur. İkinci periyot için Simulaworks'de petri ağını sıfırlayıp yeni başlangıç değerlerini (yerler için jetonları) girmek gerekir. Bu şekilde Tablo 4.3'deki örnek başlangıç durumlarına göre 360 sn'lik periyotta L1'in ışık sürelerini zamanlı Petri ağı tasarımı ile gözlemlemiş oluruz (Şekil 4.4).

TEZ CD'sindeki Simulaworks program dosyası "isik.sim"'de bu petri ağının başlangıç durumları değiştirilerek (giriş yerlerine jetonlar eklenerek) L1 lambasının ışık süreleri ile ilgili ağdaki değişmeler gözlemlenebilir. Örnek olarak Yol1 = A ve Yol3 = A koşulları için Simulaworks programında Şekil 4.5, Şekil 4.6, Şekil 4.7, Şekil 4.8, Şekil 4.9'daki durumlar gözlenmişti [4].



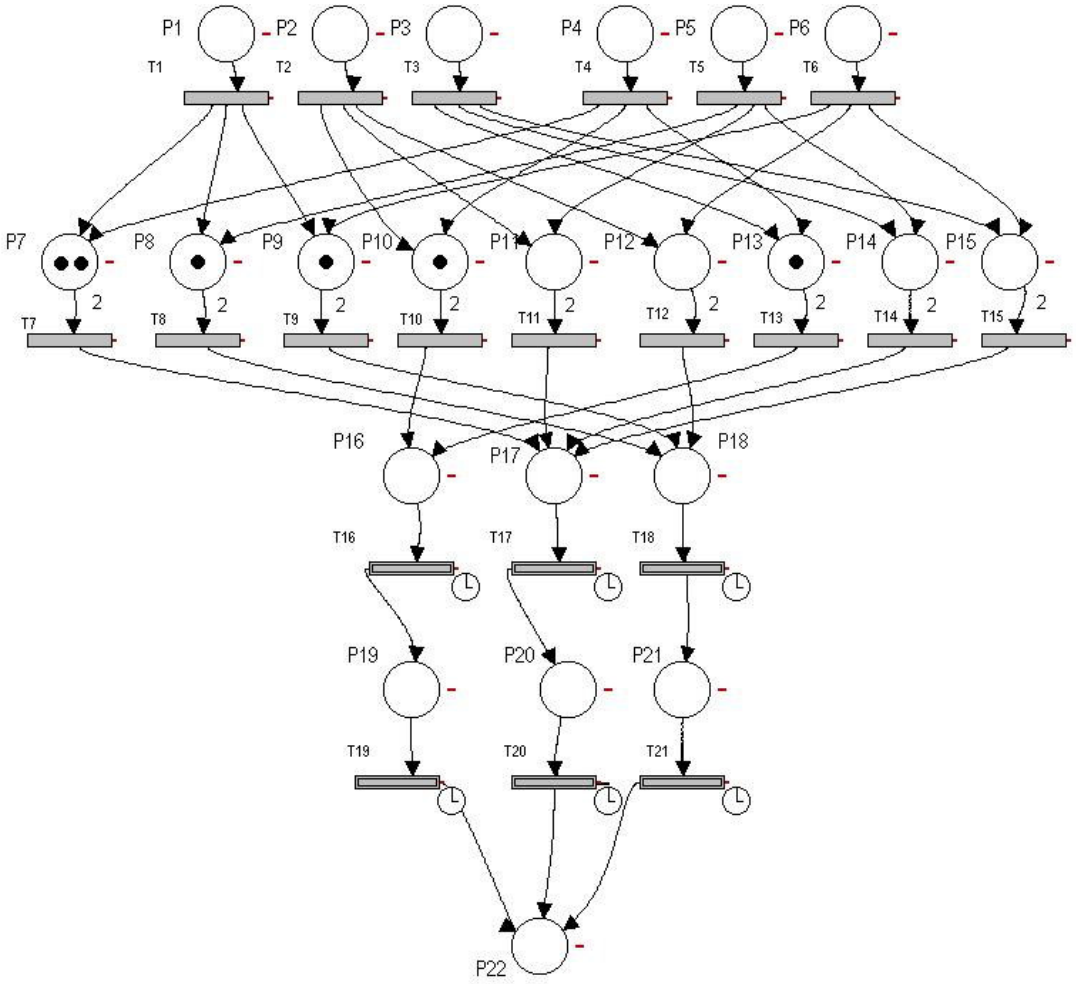
Şekil 4.4 : Trafik ışığı için Petri ağındaki zamanlı geçişler

Başlangıç durumunda P1, P4'de birer jeton bulunur yani durum vektörü X_0=[1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]'dır (Şekil 4.5).



Şekil 4.5 : Trafik ışığı zamanlı Petri ağı için başlangıç durumu

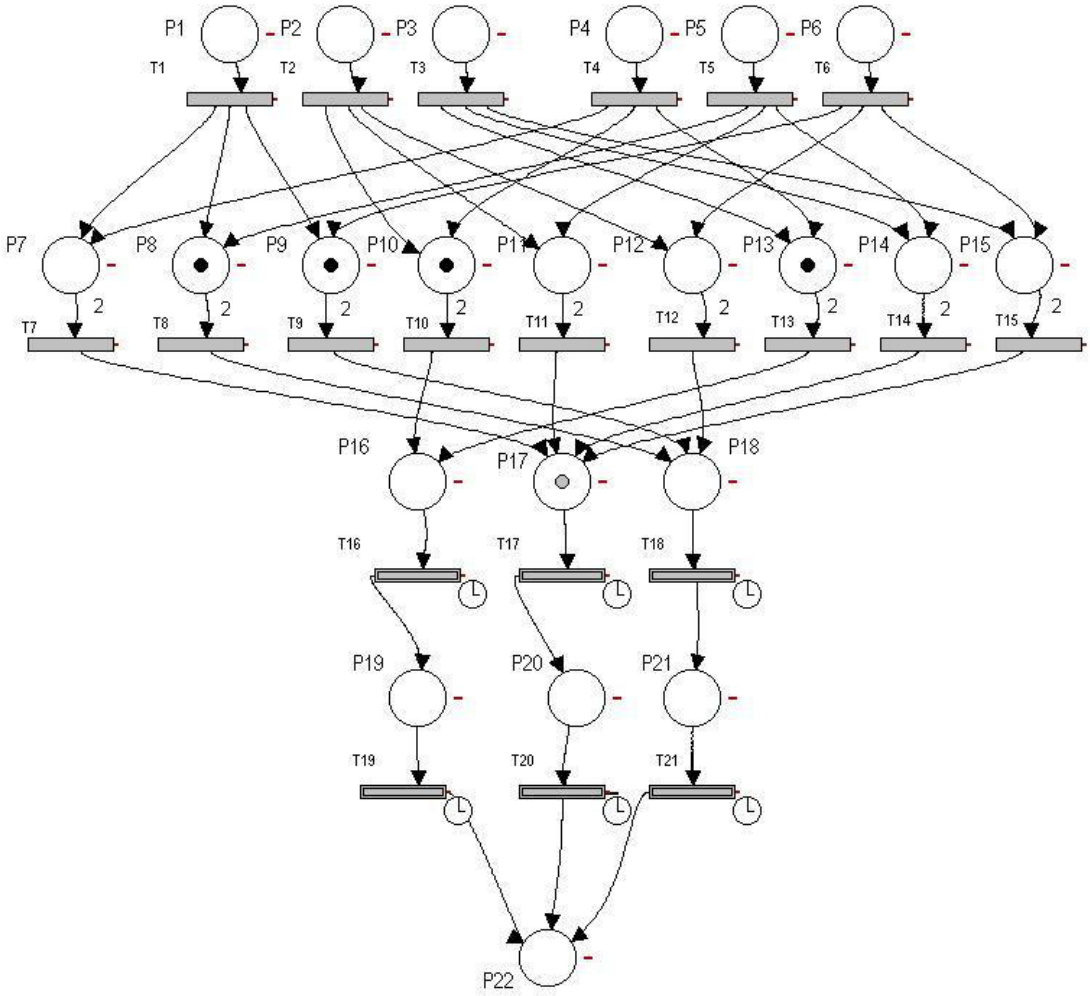
İkinci aşamada T1 ve T4 ateşlenir. Sonrasında P8, P9, P10, P13 birer jeton P7 ise iki jeton kazanır. X_1=[0 0 0 0 0 0 2 1 1 1 0 0 1 0 0 0 0 0 0 0 0 0]'dır (Şekil 4.6).



Şekil 4.6 : Trafik ışığı zamanlı Petri ağı için ikinci aşama

Üçüncü aşamada iki jetona sahip olan P7 jetonunu kaybeder ve T7'nin ateşlenmesini sağlar ve P17 jeton kazanır.

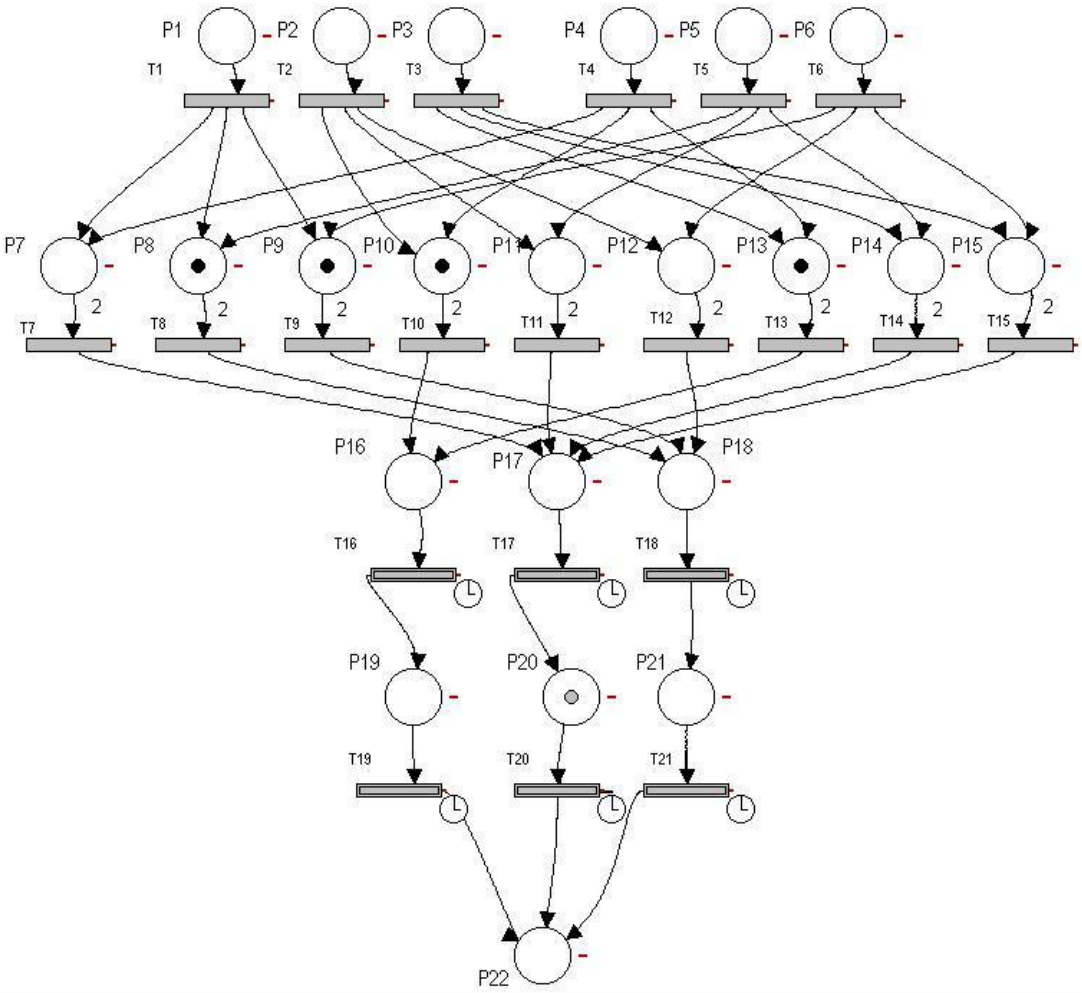
X_2=[0 0 0 0 0 0 0 1 1 1 0 0 1 0 0 0 1 0 0 0 0 0]'dır (Şekil 4.7).



Şekil 4.7 : Trafik ışığı Petri ağı için üçüncü aşama

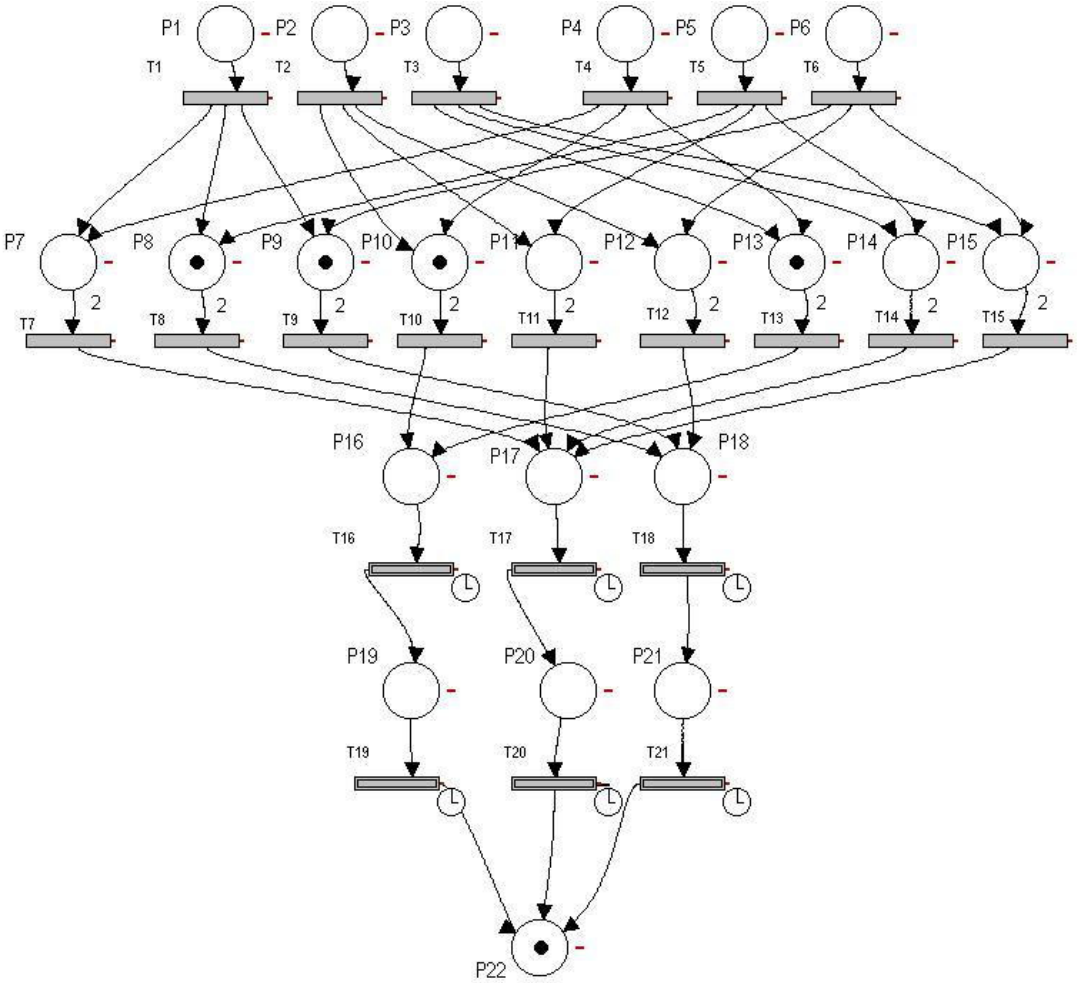
Dördüncü aşamada T17 temsilen 60 sn. gecikmeli olarak (1,5 hızlı olarak) ateşlenir ve P17 jetonunu kaybeder, P20 jeton kazanır.

X_3=[0 0 0 0 0 0 0 1 1 1 0 0 1 0 0 0 0 0 0 1 0 0]'dır (Şekil 4.8).



Şekil 4.8 : Trafik ışığı Petri ağı için dördüncü aşama

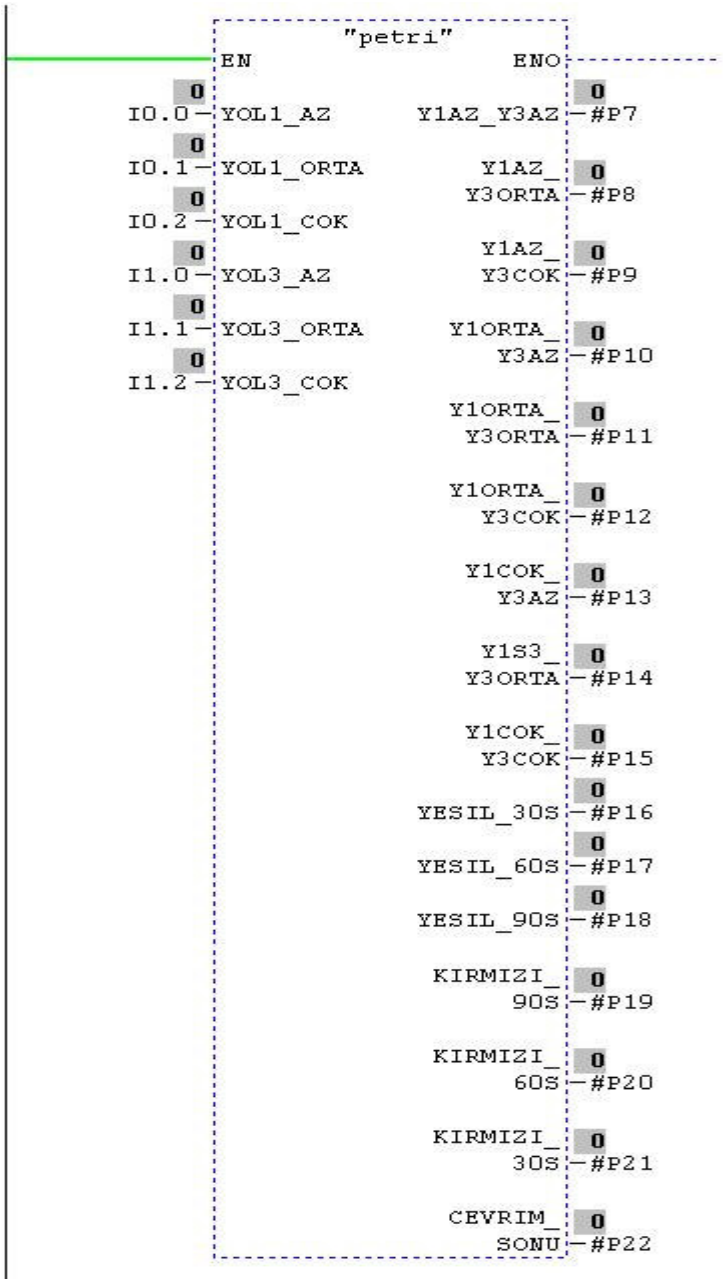
Son aşamada T20 temsilen 60 sn. gecikmeli olarak (1,5 hızlı olarak) ateşlenir ve P20 jetonunu kaybeder, P22 jeton kazanır ve temsilen 120 saniyelik ışık sureli Petri ağı sonlanmış olur. X_4=[0 0 0 0 0 0 0 1 1 1 0 0 1 0 0 0 0 0 0 0 1]'dır (Şekil 4.9).



Şekil 4.9 : Trafik ışığı Petri ağı için son aşama

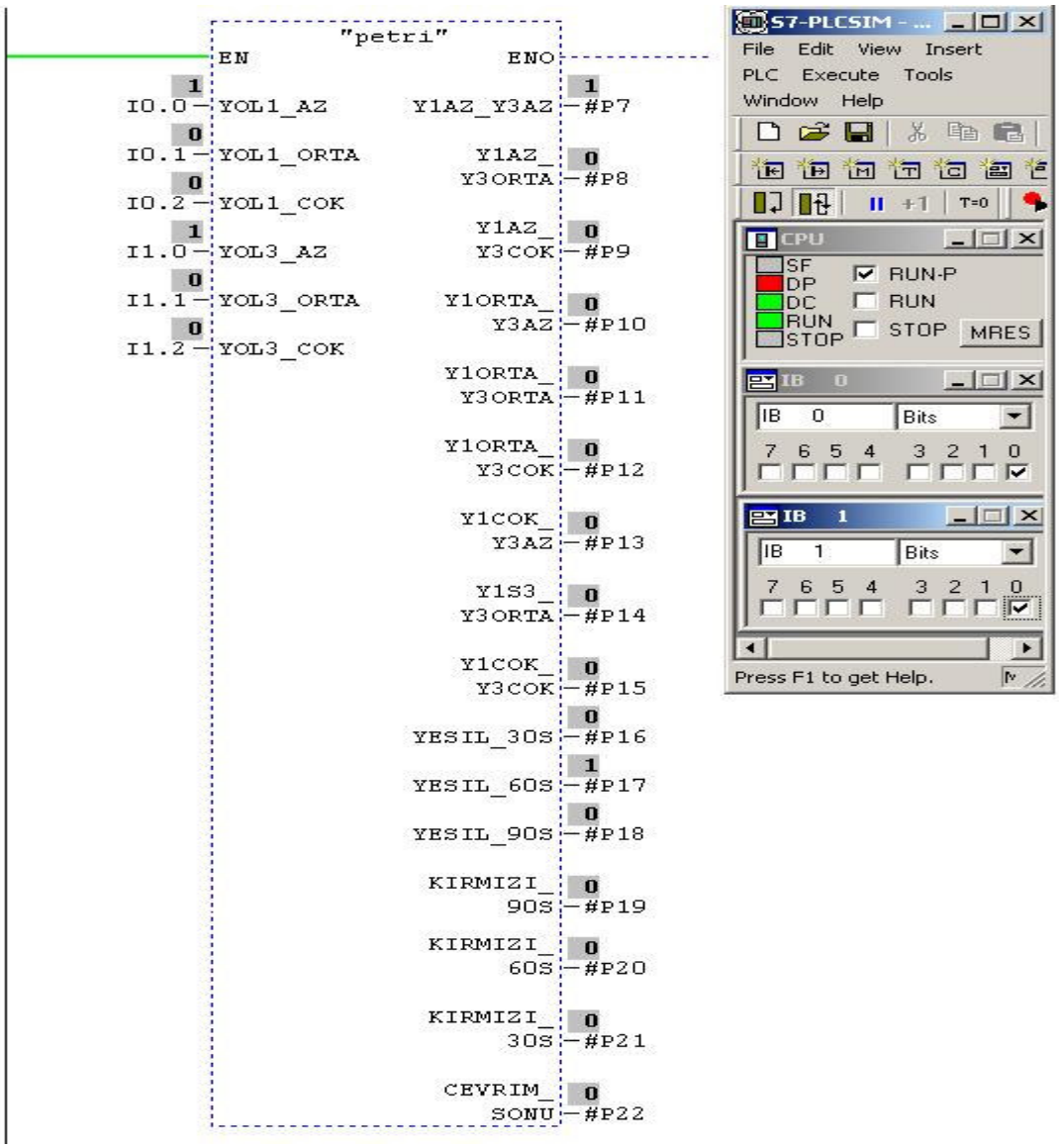
4.3 Zamanlı Petri Ağ Modeli Bilinen Sistemin Simatic Manager Uygulaması

Siemens Simatic Manager program dosyasında Petri ağının başlangıç durumları değiştirilerek (giriş olarak farklı sensör bilgileri girilerek) L1 lambasının ışık süreleri ile ilgili ağdaki değişmeler gözlemlenebilir. Örnek olarak Yol1=Az ve Yol3=Az koşulları için Simatic Manager programında Şekil 4.10, Şekil 4.11, Şekil 4.12, Şekil 4.13'deki durumlar gözlenmiştir.



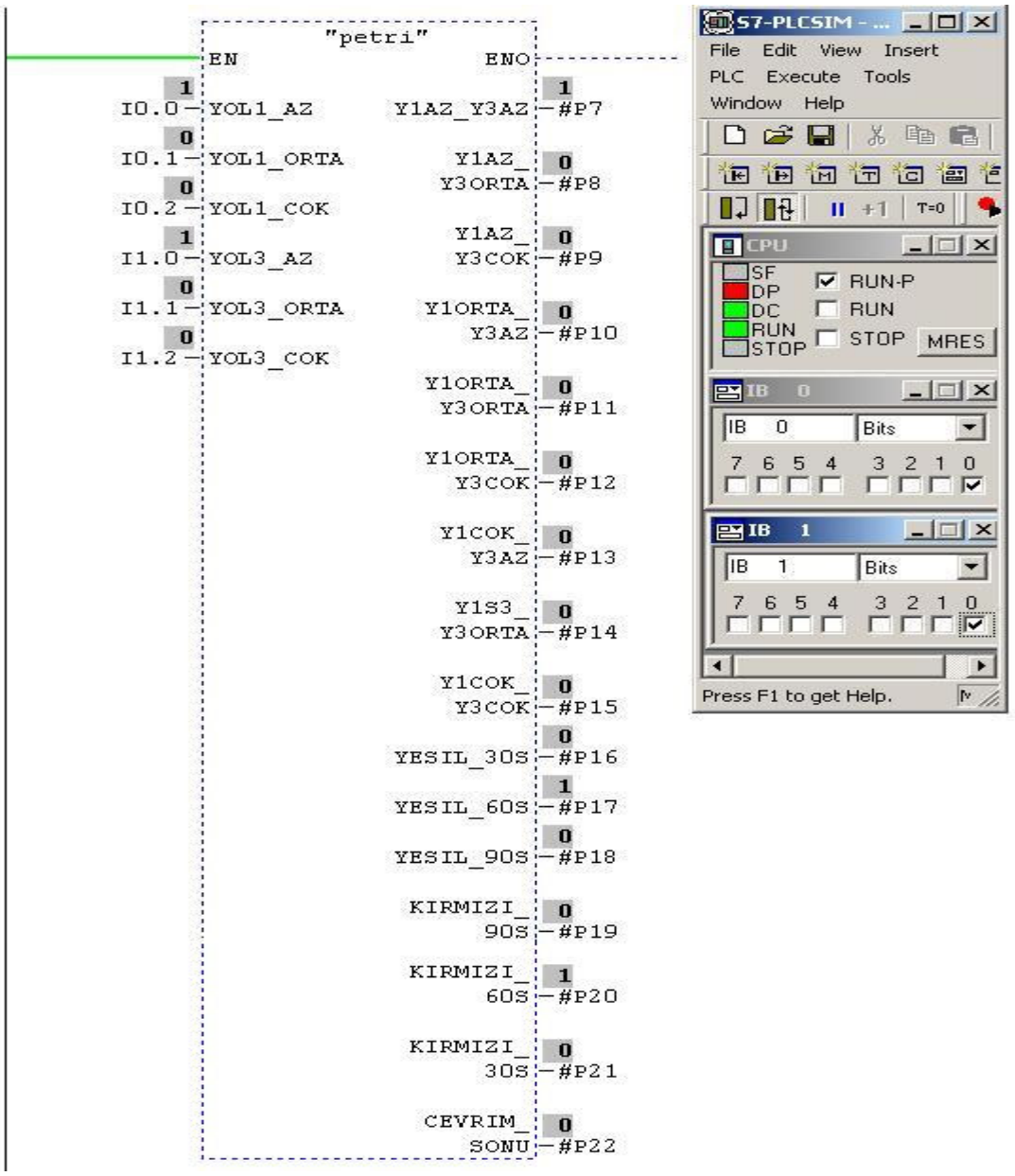
Şekil 4.10 : Data bloğun genel görünüşü

Programı oluştururken ilk önce bir fonksiyon içerisine gerekli olan lojik yapılar girilmiştir. Daha sonra organizasyon bloğun içerisinden fonksiyon çağırılıp toplu bir görüntü elde edilmiştir. Sisteme giriş olarak Yol 1 ve Yol 3'ten alınan sensör bilgileri bulanık mantık kurallarından yararlanılarak uzman bilgisine göre her iki yol için AZ, ORTA ve ÇOK yoğun olmak üzere üç parametre kullanılmıştır. Seçilen girişlerin (sensörlerden alınan bilgilere göre) Yol1 için AZ, Yol 3 için AZ yoğun olduğunu varsayalım.



Şekil 4.11 : YOL 1 ve YOL 3 bilgilerinin verilme durumu

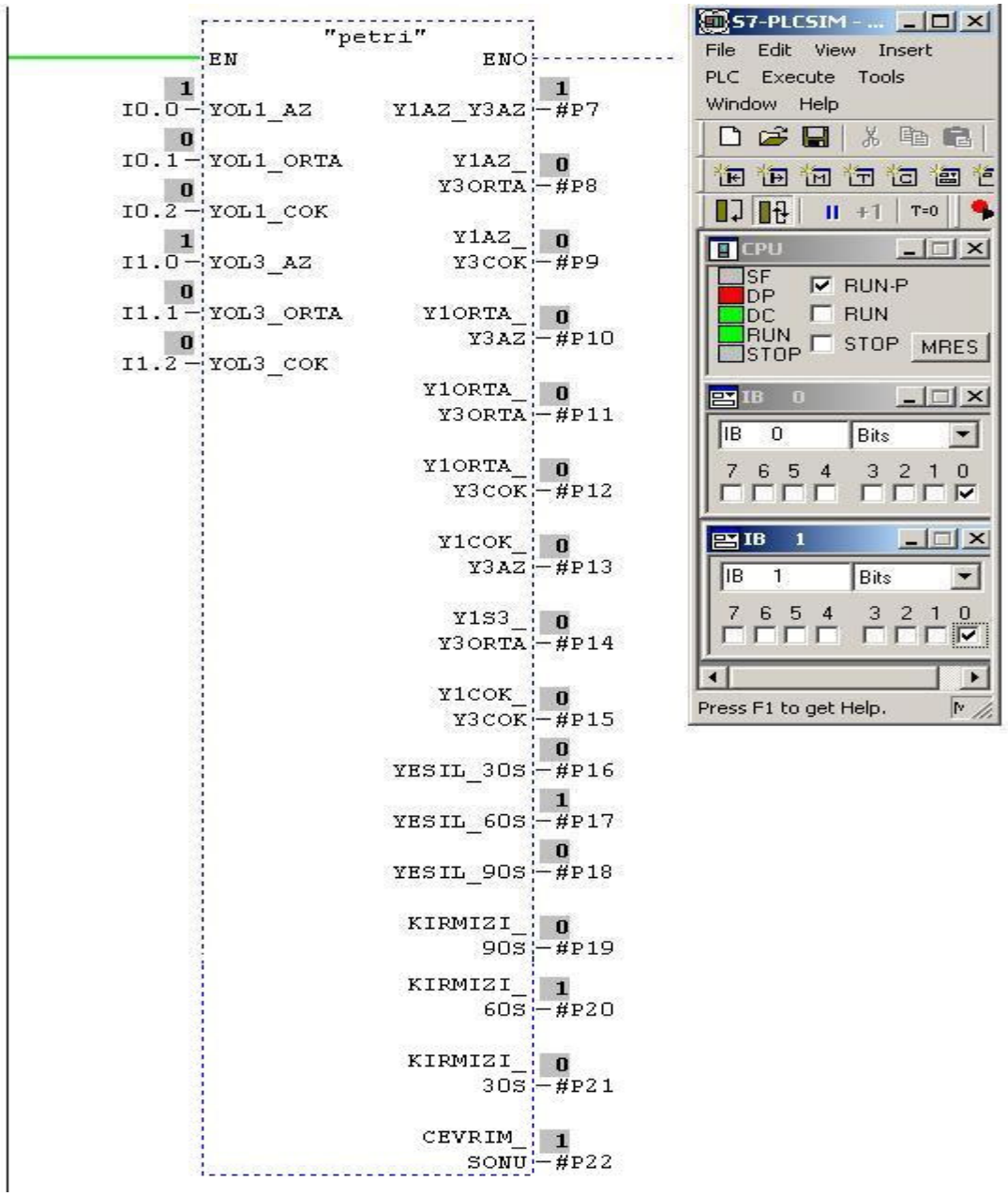
Bu durumda Petri ağımızdaki P7 çıkışı aktif olacak ve böylece bulanık mantık kurallarına göre hazırlanan program, P17 çıkışını aktif hale getirecektir. Bu çıkışın aktif olmasıyla 60 saniyelik bir gecikme süresi saymaya başlar. Saymaya başlayan süre aynı zamanda yeşil ışığın yanma süresidir. Süre sona erdiğinde yeşil ışık söner ve bu seferde P20 çıkışı aktif olur. Bu da kırmızı ışığın yanma süresini başlatır. Yine 60 saniyelik bir gecikme vardır.



Şekil 4.12 : Yol bilgilesine göre L1 lambasının yeşil-kırmızı yanma süreleri

Bu süre dolduğunda anda 120 saniyelik yeşil ve kırmızı ışık periyodu sona erer. Petri ağında olduğu P22 çıkışı da çevrimin bittiğini gösterir.

Burada farklı sensör bilgileri için farklı zamanlama gecikmeleri olacağı açıktır. Bunun yanı sıra bulanık mantık kuralları artırılıp azaltılarak daha fazla sayıda seçenek bulunabilinir.



Şekil 4.13 : Son durumun oluşma hali

5. SONUÇ

Bu çalışmada zamanlı modellerin özellikleri ve analiz yöntemleri incelenerek bir ayrık olaylı sistemin modellenmesine uygulanmıştır. Sıradan modeller hiç zaman kavramı içermezler. Bu tip ağ sınıflarıyla ağın zamansal gelişimini değil sadece modellenen sistemin lojik yapısını açıklamak mümkündür. Bu çalışmada ayrık olay sisteminin zamansal analiz ihtiyacından dolayı, Petri ağlara ve otomat yapısına zaman kavramı getirilmiştir.

Ayrık olay sistemlerinde diğer sürekli sistemlerde olduğu gibi bilinen sürekli bir zaman kavramı yerine bir olayın gerçekleşme anlarının önemi olduğu için trafik sistemlerinde araçların otoyollara girişi, çıkışı, manevraları hepsi birer olaydır ve ayrık olay sistemi ile modellenebilirler. Bu yüzden 'Otoyol Giriş Denetimi' konusu incelenmiş ve ayrık olay sistem dillerinden olan zamanlı otomat yapısı ve zamanlı Petri ağlarıyla ilgili olarak iki adet uygulama gerçekleştirilmiştir. Petri ağları ayrık dinamik sistemleri modellemeye çok uygundur ve anlamı kesin olarak belli olan matematiksel tanımlara sahiptir. Ayrıca Petri ağının gerçeklenmesiyle ilgili çok sayıda analiz metodu vardır. Petri ağının tasarımı Simulaworks paket programı yardımıyla gerçekleştirilmiş, içinde bulunan "jeton-oyun animasyonları" sayesinde ağın işleyişi gözlemlenmiştir. Sonrasında Siemens Simatic Manager yazılım programı ile DES modeli olan zamanlı Petri ağı merdiven diyagramı yöntemiyle yazılıp bilgisayar ortamında gerçeklenmiştir. Sonuç olarak zamanlı modellerden zamanlı otomat ve zamanlı Petri ağının gerçek sistemlere verimli bir şekilde uygulanabildiği incelenmiş ve görülmüştür.

KAYNAKLAR

[1] **Cassandras G.C., Lafortune S.,** 1999. Introduction To Discrete Event Systems, Kluwer Academic Publishers, Massachusetts.

[2] **Alur R., Dill D.L.,** 1994. A Theory of Timed Automata, *Theoretical Computer Science*, No. 126, pp. 183-235.

[3] **Ostroff J.S.,** 1989. Temporal Logic for Real-Time Systems, Research Studies Press and John Wiley & Sons, New York.

[4] **Balta M.C.**, 2002. Otoyol Giriş Denetimi, Yüksek Lisans Tezi, İ.T.Ü. Fen Bilimleri Enstitüsü, İstanbul.

[5] *Bulanık Mantık.* Alıntı Tarihi: Kasım 3, 2007

URL1: [http://science.ankara.edu.tr/~ozbek/bulanik-1.htm]

[6] **Brandin B., Wonham W.M.,** 1994. Supervisory Control of Timed Discrete Event Systems, IEEE Transactions on Automatic Control, Vol. 39, No. 2, pp. 329-342.

[7] **David R., Alla H.,** 1992. Petri Nets & Grafcet: Tools for Modelling Discrete event Systems, Prentice-Hall, New York.

[8] **Gündüz S.**, 1990. Petri ağlarının incelenmesi ve bilgisayar ile simülasyonu, Yüksek Lisans Tezi, İ.T.Ü. Fen Bilimleri Enstitüsü, İstanbul.

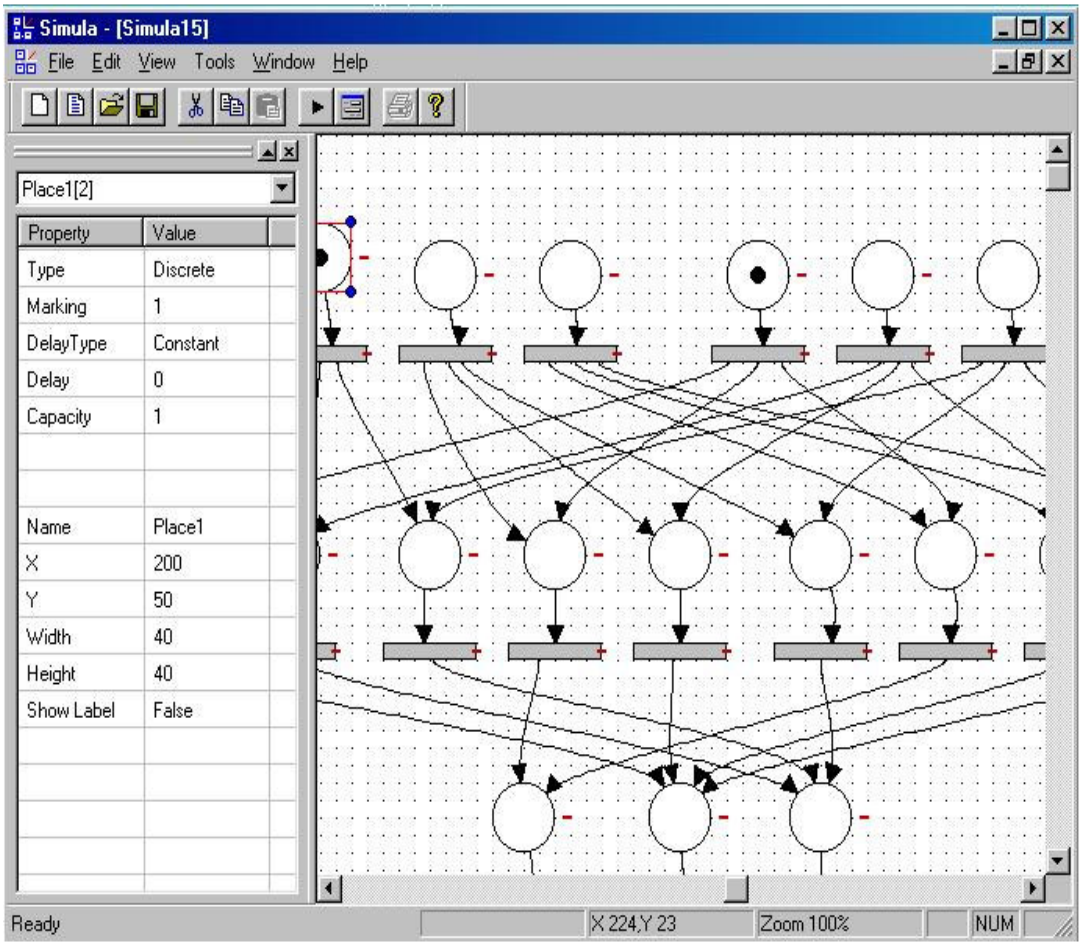
[9] *Petri Nets World.* Alıntı Tarihi: Ekim 24, 2007

URL2: [http://www.daimi.au.dk/PetriNets/tools/quick.html]

EK A

SIMULAWORKS

HAKKINDA ÖZET BİLGİ



Şekil A.1 Simulaworks ekran görüntüsü

Web sitesi: http://www.simulaworks.com/

Kullanılabilirliği: Ücretsiz

Programın Özellikleri:

Desteklenen Petri Ağları

- Yer/Geçiş Ağları
- Stokastik Petri Ağları
- Zamanlı Petri Ağları

Bileşenleri

- Grafik Editör
- Jeton Oyunu Animasyonu
- Hızlı Simülasyon

Çevreler

- PC, MS Windows 95
- PC, MS Windows 98
- PC, MS Windows NT
- PC, MS Windows 2000
- PC, MS Windows XP

Program Tanımı

SimulaWorks paketi çoğu simulasyon dillerine ve AI tekniklerine açık genel amaçlı bir paket programdır. Kontrol çevrimleri, lojik ve donanım devreleri, üretim sistemleri, trafik sistemleri v.b.'de kullanılabilir. Bu tezde kullanılmasının amacı zamanlı petri ağının modellemesine olanak tanımasıdır.

EK B

PETRİ AĞLARI ANALİZİNDE KULLANILAN STATEMENT LIST DİLİNDE YAZILMIŞ KOD DOSYALARI

Fonksiyon:

```
A    #YOL1_AZ
=    #M00
A    #YOL1_ORTA
=    #M01
A    #YOL1_COK
=    #M02
A    #YOL3_AZ
=    #M10
A    #YOL3_ORTA
=    #M11
A    #YOL3_COK
=    #M12
A    #M00
A    #M10
=    #Y1AZ_Y3AZ
A    #M00
A    #M11
=    #Y1AZ_Y3ORTA
A    #M00
A    #M12
=    #Y1AZ_Y3COK
A    #M01
A    #M10
=    #Y1ORTA_Y3AZ
A    #M01
A    #M11
=    #Y1ORTA_Y3ORTA
A    #M01
A    #M12
=    #Y1ORTA_Y3COK
A    #M02
A    #M10
=    #Y1COK_Y3AZ
A    #M02
A    #M11
=    #Y1S3_Y3ORTA
O    #Y1ORTA_Y3AZ
O    #Y1COK_Y3AZ
```

```
=    #YESIL_30S
O    #Y1AZ_Y3AZ
O    #Y1ORTA_Y3ORTA
O    #Y1S3_Y3ORTA
O    #Y1COK_Y3COK
=    #YESIL_60S
O    #Y1AZ_Y3ORTA
O    #Y1AZ_Y3COK
O    #Y1ORTA_Y3COK
=    #YESIL_90S
A    #YESIL_30S
L    S5T#3S
SD   T     1
A    T     1
=    #KIRMIZI_90S
A    #KIRMIZI_90S
L    S5T#9S
SD   T     2
A    #YESIL_60S
L    S5T#6S
SD   T     3
A    T     3
=    #KIRMIZI_60S
A    #KIRMIZI_60S
L    S5T#6S
SD   T     4
A    #YESIL_90S
L    S5T#9S
SD   T     5
A    T     5
=    #KIRMIZI_30S
A    #KIRMIZI_30S
L    S5T#3S
SD   T     6
O    T     6
O    T     4
O    T     2
=    #CEVRIM_SONU
A    #CEVRIM_SONU
L    S5T#3S
SD   T     7
A    T     7
R    #CEVRIM_SONU
R    #Y1AZ_Y3AZ
R    #Y1AZ_Y3ORTA
R    #Y1AZ_Y3COK
R    #Y1ORTA_Y3AZ
```

R #Y1ORTA_Y3ORTA
R #Y1ORTA_Y3COK
R #Y1COK_Y3AZ
R #Y1S3_Y3ORTA
R #Y1COK_Y3COK
R #YESIL_30S
R #YESIL_60S
R #YESIL_90S
R #KIRMIZI_30S
R #KIRMIZI_60S
R #KIRMIZI_90S

EK C

TEZ CD'SİNDE BULUNANLAR

Paket Programlar ve Bunlarda Kullanılan İlgili Dosyalar

SimulaWorks

- isik.sim

ÖZGEÇMİŞ

İbrahim ÜNAL, 1982 yılında Antalya'da doğdu. Lise eğitimini Manisa'da tamamladıktan sonra 2000 yılında Yıldız Teknik Üniversitesi Elektrik Mühendisliği Bölümünü okumaya hak kazandı. 2005 yılında mezun olduktan sonra aynı yıl içinde İstanbul Teknik Üniversitesi Kontrol ve Otomasyon Mühendisliği Yüksek Lisans eğitimi almaya hak kazandı. 2007 yılında Kalealtınay Robotik ve Otomasyon A.Ş.'de Yazılım Geliştirme Mühendisi olarak işe girdi ve halen aynı şirkette çalışmalarını sürdürmektedir.