ZAMANLANDIRILMIŞ OTOMATA MODELİNDE ZAMAN ADIMI YAKLAŞIMI

İbrahim AÇIKSÖZ

Yüksek Lisans Tezi

Elektrik-Elektronik Mühendisliği Anabilim Dalı

Haziran - 2006

JÜRİ VE ENSTİTÜ ONAYI

İbrahim Açıksöz'ün **"Zamanlandırılmış Otomata Modelinde Zaman Adımı Yaklaşımı"** başlıklı **Elektrik-Elektronik Mühendisliği** Anabilim Dalındaki, Yüksek Lisans Tezi 12.05.2006 tarihinde, aşağıdaki jüri tarafından Anadolu Üniversitesi Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliğinin ilgili maddeleri uyarınca değerlendirilerek kabul edilmiştir.

	Adı – Soyadı	**İmza**
Üye (Proje Danışmanı)	**: Yard. Doç. Dr. AYDIN AYBAR**	**.....**
Üye	**: Prof. Dr. ALTUĞ İFTAR**	**.....**
Üye	**: Yard. Doç. Dr. YUSUF OYSAL**	**.....**

Anadolu Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun …… tarih ve …… sayılı kararıyla onaylanmıştır.

Enstitü Müdürü

ÖZET

Yüksek Lisans Tezi

ZAMANLANDIRILMIŞ OTOMATA MODELİNDE ZAMAN ADIMI YAKLAŞIMI

İbrahim AÇIKSÖZ

Anadolu Üniversitesi
Fen Bilimleri Enstitüsü
Elektrik-Elektronik Mühendisliği Anabilim Dalı

Danışman: Yard. Doç. Dr. Aydın AYBAR
2006, 99 sayfa

Bu tezde, kesikli olay sistemlerinin modelleme yöntemlerinden, otomata gösterimi ele alınmıştır. Literatürdeki zamanlandırılmış otomata gösterimleri örneklerle birlikte sunulmuştur. Zaman adımı yaklaşımı kullanılarak, zamanlandırılmış otomatalar için yeni bir gösterim sunulmuştur. Ayrıca sunulan model için çeşitli algoritmalar geliştirilmiştir. Bu algoritmalar, sistem çıkmazı oluşan durumları bulmakta, sistem çıkmazı meydana gelmesini önlemekte ve sistemin istenilen durumlara en kısa sürede ulaşmasını sağlayan olay dizisini bulmaktadır. Daha sonra, bu algoritmalar Matlab kullanılarak gerçeklenmiştir.

Anahtar Kelimeler: Kesikli Olay Sistemleri, Zamanlandırılmış Otomata, Sistem Çıkmazı, Kontrolör, Durum Kontrolü

ABSTRACT

Master of Science Thesis

TIME STEP APPROACH
AT TIMED AUTOMATA MODEL

İbrahim AÇIKSÖZ

Anadolu University
Graduate School of Sciences
Electrical and Electronics Engineering Program

Supervisor: Assist. Prof. Dr. Aydın AYBAR
2006, 99 pages

In this thesis, automata model which is a modeling method of discrete event systems is considered. Timed automata models which are in the literature with example are presented. A new mathematical model for the timed automata is presented by using the time step. Furthermore the algorithms for the presented model are developed. These algorithms determine the deadlock states, avoid deadlock and determine the event sequence which is formed in the shortest time that contains the desired states. Afterwards, they are implemented by using Matlab.

Keywords: Discrete Event Systems, Timed Automata, Deadlock, Controller, State Control

TEŞEKKÜR

Hazırlamış olduğum bu yüksek lisans tezinde, öncelikle bu tezin konusunu bana öneren, çalışmalarımın her aşamasında bana yol gösteren, yardımlarını esirgemeyen ve çalışmamdaki gecikmelere gösterdiği anlayıştan dolayı danışmanım Yard. Doç. Dr. Aydın AYBAR'a ve tez çalışmalarım sırasında ve öncesinde benden hiçbir maddi ve manevi desteklerini esirgemeyen aileme çok teşekkür ederim.

İbrahim AÇIKSÖZ

Haziran - 2006

İÇİNDEKİLER

Sayfa

ŞEKİLLER DİZİNİ

ŞEKİLLER DİZİNİ

SİMGELER ve KISALTMALAR DİZİNİ

O : Otomata

X : Sonlu durumlar kümesi

Σ : Sonlu olaylar kümesi

δ : Kısmi geçiş fonksiyonu

x_0 : Başlangıç durumu

X_m : İşaretlenmiş (hedef veya son) durumlar kümesi

Σ^* : Olaylarla oluşturulan bütün sonlu dizilerin kümesi

ε : Boş olay

G : Zamanlandırılmış otomata

C : Sonlu zamanlayıcılar kümesi

E : Sonlu kenarlar kümesi

R^+ : Pozitif gerçek sayılar

N : Doğal sayılar

A : Sonlu aktivite kümesi

δ_a : Kısmi aktivite geçiş fonksiyonu

a_0 : Başlangıç aktivitesi

A_m : İşaretlenmiş (hedef veya son) aktivitelerin kümesi

u_e : e olayına ait üst zaman sınırı

$\imath_e$: e olayına ait alt zaman sınırı

T_e : e olayına ait zamanlayıcı aralığı

T : Geçiş-zaman fonksiyonu

K : Koşul kümesi

Y : Sonlu çıktı kümesi

λ : Çıktı fonksiyonu

F : Hata modu

D : Zaman gecikmeleri kümesi

d_u : Birim zaman gecikmesi

e.b.o.b : En büyük ortak bölen

SİMGELER ve KISALTMALAR DİZİNİ

τ : Olaylara ait zaman adımı sayılarının kümesi

$Q(e)$: e olayının çıktı olarak bağlı olduğu durumların oluşturduğu küme

$\lambda(e,q)$: q durumundan sonra e olayı için ilave edilen ara durumların kümesi

$\varphi(e,q)$: e olayının zaman gecikmesine bağlı olarak her bir çıktı olarak bağlı olduğu q durumu için ilave edilen olaylar kümesi

δ^T : Zaman adımlarını işleyen kısmi geçiş fonksiyonu

1. GİRİŞ

Olay etkileşimli sistemler; bir olayın ortaya çıkışının, başka olayların meydana gelişine bağlı olduğu sistemlerdir. Kesikli olay sistemler olarak da adlandırılabilen olay etkileşimli sistemlere örnek olarak otomatik üretim sistemleri, haberleşme ağları, bilgisayar ağları, vb. sistemler verilebilir [1].

Kesikli olay sistemleri, çok fazla değişkene sahip olduğu ve bir olayın meydana gelişinin diğer olay veya olaylara bağlı olması nedeniyle, bu sistemleri ifade edebilmek için çeşitli modelleme yöntemleri geliştirilmiştir. Kesikli olay sistemlerinin modellenmesinde, [2]'de gösterildiği gibi, Petri ağları, veri akış tanımlamaları (işaretlenmiş grafikler, senkronize veri akış grafikleri veya Boolean veri akış grafikleri gibi), Kahn işlem ağları, sonlu durum makineleri (Markov zincirleri), sıralı işlem modelleri, kuyruk sistemi modelleri kullanılabilir. Kesikli olay sistemlerinin modelleri iki gruba ayrılabilir [3]: i) Olayların oluş zamanını ihmal edip sadece olayların oluş sırasıyla ilgilenildiği modeller (kesikli olay sistemlerde bir durumdan başka bir duruma geçiş olay olarak adlandırılmaktadır) [1], ii) Zaman bilgisinin önemli olduğu modeller. i. gruba ait modellere mantıksal modeller veya zamansız modeller denir. Mantıksal modelleme, haberleşmedeki sıralı işlemler, işletim sistemlerinin eşlemesi, merkezi kontrol, haberleşme protokolleri, sayısal devrelerin mantıksal analizi, veritabanı protokolleri gibi birçok uygulamada başarılı bir şekilde kullanılmıştır. ii. gruba ait modeller ise, mantıksal modellemenin tersine zaman bilgisinin önemli olduğu ve modele eklenmesi gereken durumlarda kullanılır. Böyle modellere de zamanlandırılmış modeller veya performans modelleri denir. Sinyal işleme bilgilerinin tasarımında, üretim sistemlerinin periyodik davranışlarını incelemek için basit olarak bu model kullanılabilir [3].

Kesikli olay sistemlerinin modellenmesi ve kontrolü için otomata tabanlı ilk çalışma Ramadge ve Wonham [4] tarafından 1982'de yapılmıştır. Otomata modeli, durumları, olayları ve durumlarla olaylar arasındaki ilişkileri kapsayan ve gösteren matematiksel ve grafiksel bir model olarak verilmiştir. Ramadge ve Wonham'ın çalışmasından sonra, otomata modeli üzerinde kapsamlı olarak durulmuştur (örneğin bkz. [3, 5–10]). Fakat sistemlerin daha gerçekçi

modellenebilmesi için, zaman faktörünün de modele eklenmesi gerektiği ortaya çıkmıştır [3].

Zamanlandırılmış otomata teorisi Alur ve Dill tarafından [11, 12]'de verilmiştir. Bu model sürekli zaman modelindeki gibi zamanın sürekli artması mantığı temel alınarak geliştirilmiştir. Zaman bilgisi, gerçek değerli zamanlayıcılar, durum geçiş grafiklerine eklenerek verilmiş ve zamanlandırılmış model oluşturulmuştur. Zaman hesaplamaları için gerçek sayılar kullanıldığı için, sistem tam zaman değerleriyle modellenmiştir. Bu modelleme yönteminden farklı olarak, Brandin ve Wonham'ın zamanlandırılmış modeli kesikli zaman modelini temel almıştır [13]. Bu modelleme yönteminde, gerçek zamana göre çalışan tek bir zamanlayıcı kullanılmıştır. Zaman hesaplamaları, her olay meydana geldiği zaman durumlardaki zamanlayıcı değerleri yenilenerek yapılmaktadır. Bu yenileme işlemi, birim zaman kadar süre geçtikçe zaman atlaması olayının meydana gelmesiyle durumdaki zamanlayıcı değerlerinin bir birim azaltılması veya bir olay ateşlendiği zaman durumdaki zamanlayıcı değerlerinden o olaya ait değere başlangıç değerinin atanması şeklinde yapılmıştır. Zad ve ark. [14] çalışmasında ise, bir olay meydana gelmeden önce geçen süre (zaman atlaması olayı sayısı olarak) oklarla gösterilen geçişlere eklenerek zamansız otomata modeline daha çok benzer bir zamanlandırılmış otomata modeli ortaya koymuştur. Yapılan tüm bu çalışmalarda zaman bilgisi durumlar ile ilişkilendirilmiş ve olaylar meydana gelirken oluşan zaman gecikmeleri ihmal edilmiştir.

Bu tezde, zamanlandırılmış kesikli olay sistemlerinin modelleme yöntemlerinden biri olan zamanlandırılmış otomata modelleme yöntemi üzerinde durulmuştur. Bunun için öncelikle kesikli olay sistemlerin modelleme yöntemlerinden biri olan sonlu durum otomata hakkında bilgi verilmiştir. Daha sonra önceden yapılmış olan zamanlandırılmış otomata modelleri örneklerle birlikte açıklanmıştır. Son olarak da, zaman gecikmelerinin olaylar ile ilişkilendirildiği yeni bir zamanlandırılmış otomata gösterimi ortaya konulmuştur. Bu yeni modelde, olaylara ait gerçek zamanlı gecikmelerden elde edilen birim zaman gecikmesine (en büyük ortak bölen) göre bulunan zaman adımı sayıları (pozitif tam sayılar şeklinde) her bir geçişe eklenmiştir. Zaman atlaması olayına benzer olarak zaman hesaplamaları, zaman adımlarıyla yapılmıştır.

Bölüm 2'de, kesikli olay sistemlerinin modelleme yöntemlerinden biri olan sonlu durum otomata modelinin, grafiksel ve matematiksel gösterimi ilgili tanımların verildiği çalışmalar [1, 6, 7], örneklerle beraber detaylı olarak incelenerek elde edilen sonuçlar verilmiştir. Ayrıca gerçek zamanlı sistemlerin modellenmesinde kullanılan zamanlandırılmış otomata modeli üzerinde durulmuştur. Bu amaçla Alur ve Dill'in zamanlandırılmış modeli [11, 12] çalışmaları, Brandin ve Wonham'ın zamanlandırılmış modeli [13] çalışma ve Zad ve ark. zamanlandırılmış modeli [14] çalışması kullanılarak verilmiştir.

Bölüm 3'de, olayların zaman gecikmelerinin modele zaman adımı olarak eklendiği yeni bir zamanlandırılmış otomata modeli detaylı olarak verilmiştir. Ayrıca tanımlanan zamanlandırılmış otomata modeli için hazırlanan algoritmalar ve algoritmaların gerçeklenmesi ile geliştirilen yazılımlar hakkında bilgi verilmektedir. Bölüm 4'de ise, bu çalışmada elde edilen sonuçlar tartışılmıştır.

2. OTOMATA GÖSTERİMİ

Bu bölümde, öncelikle [1, 6, 7]'de sunulan çalışmalar kullanılarak, otomata üzerine bilgi verilecektir. Daha önce yapılmış olan zamanlandırılmış otomata modelleri de [11-16]'de sunulan çalışmalar kullanılarak anlatılacaktır.

2.1. Otomata

Bir önceki bölümde bahsedildiği gibi, otomata modeli, durumları, olayları ve durumlarla olaylar arasındaki ilişkileri kapsayan ve gösteren matematiksel ve grafiksel bir modeldir.

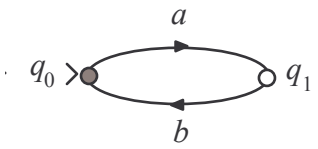
Otomatanın grafiksel gösterimi; durumların, dairelerle, olayların ise yönlü oklarla ifade edilmesidir (bkz. Şekil 2.1). Sistem herhangi bir durumda iken geçerli bir olay meydana geldiğinde, sistem okun gösterdiği duruma geçer.

Otomatanın grafiksel gösteriminde başlangıç durumu gösterilmek istenirse, grafikte bu duruma ">" eklenerek gerçekleştirilir. Ayrıca işaretlenmiş durumlar (hedef durumlar) da eklenmek istenirse, grafikte bu durumlar, diğer durumlardan farklı olarak "O" yerine "●" (bkz. Şekil 2.1) veya iç içe iki daire (bkz. Şekil 2.2) kullanılarak gösterilir.

Otomata modeli, matematiksel olarak ise $O=(X,\Sigma,\delta,x_0)$ şeklinde ifade edilir. Burada, X : durumların oluşturduğu kümeyi, Σ : olayların oluşturduğu kümeyi, x_0 : başlangıç durumunu, $\delta:\Sigma\times X\to X$ kısmi geçiş fonksiyonunu gösterir. $q\in X$ ve $e\in\sum$ için tanımlanan geçiş fonksiyonunda, q durumundayken e olayı meydana gelirse, $q'=\delta(e,q)$ bir sonraki durumu gösterir ($\delta(e,q)$, eğer ve yalnızca eğer e olayı q durumundayken meydana gelebiliyorsa tanımlıdır). Σ^* ise, Σ kümesine ε boş olayının da eklenmesi ile oluşan kümenin elemanlarından oluşturulan tüm sonlu dizileri kapsar [1].

Modellenecek sistemde işaretlenmiş durumlar (hedef durumlar) var ise, bu durumlar X_m ile gösterilir ($X_m\subseteq X$).

Örneğin, Şekil 2.1'deki basit örnek için $\Sigma=\{a,b\}$ ve $X=\{q_0,q_1\}$ 'dır. Başlangıç ve aynı zamanda son durumu q_0 'dır ($x_0=X_m=\{q_0\}$). $\delta(a,q_0)=q_1$ ve $\delta(b,q_1)=q_0$ ise bu sistem için geçerli olan geçişlerdir.

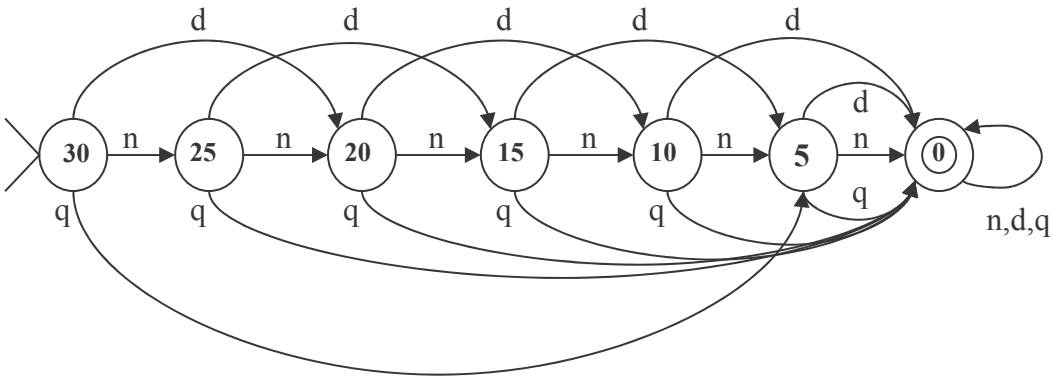


Şekil 2.1. Bir kesikli olay sistem için sonlu otomata modeli

Otomata, bir olay dizisinin, sistemin istenilen (hedef) durumlara ulaşıp ulaşmadığını belirlenmesinde kullanılan etkili bir gösterimdir (Σ^* kümesine ait bir olay dizisinin sistemi, başlangıç durumu x_0'dan ulaşılmak istenen durum $q \in X_m$'a götürüp götürmediğini belirlenmesi). Böylece sistemin istenilen şekilde çalışması ve istenilen işi yapması garanti altına alınır. Örneğin, otomatik satış yapan bir makineyi ele alalım. Bu tür makineler, atılan bozuk paraları girdi olarak kabul ederken, çıktı olarak da yiyecek, içecek vb. verirler. Makine bir dizi girdiyi bekler ve eğer girdi dizisi doğruysa çıktı verir. Bu tür makinelere yapılan girdiler, otomata gösterimindeki Σ^* kümesine ait bir olay dizisidir. Makinenin çıktı vermesi ise, sistemin hedef duruma ulaşmasıdır. Hesaplamanın sonucu; girdi dizisinin kabul edilir olup olmadığını gösterir. Kabul edilebilirlik kavramı makinenin çıktı vermesi olarak kullanılmaktadır [6].

Örneğin, Sudkamp [6]'ın otomatik satış yapan gazete makinesini ele alalım (bkz. Şekil 2.2). Makineye olan girdiler bozuk paralardır ve para birimi sent cinsindendir. 30 sent tamamlandığında, kilit açılır ve gazete alınabilir. 30 sentten fazla para atılırsa makine bunu da kabul eder. Fakat para üstünü vermez. Bu makine fiziksel bir hafızaya sahip değildir. Fakat gazete makinesi örneğin, 25 sentten sonra 5 sent daha atılırsa kilidi açıp gazete alınmasına izin vereceğini bilmektedir. Bunu yeni bir girdi alınıp işlendiği herhangi bir anda makinenin durumunu değiştirmesi ile yapar. Böylece makine her an ne kadar girdi yapıldığını ve çıktı vermek için ne kadar daha girdi yapılması gerektiğini hesaplayabilmektedir.

Şekil 2.2'de de görüldüğü gibi, gazete makinesi için $\Sigma = \{n, d, q\}$, $X = \{30,25,20,15,10,5,0\}$, $x_0 = \{30\}$ ve $X_m = \{0\}$'dır. Sistemin çalışması ise aşağıda verilmektedir:



Şekil 2.2. Gazete makinesinin otomata modeli

- 30 durumu, başlangıç durumunu ve 30 sentte gereksinim olduğunu gösterir. Hiçbir para atılmadığı sürece sistem bu durumdadır.
- 25 durumu, 25 sentte gereksinim olduğunu gösterir. 5 sent atıldıktan sonra bu duruma gelinir.
- 20 durumu, 20 sentte gereksinim olduğunu gösterir. İki tane 5 sentlik veya bir tane 10 sentlik atıldıktan sonra bu duruma gelinir.
- 15 durumu, 15 sentte gereksinim olduğunu gösterir. Üç tane 5 sentlik veya bir tane 10 sentlik ve bir tane 5 sentlik atıldıktan sonra bu duruma gelinir.
- 10 durumu, 10 sentte gereksinim olduğunu gösterir. Dört tane 5 sentlik veya iki tane 10 sentlik veya iki tane 5 sentlik ve iki tane 10 sentlik atıldıktan sonra bu duruma gelinir.
- 5 durumu, 5 sentte gereksinim olduğunu gösterir. Beş tane 5 sentlik veya iki tane 10 sentlik ve bir tane 5 sentlik veya üç tane 5 sentlik ve bir tane 10 sentlik veya bir tane 25 sentlik atıldıktan sonra bu duruma gelinir.
- 0 durumu, en azından 30 sentlik girişin yapıldığı durumu gösterir**.**

Her para atılışında makine durumunu yeniler ve 0 durumuna ulaşınca kilit açılır ve gazetenin alınmasına izin verir. 0 durumu gibi bir durum, girdinin doğruluğunu gösterdiği için bu girdi dizisi kabul edilir olarak adlandırılır.

Makinenin grafiksel gösterimindeki bütün girdiler, bütün durumlar, ilk durumu ve hedef durum sembolik olarak gözlenebiliyor. n, d ve q ile adlandırılan

oklar 5, 10 ve 25 sentlik girdileri gösteriyor. Bu girdilerden birinin uygulanması sistemin durum değiştirmesine neden oluyor.

Bu makineye yapılan girdiler, n, d ve q'ların değişik kombinasyonlarından oluşan diziler olarak tanımlanabilir. Örneğin: dndn, nndn gibi. Dizideki her girdi sırasıyla uygulanır. Eğer uygulanan dizi sistemi hedef duruma götürüyorsa, dizi kabul edilebilir bir dizidir. Örneğin, dndn dizisi makine tarafından kabul edilir. Fakat nndn dizisi kabul edilebilir değildir. Çünkü makine 5 durumunda kalır.

2.2. Zamanlandırılmış Otomata

Zaman bilgisinin, kesikli olay sistemlerinin modellerine eklenmesi bu alanda yapılan çalışmalara yeni bir boyut getirmiştir. Bununla birlikte, modeller belirgin bir şekilde karmaşıklaşmıştır. Bu karmaşıklık, zaman bilgisini herhangi bir modele eklemenin zorluğundan kaynaklanmaktadır [16].

Alur ve Dill tarafından yapılan, [11, 12]'deki çalışmalarda, zaman bilgisinin gerçek değerleriyle verildiği ve gerçek sayılarla gösterildiği bir model ortaya koyulmuştur. Bu modelde, sistemin hangi duruma geçtiği bilgisine ek olarak, sistemin bu duruma geçtiği zaman ve o durumda kaldığı süre ile de ilgilenilmiştir. Sistemin durum değiştirdiği süre (durum değiştirmeye neden olan olayın meydana geldiği an) zamanlandırılmış kelimeler tanımlanarak verilmiştir.

Zamanlandırılmış kelimeleri girdi olarak kabul eden zamanlandırılmış otomata $G = (\Sigma, X, C, E, x_0, x_m)$ şeklinde verilmiştir. Burada, Σ: sonlu olaylar kümesi, X: sonlu durumlar kümesi, C: sonlu zamanlayıcılar kümesi, E: sonlu kenarlar kümesi, x_0: başlangıç durumu ve x_m: son (işaretlenmiş) durumlar kümesi olarak tanımlanmıştır.

Tanım 2.1: $e_{i_1} e_{i_2} ... e_{i_n}$ şeklindeki olay dizisi bir kelime göstermektedir ($e_{i_1}, e_{i_2}, ..., e_{i_n} \in \Sigma^*$). (e_i, t_i) zamanlandırılmış bir olayı göstermek üzere, $(e_{i_1}, t_{i_1})(e_{i_2}, t_{i_2}) \ldots (e_{i_n}, t_{i_n})$ şeklindeki zamanlandırılmış olay dizisi ise zamanlandırılmış kelimeyi göstermektedir. Burada $t_i \in R^+$ sayısı e_i olayının geçen zamana göre meydana geldiği anı gösterir. e_{i_j} olayı e_{i_k} olayından sonra meydana geliyorsa, $t_j > t_k$ olmalıdır. Fiziksel olarak da bir sistemde olayların

farklı anlarda meydana gelmesi ve bir sonraki olayın geçen zamana göre daha ileriki bir zamanda meydana gelmesi bunun nedenidir. Sistemin herhangi bir durumda beklediği süre ise zamanlayıcılar ve bu zamanlayıcılar kullanılarak oluşturulan zaman koşulları tanımlanarak verilmiştir.

Tanım 2.2: Bir zamanlayıcı, bir olay meydana geldiğinde sıfırlanabilen pozitif tamsayılar alan değişkenlerdir. Zamanlayıcılar, gerçek zaman geçerken eşzamanlı olarak artarlar ve bir önceki sıfırlandığı andan itibaren geçen süreyi hesaplarlar. Zamanlayıcıların kümesi ise $C = \{c_1, c_2, ..., c_n\}$ şeklindedir. Burada, n sayısı (zamanlayıcı sayısı), sistem modellenirken gereksinimlere göre belirlenir.

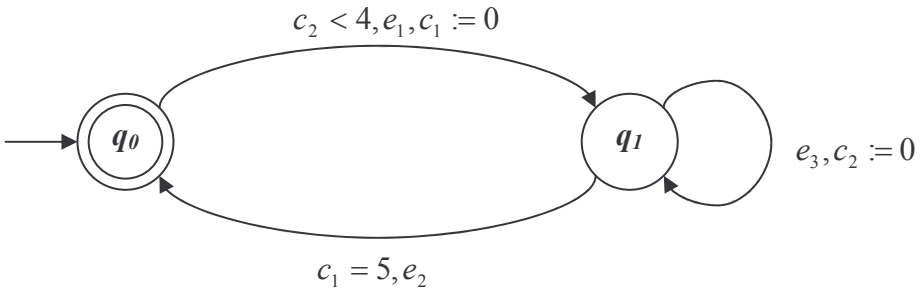
Tanım 2.3: Bir zaman koşulu $c_i \Theta k$ şeklinde gösterilmektedir. Burada Θ, bağıntısal simgeleri $(<, >, \leq, \geq, =)$ ifade ederken $c_i \in C$ ve $k \in N$ (N:doğal sayılar)'dir. Grafiksel gösterimde, zaman koşullar geçişlere eklenir.

Bir durumdan başka bir duruma geçiş (olayın gerçekleşmesi), sadece ve sadece zamanlayıcı veya zamanlayıcıların o anki değerinin zaman koşulunu sağlaması ile mümkündür. Örneğin, "$c_1 \geq 1$" bir koşul olarak ele alalım. Eğer c_1 1'e eşit veya büyük ise, geçişi sağlayan olayın meydana gelebileceği anlamına geliyor.

Tanım 2.4: Bir kenar, $(q.q'.e.\delta.c')$ şeklinde verilen, her bir geçişi tanımlayan kısmi geçiş fonksiyonudur. Burada, $q, q' \in X$, $e \in \Sigma$, $c' \in C$'dır. $c' \in C$ zamanlayıcısı olay meydana geldiği anda eğer varsa sıfırlanacak zamanlayıcı gösterir. δ ise Tanım 2.3'de verilen zaman koşuludur. Yapılan tanıma ek olarak, bir geçişe ait zaman koşulu $\wedge$(ve) ve $\vee$(veya) operatörlerini de kullanarak birden fazla olabilir. Örneğin: $c_1, c_2 \in C$ iken $c_1 \geq 1 \vee c_2 = 2$ bir koşuldur. Bu koşul c_1 zamanlayıcısının büyük eşit 1 birim zaman aralığında iken veya c_2 zamanlayıcısının da 2 birim zamanda iken olayın meydana gelebileceği anlamına geliyor. Olayın meydana gelebilmesi için iki zaman koşulunun da sağlanması gerekir. Burada gerçek zaman tekdüze artıyor ve koşulların doğruluğu da bu zamana ve zamanlayıcıların o anki değerlerine bakılarak yapılmaktadır.

Örneğin [15]'dan alınan sistem Şekil 2.3'de verilmiştir. Bu sistem için $\Sigma = \{e_1, e_2, e_3\}$, $X = \{q_0, q_1\}$, $x_0 = x_m = \{q_0\}$, $C = \{c_1, c_2\}$'dir. Kenar kümesi olarak tanımlanan E için ise $q_0 \xrightarrow{e_1} q_1$ $c_2 < 4$ iken, $q_1 \xrightarrow{e_3} q_1$, $q_1 \xrightarrow{e_2} q_0$

$c_1 = 5$ iken geçerlidir. e_1 ve e_3 olayları meydana geldiğinde, sırasıyla c_1 ve c_2 değerleri sıfırlanır. Geçerli olaylar ve meydana gelebilecekleri anlarında verildiği zamanlandırılmış olaylar ise $(e_1,3.2)$, $(e_3,5.1)$ ve $(e_2,8.2)$'dir.



Şekil 2.3. Zamanlandırılmış otomata modeli

- Başlangıçta gerçek zaman ve zamanlayıcıların (c_1, c_2) değeri 0 (sıfır)'dır. Sistem çalışmaya başladığında bu değerler eş zamanlı artmaya başlar.
- Sistem başlangıçta q_0 durumundadır. Sistem çalışmaya başladıktan 3.2 birim zaman sonraki anda $c_1 = c_2 = 3.2$ birim zaman olur. $c_2 < 4$ zaman koşulunu sağlayan zamanlandırılmış e_1 olayı anlık olarak meydana gelir ve sistem q_1 durumuna geçer. Bu geçiş olurken c_1 zamanlayıcısı sıfırlanır. Sistem q_1 durumuna geçtiği anda gerçek zamanın değeri 3.2, $c_2 = 3.2$ ve $c_1 = 0$ birim zamandır. Bu yeni değerlerden başlayarak zaman artmaya devam eder.
- Sistem çalışmaya başladıktan 5.1 birim zaman sonraki anda, gerçek zamanın değeri 5.1, $c_2 = 5.1$ ve $c_1 = 1.9$ birim zamandır. Bu anda zamanlandırılmış e_3 olayı anlık olarak meydana gelir ve sistem q_1 durumundan q_1 durumuna geçer. Bu geçiş olurken c_2 zamanlayıcısı sıfırlanır. Sistem q_1 durumuna geçtiği anda gerçek zamanın değeri 5.1, $c_2 = 0$ ve $c_1 = 1.9$ birim zamandır. Bu yeni değerlerden başlayarak zaman artmaya devam eder.
- Sistem çalışmaya başladıktan 8.2 birim zaman sonraki anda, gerçek zamanın değeri 8.2, $c_2 = 3.1$ ve $c_1 = 5$ birim zamandır. Bu anda $c_1 = 5$ koşulunu sağlayan zamanlandırılmış e_2 olayı anlık olarak meydana gelir ve sistem q_1

durumundan q_0 durumuna geçer. Sistem q_1 durumuna geçtiği anda gerçek zamanın değeri 8.2, $c_2 = 3.1$ ve $c_1 = 5$ birim zamandır.

Brandin ve Wonham, [13]'deki çalışmada, olayların meydana gelebileceği anın, her olaya ait birer alt ve üst sınır ile belirlendiği kesikli zaman bir model ortaya koymuşlardır. Bu modelde; zaman bilgisi, hiçbir olay meydana gelmese dahi, zaman atlaması olayının meydana gelmesi ve değişen durumlardaki zamanlayıcı değerlerinin değişimi ile verilmiştir. Zaman değişimi, sistemin durum değiştirmesi gibi düşünülerek, zamansız otomata modeline daha çok benzer bir zamanlandırılmış model oluşturulmuştur. Olaylar, her birine ait alt ve üst sınırlar ile birleştirilerek zamanlandırılmış olaylar oluşturulmuştur.

Zamanlandırılmış otomata modeli, zamansız otomata modeline benzer olarak $G = (X, \Sigma, \delta, x_0, x_m)$ şeklinde verilmektedir. Fakat bu zamanlandırılmış modeli oluşturabilmek için modellemeye $G_a = (\Sigma_a, A, \delta_a, a_0, A_m)$ şeklindeki zamansız model ile başlanmıştır. Burada A: sonlu aktivite kümesi, Σ_a: sonlu olaylar kümesi, δ_a: kısmi aktivite geçiş fonksiyonu, a_0: başlangıç aktivitesi ve A_m: işaretlenmiş (hedef) aktivitelerin kümesidir ($A_m \subseteq A$). Kısmi aktivite geçiş fonksiyonu $\delta_a : \Sigma_a \times A \to A$ şeklinde tanımlanır ve $[a, e, a']$ veya $a' = \delta_a(e, a)$ ile gösterilir.

Tanım 2.5: Zaman atlaması olayı, gerçek zamana göre belirli birim zaman geçtikçe meydana geldiği kabul edilen olaydır. Zaman atlaması olayı her meydana geldiğinde, geçerli durumdaki zamanlayıcı değerleri bir birim azaltılır.

Tanım 2.6: Zamanlandırılmış olay, $\sum$ olay kümesine ait herbir e olayına, bir alt zaman sınırı ve bir üst zaman sınırı eklenmesi ile oluşturulur ve (e, l_e, u_e) şeklinde gösterilir. Alt sınır, olayın meydana gelebilmesi için geçmesi gereken minimum zamanı gösterirken; üst sınır, olayın meydana gelebileceği son zamanı gösterir. Bir olaya ait alt sınır sonlu ($l_e \in N$) olmak zorunda iken, bu koşul üst sınır için geçerli olmayabilir. Bunun nedeni olayın meydana gelebileceği zaman aralığının bir üst sınırının olmaması (üst sınırın açık olması), başka bir deyişle, alt sınır kadar zaman geçtikten sonra herhangi bir anda olayın meydana gelebilmesidir.

Olay, alt sınır ve üst sınır arasındaki bir değerde meydana gelebilir. Eğer alt ve üst sınır birbirine eşit ise olay o anda meydana gelebilir. Ayrıca başlangıç durumuna ait zamanlayıcı değeri, olaya ait alt ve üst sınırlara bakılarak yapılmaktadır.

Tanım 2.7: Zamanlayıcı değerleri, başlangıç durumundan başlayarak her duruma eklenen ve olayların meydana gelebilirliği hakkında zaman bilgisi veren değerlerdir. Durumlara atanan zamanlayıcı değerleri ve bu işlem yapılırken oluşan durum sayısı zamanlayıcı aralığına bakılarak yapılmaktadır. Zamanlayıcı aralığı, T_e ile gösterilir. Eğer bir olaya ait üst sınır sonlu bir sayı ($u_e \in N$) ise zamanlayıcı aralığı sıfır ile üst sınır aralığına eşittir. Eğer bir olaya ait üst sınır sonlu bir sayı değil (üst sınır açık) ise o olaya ait zamanlayıcı aralığı sıfır ile alt sınır aralığına eşittir. Buna göre,

$$T_e = \begin{cases} [0, u_e], \text{eğer } u_e \text{ sonlu bir sayı ise} \\ [0, l_e], \text{eğer } u_e \text{ sonlu bir sayı değil ise} \end{cases} \quad (2.1)$$

olur. Durumlara ait zamanlayıcı değerleri, olaylara ait zamanlayıcı aralıklarının olası tüm birleşimlerinin kullanılması ile bulunur. Herhangi bir durumdaki, e olayına ait zamanlayıcı değeri t_e değişkeni ile gösterilir. Bu değişkenin alabileceği değerler geçerli olaya ait zamanlayıcı aralığının elemanı olmalıdır ($t_e \in T_e$). Başlangıç durumuna ait zamanlayıcı değeri $t_{e,0}$ ile gösterilir. Eğer bir olaya ait üst sınır sonlu bir sayı ise başlangıç durumuna ait zamanlayıcı değeri üst sınıra eşittir. Eğer bir olaya ait üst sınır sonlu bir sayı değil ise başlangıç durumuna ait zamanlayıcı değeri alt sınıra eşittir (Denklem (2.1)'de verilen zamanlayıcı aralığının üst sınırına bakılarak da bulunabilir). Buna göre,

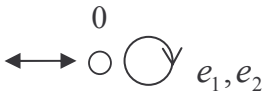
$$t_{e,0} = \begin{cases} u_e, \text{eğer } u_e \text{ sonlu bir sayı ise} \\ l_e, \text{eğer } u_e \text{ sonlu bir sayı değil ise} \end{cases} \quad (2.2)$$

olur. Sistem, başlangıç durumundan itibaren her olay veya zaman atlaması meydana geldiğinde zamanlayıcı değerini veya değerlerini dolayısıyla durumunu değiştirir. Eğer meydana gelen zaman atlaması ise tüm olaylara ait zamanlayıcı değerleri bir birim azaltılır. Eğer meydana gelen olay ise sadece o olaya ait zamanlayıcı değerine başlangıç değeri atanır ve diğer zamanlayıcı değerleri değişmeden kalır.

Aktivite kümesi, zaman bilgisinin daha modele eklenmediği durum kümesi olarak kullanılmaktadır. Aktivite kümesi elemanlarına, (2.1) ve (2.2) kullanılarak ve Tanım 2.7'de verilen açıklamaya uygun şekilde zamanlayıcı değerleri eklenerek X durum kümesi oluşturulur. Σ_a, sonlu olaylar kümesine zaman atlamasının da eklenmesiyle Σ, olaylar kümesi oluşturulur ($\Sigma = \Sigma_a \cup \{\text{zaman atlaması}\}$). Başlangıç durumu, başlangıç aktivitesine bu aktiviteye ait zamanlayıcı değerlerinin atanması ile bulunur ve $x_0 = (a_0, \{t_{e,0} | e \in \Sigma_a\})$ şeklinde gösterilir. x_m, işaretlenmiş durumlar kümesi de aynı yöntemle işaretlenmiş aktivite kümesi kullanılarak bulunur. Kısmi geçiş fonksiyonu ise $\delta : X \times \Sigma \to X$ şeklinde gösterilir.

Örneğin, Şekil 2.4'de aktivite geçiş fonksiyonu verilen sistem için $\Sigma_a = \{e_1, e_2\}$, $A = A_m = \{0\}$, $a_0 = 0$ ve $\delta_a(e_1,0) = \delta_a(e_2,0) = 0$ olarak veriliyor. Zamanlandırılmış olaylar (e_1,1,1) ve (e_2,2,3)'dir.

$G_a = (\Sigma_a, A, \delta_a, a_0, A_m)$ zamansız modelinden $G = (X, \Sigma, \delta, x_0, x_m)$ zamanlandırılmış modeline geçebilmek için durum kümesi $X = \{0\} \times T_{e_1} \times T_{e_2} = \{0\} \times [0,1] \times [0,3]$ şeklinde oluşturulur. Burada, durum kümesi, aktivitelere e_1, e_2 olayları için tanımlanan zamanlayıcı aralıklarının olası tüm kombinasyonlarının ([0,0], [0,1], [0,2], [0,3], [1,0], [1,1], [1,2], [1,3]) ayrı ayrı atanması ile meydana getirilir (bkz. Şekil 2.5). Buradan durum kümesinin 8 elemanlı olduğu görülmektedir.

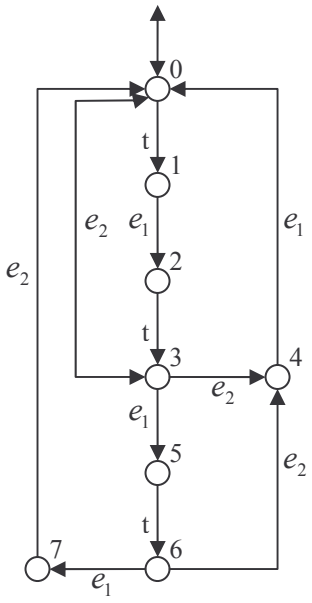


Şekil 2.4. Geçerli aktivite geçişleri

Durum:	0	1	2	3	4	5	6	7
$[t_{e_1}, t_{e_2}]$:	[1,3]	[0,2]	[1,2]	[0,1]	[0,3]	[1,1]	[0,0]	[1,0]

Şekil 2.5. Durumlardaki zamanlayıcı değerlerini gösteren tablo

$A = A_m = \{0\}$ olduğu için olaylardan herhangi biri meydana geldikten sonra sistem işaretlenmiş duruma geçer. $t_{e_1,0} = 1$ ve $t_{e_2,0} = 3$ başlangıç durumu 0 için, başlangıç zamanlayıcı değerleridir. İki olaya da ait üst sınırlar sonlu bir sayı olduğu için $x_0 = x_m = \{(0,[1,3])\}$ dır. Olaylardan e_1'in meydana gelebilmesi için 1 birim zaman, e_2'nin meydana gelebilmesi için 2 birim zaman geçmelidir. Sistem çalışırken zaman atlaması olayı her meydana geldiğinde, meydana geldiği durumdaki zamanlandırma değerleri 1 birim azaltır ve diğer duruma geçilir. Eğer sistem çalışırken bir olay meydana gelirse, sistem bir sonraki duruma geçer ve o olaya ait zamanlandırma değerine ilk değeri atanır.



Şekil 2.6. Zamanlandırılmış geçiş grafiği

Zad ve ark. [14]'deki çalışmada, [13]'deki çalışmadan farklı olarak, zamanlandırılmış sistem sadece yeni bir çıktı ürettiğinde durum değiştirir. Bu durum değiştirme, sistemin ürettiği çıktıların ve farklı çıktıların arasında meydana gelen zaman atlaması olayı sayısına bakılarak yapılmaktadır. Bu model, sistemlerdeki fiziksel hataların algılanmasında ve oluşan hatalar sonrası sistemin davranışlarının modellenmesinde kullanılmaktadır.

Zamanlandırılmış sonlu durum Moore otomata modeli $G = (X, \Sigma, \delta, T, x_0, Y, \lambda)$ şeklinde verilmiştir. Bu zamanlandırılmış modeli

oluşturmak için $G_\tau = (X_\tau, \Sigma_\tau, \delta, x_0, Y_\tau, \lambda_\tau)$ şeklindeki sonlu durum Moore otomata modeli ile başlanmıştır. Burada, X_τ: sonlu durum kümesini, Σ_τ: sonlu olay kümesini, Y_τ: sonlu çıktı kümesini, x_0: başlangıç durumunu, $\delta_\tau : X_\tau \times \Sigma_\tau \to X_\tau$: geçiş fonksiyonunu, $\lambda_\tau : X_\tau \to Y_\tau$: çıktı transfer fonksiyonunu gösterir.

Σ_τ'deki olaylardan bir tanesi [13]'deki gibi kesikli zaman mantığı ile çalışan ve τ ile gösterilen zaman atlamasıdır. τ olayının gözlenebilir olduğu kabul edilmiştir. Bu sistemin herhangi bir anda hangi durumda olduğunun belirlenebilmesi için gereklidir. Buradan G_τ, zamanlandırılmış kesikli olay sistemine referans olarak alınmıştır. Ayrıca X_τ, x_0'dan ulaşılabilir olarak kabul edilmiştir. Bu sistemin çalışabilmesi için gereklidir. Sistemde başka bir duruma ulaşmayan durumlar olmamalıdır (herhangi $q \in X_\tau$ için $\delta_\tau(q,e) \neq \phi$ olmalıdır). Çünkü sistemde herhangi bir olay meydana gelmese bile en azından zaman atlaması olayı periyodik olarak meydana gelir. Bu nedenle sistemde geçiş meydana geldiğinde sistem bir duruma ulaşmalıdır ($\delta_\tau(q,\tau) \neq \phi$).

G_τ, sistemin hem normal hem de hata durumundaki davranışını gösterir. $F_1,...F_p$ ile gösterilen p tane hata modunun olduğunu varsayılmıştır. Her hata modu, bir aygıtta (sensör, valf gibi) meydana gelen çeşitli hatalarla ilişkilendirilmiştir. Σ_τ olay kümesi, hata olaylarını da kapsar. Zaman içinde herhangi bir zamanda en fazla bir hata modunun meydana gelebileceğini varsayılmıştır. Bundan başka X_τ durum kümesinin, sistemin koşuluna göre $X_\tau = X_{\tau,N} \cup X_{\tau,F_1} \cup ... \cup X_{\tau,F_p}$ şeklinde bölünebildiği varsayılmıştır. $K = \{N, F_1,..., F_p\}$ koşul kümesi iken, G_τ'in koşul haritası $\kappa_\tau : X_\tau \to K$ şeklinde tanımlanmıştır ($i \in \{1,..., p\}$ için eğer $q \in X_{\tau,F_i}$ ise $\kappa_\tau(q) = F_i$, $\kappa_\tau(q) = N$ ise $q \in X_{\tau,N}$).

Hata tanımlayan modeller için zaman atlaması olayını zamanlandırılmış kesikli olay sistemlerinden çıkarmak modeli daha anlaşılır yapacaktır. Sonuçta oluşan modelde meydana gelen olayların (zaman atlaması olayı haricindeki) sırası ve devam ettiği süre (zaman atlaması olarak) gösterilmiştir. Bu amaçla yapılan

açıklamalardan sonra $G=(X,\Sigma,\delta,T,x_0,Y,\lambda)$ şeklinde gösterilen zamanlandırılmış sonlu durum Moore otomata modeli $G_\tau=(X_\tau,\Sigma_\tau,\delta,x_0,Y_\tau,\lambda_\tau)$ kullanılarak tanımlanmıştır. Burada, X: sonlu durum kümesi, Σ: sonlu olay kümesi ($\Sigma=\Sigma_\tau-\{\tau\}$), Y: sonlu çıktı kümesi ($Y=Y_\tau$), x_0: başlangıç durumu ($x_0=q_0$), δ: kısmi geçiş fonksiyonu, T: geçiş-zaman fonksiyonu, λ: çıktı fonksiyonudur.

X, zaman atlaması dışındaki geçişlerle sistemin gidebileceği durumların kümesidir ve $X=\{q_0\}\cup\{x\in X_\tau \mid \exists e\in\Sigma_\tau-\{\tau\}, x'\in X_\tau : x\in\delta_\tau(x',e)\}$ şeklinde tanımlanır.

Geçiş fonksiyonu, $\delta: X\times\Sigma\to X$ şeklinde gösterilir ve

$$\forall x_1,x_2\in X, e\in\Sigma:\ x_2\in\delta(x_1,e)\Leftrightarrow x_1\overset{e}{\Rightarrow}x_2 \tag{2.3}$$

şeklinde tanımlanır.

T, geçiş-zaman fonksiyonu $T: X\times\Sigma\times X\to \mathrm{N}$ şeklinde tanımlanır. $T(x_1,e,x_2)$, x_1'den x_2'ye geçiş olurken geçen zaman (zaman atlaması olayı sayısının) değeridir.

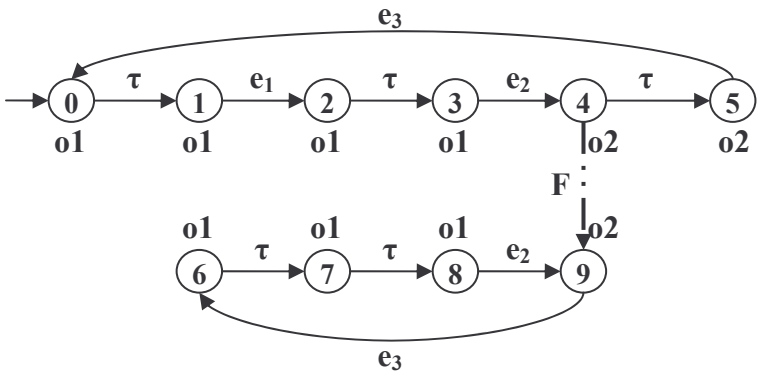
Çıktı transfer fonksiyonu $\lambda: X\times Y$ şeklinde gösterilir (bütün $x\in X\subseteq X_\tau$ için $\lambda(x)=\lambda_\tau(x)$'dır).

Koşul kümesi, $\kappa: X\to K$, bütün $x\in X\subseteq X_\tau$ için $\kappa(x):=\kappa_\tau(x)$ ile tanımlanmaktadır. Kısaca koşul kümesi, sadece zaman atlaması olayından farklı bir olayın meydana gelmesi ile geçilen yeni durumlara bakılarak yapılmaktadır.

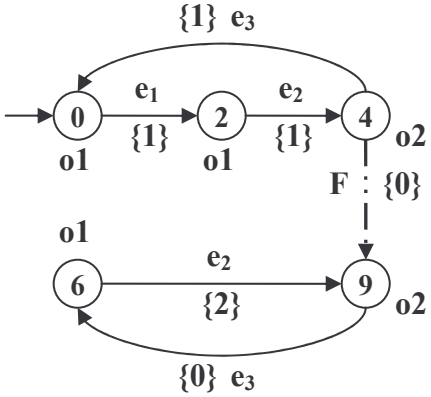
Örneğin Şekil 2.7'deki sistemde, $X_\tau=\{0,1,2,3,4,5,6,7,8,9\}$, $q_0=0$, $\Sigma_\tau=\{e_1,e_2,e_3,\tau,F\}$, $Y=\{o1,o2\}$, $K=\{N,F\}$, $X_{\tau,N}=\{0,1,2,3,4,5\}$, $X_{\tau,F}=\{6,7,8,9\}$ 'dır. Tüm geçerli geçişler ise, (2.3) kullanılarak bulunur.

Sistem F ile gösterilen bir tane hata moduna sahiptir. Hem hata modunda hem de normal modda, sistemin ürettiği çıktı dizisi $(...o1o2o1o2...)$'dır. Bununla birlikte iki modda da zamanlama farklıdır. Normal modda $o1$, $o2$ çıktısından bir zaman atlaması olayından sonra üretilirken; hata modunda $o1$, $o2$ çıktısından hemen sonra meydana gelir. Buna uygun zamanlandırılmış sonlu durum Moore otomata modeli Şekil 2.8'de görülmektedir. Şekil 2.8'de de görüldüğü gibi, zaman

atlaması olayı sayıları $\{.\}$ şeklindeki her bir geçişe eklenerek gösterilmektedir. Örneğin, $\delta(0,e_1)=2$ geçişi için, $T(0,e_1,2)=1$ geçiş-zaman fonksiyonu ile bulunan zaman atlaması olayı sayısı $\{1\}$ şeklinde geçişe eklenerek gösterilmiştir.



Şekil 2.7. Zamanlandırılmış kesikli olay sistemlere basit bir örnek



Şekil 2.8. Zamanlandırılmış sonlu durum Moore otomata modeli

3. ZAMAN ADIMI YAKLAŞIMI İLE OTOMATA GÖSTERİMİ

Bu bölümde, öncelikle verilen zamanlandırılmış otomata modelleme yöntemlerinden farklı olarak, zaman gecikmelerini olaylar ile ilişkilendiren ve sürekli zaman mantığı ile çalışan bir zamanlandırılmış otomata modeli ortaya konacaktır. Daha sonra oluşturulan zamanlandırılmış otomata modeli kullanılarak kesikli zaman mantığı ile çalışan zaman adımı yaklaşımı ile otomata gösterimi verilecektir. Ayrıca zaman adımı yaklaşımı ile oluşturulan otomata için hazırlanan algoritmalar ve yazılımlar hakkında bilgi bu bölümde verilecektir.

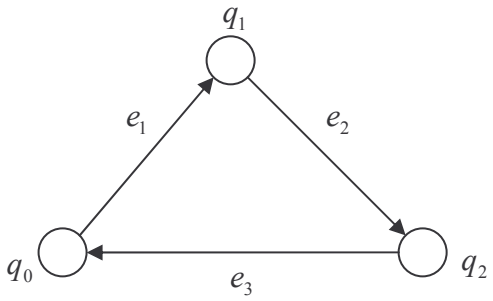
3.1. Zamanlandırılmış Otomata

Bölüm 2.1'de de anlatıldığı gibi, otomata modeli $\mathrm{A} = (X, \Sigma, \delta, x_0)$ şeklinde ifade edilir. Bu tanımdan faydalanarak olaylara D ile gösterilen ve $D = \{d_{e_1}, d_{e_2}, ...\}$ şeklinde tanımlanan zaman gecikmeleri kümesinin her bir elemanı eklenerek $A_T(A, D)$ şeklinde zamanlandırılmış otomata gösterimi elde edilmektedir. T, zamanı, d_e, $e \in \Sigma$ iken olaya ait zaman gecikmesini, T_e, olayın meydana gelmeye başladığı anı göstermek üzere geçiş fonksiyonunun çalışması sürekli zamanda,

$$\delta^T(q_i, e, T) = \begin{cases} q = \delta(q_i, e), \text{eğer } T = d_e + T_e \text{ ise} \\ q_e^* \quad\quad , \text{eğer } T < d_e + T_e \text{ ise} \end{cases} \tag{3.1}$$

şeklinde elde edilir. q_e^*, q_i durumunda e olayının meydana geldiği ancak tamamlanmadığını gösteren durumdur. Otomataya ait tüm olaylar için bu durumların kümesi $\hat{X} = \{q_{e_1}^*, q_{e_2}^*, ...\}$ şeklinde oluşur. Böyle bir geçiş fonksiyonun kullanılması zamanlandırılmış otomata gösteriminde ve algoritma tasarlanmasında zorluklara neden olacaktır.

Örneğin, Şekil 3.1'deki basit örnek için $\Sigma = \{e_1, e_2, e_3\}$ ve $X = \{q_0, q_1, q_2\}$ dır. Başlangıç durumu q_0'dır. $\delta(q_0, e_1) = q_1$, $\delta(q_1, e_2) = q_2$ ve $\delta(q_2, e_3) = q_0$ ise bu sistem için geçerli olan geçişlerdir. Bu sistemde, e_1 olayı için zaman gecikmesi d_{e_1} = 0.05 sn, e_2 olayı için d_{e_2} = 0.2 sn, e_3 olayı için ise d_{e_3} = 0.15 sn olarak alınsın. Buradan $D = \{0.05, 0.2, 0.15\}$ şeklinde oluşur.



Şekil 3.1. Bir kesikli olay sistem için otomata gösterimi

Kabullenme 3.1: Otomatadaki durumların anlık olarak meydana geldiği kabullenilmiştir. Bu nedenle, bu çalışmada sadece olaylar meydana gelirken geçen süre hesaplanmaktadır.

Kabullenme 3.1'e göre otomatada herhangi bir olay meydana gelmeye başladıktan ve tamamlandıktan sonra diğer duruma geçtiği an, ilgili durumdan sonra meydana gelecek olay için başlama anını gösterir. Şekil 3.1'deki örnek için, (3.1) kullanılarak kısmi geçiş fonksiyonu tanımlanırsa, otomatanın çalışması aşağıdaki gibi olur:

- $T = 0$ sn iken otomata q_0 durumundadır. $T = 0$ anında e_1 meydana gelmeye başlar. $d_{e_1} = 0.05$ sn olduğu için e_1 olayının tamamlanabilmesi için 0.05 sn geçmesi gerekir.
- $0 < T < 0.05$ sn iken e_1 olayının meydana gelmesi tamamlanmamıştır (otomata $q^*_{e_1}$ durumundadır). $T = 0.05$ sn ise, e_1 olayı tamamlanır ve otomata q_1 durumuna geçer. e_2 olayının meydana gelmeye başladığı an, $T_{e_2} = 0.05$ sn'dir. $d_{e_2} = 0.2$ sn olduğu için e_2 olayının tamamlanabilmesi için 0.2 sn geçmesi gerekir.
- $0.05 < T < 0.25$ sn iken e_2 olayının meydana gelmesi tamamlanmamıştır (otomata $q^*_{e_2}$ durumundadır). $T = 0.25$ sn iken e_2 olayı tamamlanır ve otomata q_2 durumuna geçer. e_3 olayının meydana gelmeye başladığı an,

T_{e_3} = 0.25 sn'dir. d_{e_3} = 0.15 sn olduğu için e_3 olayının tamamlanabilmesi için 0.15 sn geçmesi gerekir.

- $0.25 < T < 0.4$ sn iken e_3 olayının meydana gelmesi tamamlanmamıştır (otomata $q^*_{e_3}$ durumundadır). T = 0.4 sn iken e_3 olayı tamamlanır ve otomata q_0 durumuna geçer. e_1 olayının meydana gelmeye başladığı an, T_{e_1} = 0.4 sn'dir. d_{e_1} = 0.05 sn olduğu için e_1 olayının tamamlanabilmesi için 0.05 sn geçmesi gerekir.
- $0.4 < T < 0.45$ sn iken e_1 olayının meydana gelmesi tamamlanmamıştır (otomata $q^*_{e_1}$ durumundadır). T = 0.45 sn iken e_1 olayı tamamlanır ve otomata q_1 durumuna geçer. e_2 olayının meydana gelmeye başladığı an, T_{e_2} = 0.45 sn'dir. d_{e_2} = 0.2 sn olduğu için e_2 olayının tamamlanabilmesi için 0.2 sn geçmesi gerekir.

Örnek olarak verilen otomatanın çalışmasından da görüldüğü gibi, zaman gecikmelerinin tamamlanmadığı anlarda yeni durumlar meydana gelmektedir ($\hat{X} = \{q^*_{e_1}, q^*_{e_2}, q^*_{e_3}\}$). Bu durumlar, fiziksel olarak var olmadıkları halde, zamanlandırılmış otomata gösterimini oluşturabilmek için gereklidirler. Ancak sistemin sürekli zaman mantığı ile modellenmesi, zamanlandırılmış otomata gösteriminde ve algoritma tasarlanmasında zorluklara ve karmaşıklığa neden olmaktadır. Bu nedenle, örnekleme anlarının kullanıldığı bir model bir sonraki bölümde verilmiştir.

3.2 Zaman Adımı Yaklaşımı

Bu yeni zamanlandırılmış otomata modeli matematiksel olarak $A^\tau = (X^\tau, \Sigma^\tau, \delta^\tau, D, x_0)$ şeklinde verilmiştir. Bu gösterimde X^τ: sonlu durum kümesini, Σ^τ :sonlu olay kümesini, δ^τ: kısmi geçiş fonksiyonunu, D: olaylara ait zaman gecikmelerini veren pozitif gerçek sayılardan oluşan küme, x_0: başlangıç durumunu gösterir. Bu model otomata modelinden farklı olarak D, olaylara ait zaman gecikmelerini veren kümeye sahiptir.

X , durum kümesi, zaman adımı sayısı birden fazla olan olayların meydana gelmesiyle oluşan ara durumlar eklenerek X^{τ} ile gösterilen durum kümesi oluşturulmuştur. δ , kısmi geçiş fonksiyonu, zaman adımlarından dolayı meydana gelen geçişler de eklenerek δ^{τ}, kısmi geçiş fonksiyonu haline getirilmiştir. Σ, olay kümesi ise zaman adımlarından dolayı oluşan ilave olaylar ile değiştirilerek Σ^{τ} ile gösterilen olay kümesi oluşturulmuştur.

Birim zaman gecikmesi, zaman gecikmelerinin en büyük ortak böleni olarak tanımlanır ve $d_u = e.b.o.b\,(D)$ şeklinde bulunur. Olaylara ait zaman adımı sayısını veren küme ise,

$$\tau = \left\{ \frac{d_{e_1}}{d_u}, \frac{d_{e_2}}{d_u}, \cdots, \frac{d_{e_n}}{d_u} \right\} \tag{3.2}$$

şeklinde tanımlanır. Bu kümede, her eleman ayrı bir olaya ait zaman adımı sayısını vermektedir ($\tau_{e_i} = \dfrac{d_{e_i}}{d_u}$ iken $\tau_{e_i} \in \tau$ 'dir).

$Q(e)$: e olayının çıktı olarak bağlı olduğu durumların oluşturduğu küme (e olayının meydana gelebildiği durumlar kümesi) olarak tanımlanırsa, $q \in Q(e)$ için, q durumundan sonra e olayı için ilave edilen ara durumların kümesi,

$$\lambda(e,q) := \left\{ q^e_{(1)}, q^e_{(2)}, \ldots, q^e_{(\tau_e - 1)} \right\} \tag{3.3}$$

şeklinde gösterilir. Bu durumlar arasındaki bağlantı da $e^q_{(i+1)}$ ile $q^e_{(i)}$'den $q^e_{(i+1)}$'e doğru olacak şekilde yapılmaktadır. Burada, $q \in Q(e)$ için, e olayının zaman gecikmesine bağlı olarak her bir çıktı olarak bağlı olduğu q durumu için ilave edilen olaylar kümesi (e olayının meydana gelebildiği q durumunda zaman adımı sayısına bağlı olarak oluşan olayların kümesi),

$$\varphi(e,q) = \left\{ e^q_{(1)}, e^q_{(2)}, \ldots, e^q_{(\tau_e)} \right\} \tag{3.4}$$

şeklinde gösterilir. Her bir olaya bağlı olarak oluşan ara durumların da eklenmesiyle durum kümesi,

$$X^{\tau} := X \cup \left(\bigcup_{e \in \Sigma} \left(\bigcup_{\tilde{q} \in Q(e)} \lambda(e, \tilde{q}) \right) \right) \tag{3.5}$$

şeklinde oluşur. Zaman adımlarından dolayı oluşan olaylarında, zaman adımı sayısı sıfır olan olaylar kümesine eklenmesiyle, olay kümesi,

$$\Sigma^{\tau} := \Sigma \cup \left(\bigcup_{e \in \Sigma} \left(\bigcup_{\tilde{q} \in Q(e)} \varphi(e, \tilde{q}) \right) \right) \tag{3.6}$$

şeklinde oluşur. Son olarak, k: zaman adımlarını sayan zamanlayıcı, k_e: q_i durumunda e olayının meydana geldiği an (k değerine göre), $q_i \in X$, $e \in \Sigma$ ve $\tau_e \in \tau$ iken, kısmi geçiş fonksiyonu,

$$\delta^{\tau}(q_i, e, k) = \begin{cases} \delta(q_i, e) & \text{, eğer } k = k_e + \tau_e \text{ ve } \delta(q_i, e) \in X \text{ ise} \\ q^{e}_{i(n)} & \text{, eğer } k < k_e + \tau_e \text{ ve } \delta(q_i, e) \in X \text{ ise} \\ & \quad n = k - (k_e + \tau_e) \\ \text{tanımsız}, & \text{aksi durumda} \end{cases} \tag{3.7}$$

şeklinde tanımlanır.

Kabullenme 3.2: Fiziksel olarak imkansız dahi olsa, gerekli durumlarda, otomatadaki olaylardan bir veya birkaçının gerçek zamanlı gecikmesi 0 (sıfır) sn ise bu olaylar için zaman adımı sayısı 0 (sıfır) olarak alınır. Bu olay veya olaylar anlık olarak meydana geldikten sonra zaman adımı saymadan sistem, anlık olarak diğer duruma geçer. Eğer sistemdeki olaylardan bir veya birkaçının zaman gecikmesi birim zaman gecikmesi kadar ise bu olaylar için zaman adımı sayısı 1 olarak alınır. Bu olay veya olaylar meydana geldikten sonra sistem bir tane zaman adımı sayar ve diğer duruma geçer.

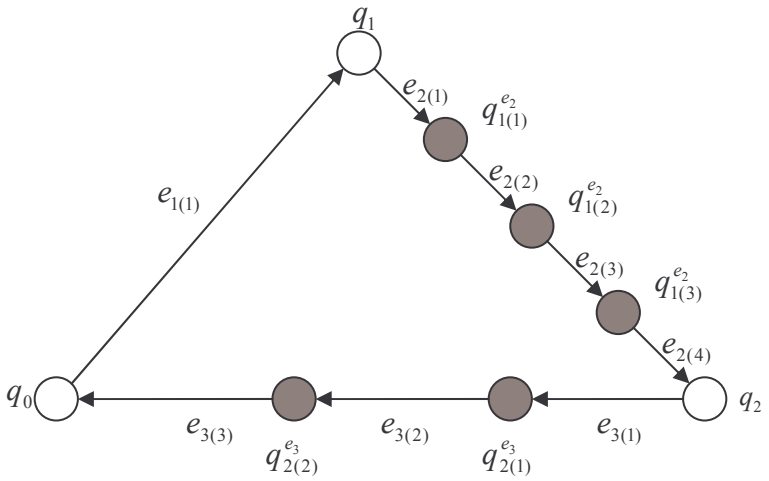
Kabullenme 3.3: Sistem gözlenebilir olmalıdır. Sistemin herhangi bir anda hangi durumda olduğu belirlenebilmelidir. Bu nedenle zaman adımı sayısı birden fazla olan olaylar meydana geldikten sonra ulaşıldığı varsayılan ara durumlar oluşturulur. Çünkü sistem istenilen zamanda bu ara durumlardan birinde olabilir.

Örneğin, Şekil 3.1'deki otomata için $D = \{0.05, 0.2, 0.15\}$ şeklinde alınmıştı. Buradan zaman gecikmelerinin en büyük ortak katı, $d_u = 0.05$ sn olduğu için (3.2) kullanılarak, $\tau = \{1, 4, 3\}$ şeklinde bulunur. Her olayın çıktı olarak bağlı olduğu durumların sayısı birdir ve bu durumların kümeleri $Q(e_1) = \{q_0\}$, $Q(e_2) = \{q_1\}$ ve $Q(e_3) = \{q_2\}$ şeklindedir.

Denklem (3.3) kullanılarak bulunan ara durumlar, $\lambda(e_1, q_0) = \phi$, $\lambda(e_2, q_1) = \{ q^{e_2}_{1(1)}, q^{e_2}_{1(2)}, q^{e_2}_{1(3)} \}$, $\lambda(e_3, q_2) = \{ q^{e_3}_{2(1)}, q^{e_3}_{2(2)} \}$ şeklindedir. Olaylara ait zaman gecikmelerinden dolayı ilave edilen olaylar ise, (3.4) kullanılarak,

$\varphi(e_1, q_0) = \{e_{1(1)}^{q_0}\}$, $\varphi(e_2, q_1) = \{e_{2(1)}^{q_1}, e_{2(2)}^{q_1}, e_{2(3)}^{q_1}, e_{2(4)}^{q_1}\}$, $\varphi(e_3, q_2) = \{e_{3(1)}^{q_2}, e_{3(2)}^{q_2}, e_{3(3)}^{q_2}\}$

şeklinde bulunur. Sistemin ilave edilen olaylar ve ara durumlarla birlikte gösterimi Şekil 3.2'deki gibidir (Bu gösterimde (3.5) ve (3.6) ile bulunan kümelerin tüm elemanlarına yer verilmiştir). Bu gösterimde $e \in \Sigma$ iken e olayına ait zaman adımları "$e_{(m)}$" şeklinde geçişlere eklenmiştir ($m = \{1,...,\tau_e\}$). Olayların bağlı olduğu durumlar açıkça görüldüğü için durumlar isimlendirmede kullanılmamıştır. Eğer olaya ait zaman adımı olmasaydı (olayın zaman gecikmesi sıfır olsaydı), geçişe sadece olayın adı "e" şeklinde eklenecekti (otomata gösterimindeki gibi). Ayrıca gösterimde, oluşan ara durumları diğer durumlardan kolaylıkla ayırt edilebilmek için "●" şeklinde gösterilmiştir.



Şekil 3.2. Şekil 4.1'deki otomata için zaman adımlarından dolayı oluşan ara durumlarında gösterildiği zamanlandırılmış otomata

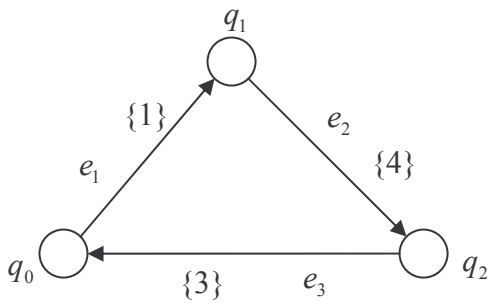
Sistemin çalışması ise, (3.7) de kullanılması ile aşağıdaki gibidir:

- q_0 durumunda e_1 olayı anlık olarak meydana gelir. Bundan sonra sistem bu olaya ait olan 1 zaman adımını saymaya başlar.
- 0.05 saniye sonunda sistem 1. zaman adımıyla q_0 durumundan q_1 durumuna geçer.

- q_1 durumunda e_2 olayı anlık olarak meydana gelir. Bundan sonra sistem bu olaya ait olan 4 zaman adımını saymaya başlar.
- 0.1 saniye sonunda sistem 1. zaman adımıyla q_1 durumundan $q_{1(1)}^{e_2}$ ara durumuna geçer.
- 0.15 saniye sonunda sistem 2. zaman adımıyla $q_{1(1)}^{e_2}$ ara durumundan $q_{1(2)}^{e_2}$ ara durumuna geçer.
- 0.2 saniye sonunda sistem 3. zaman adımıyla $q_{1(2)}^{e_2}$ ara durumundan $q_{1(3)}^{e_2}$ ara durumuna geçer.
- 0.25 saniye sonunda sistem 4. zaman adımıyla $q_{1(3)}^{e_2}$ ara durumundan q_2 durumuna geçer.
- q_2 durumunda e_3 olayı anlık olarak meydana gelir. Bundan sonra sistem bu olaya ait olan 3 zaman adımını saymaya başlar.
- 0.30 saniye sonunda sistem 1. zaman adımıyla q_2 durumundan $q_{2(1)}^{e_3}$ ara durumuna geçer.
- 0.35 saniye sonunda sistem 2. zaman adımıyla $q_{2(1)}^{e_3}$ durumundan $q_{2(2)}^{e_3}$ ara durumuna geçer.
- 0.40 saniye sonunda sistem 3. zaman adımıyla $q_{2(2)}^{e_3}$ durumundan q_0 durumuna geçer.

Sistem, bu şekilde 0.05 sn zaman aralıklarını saymaya devam ederek, istenilen zaman aralığında çalışır.

Şekil 3.1'deki üç tane durum ve üç tane olaya sahip basit otomataya uygulanan yöntemle, Şekil 3.2'deki sekiz tane durum ve sekiz tane olaya sahip bir gösterim elde edilmiştir. Buradan da açıkça anlaşılıyor ki aynı modelleme yönteminin uygulanan sistem büyüdükçe gösterimi daha da karmaşık hale gelecektir (bkz. Şekil 3.5, Şekil 3.6). Bu nedenle, Şekil 3.2'deki gösterimden oluşan ara durumlar ve zaman adımlarından dolayı oluşan geçişler çıkartılıp olaylara ait zaman adımlarının sayısını veren τ kümesinin elemanları her bir geçişe $\{\tau_e\}$ şeklinde eklenirse Şekil 3.3'deki zamanlandırılmış otomata modeli oluşturulmuş olur.

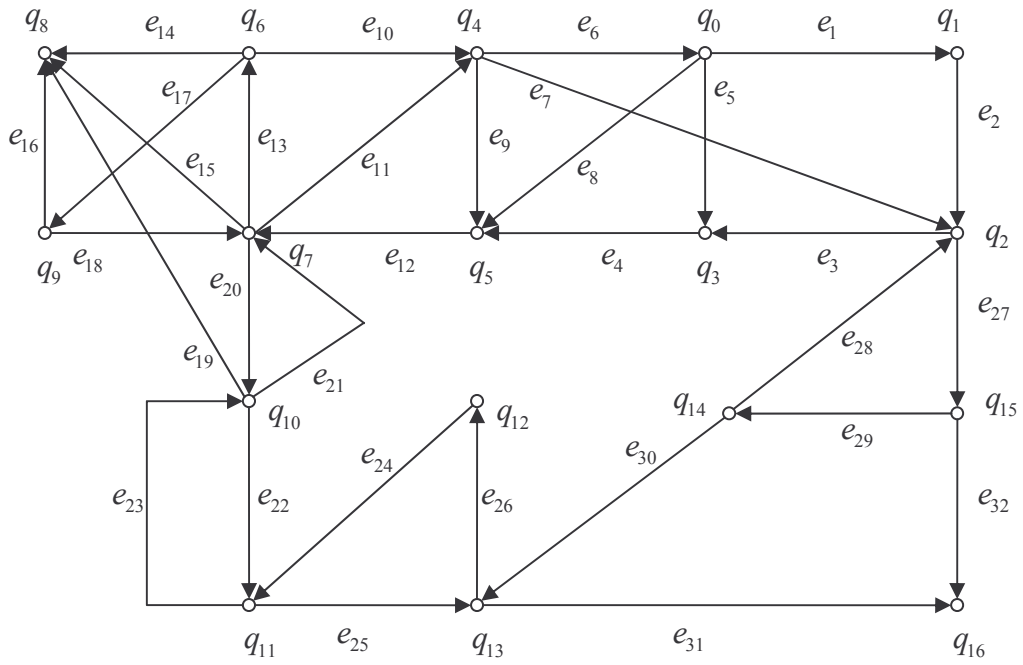


Şekil 3.3. Şekil 4.1'deki otomata için zaman adımı yaklaşımı ile zamanlandırılmış otomata modeli

Şekil 3.3'deki grafiksel gösterim kullanılarak, eklenen zaman adımları haricinde tamamen otomataya benzeyen bir grafiksel bir gösterim elde edilmiştir. Bu gösterim daha büyük sistemler için kolaylık sağlamaktadır.

Örneğin, otomata modeli Şekil 3.4 deki gibi olan sistem, zaman gecikmeleri, *D,* eklenerek zamanlandırılmış sisteme çevrilmiş, ara durumlar ve zaman adımlarından dolayı oluşan geçişler de eklenerek Şekil 3.5'deki model ortaya çıkarılmıştır. Daha sonra ara durumlar ve zaman adımlarından dolayı oluşan geçişler çıkarılarak, Şekil 3.6'daki zaman adımı ile zamanlandırılmış otomata gösterimi elde edilmiştir.

Şekil 3.4'deki sistem için $X = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9, q_{10}, q_{11}, q_{12}, q_{13}, q_{14}, q_{15}, q_{16}\}$, $\Sigma = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}, e_{12}, e_{13}, e_{14}, e_{15}, e_{16}, e_{17}, e_{18}, e_{19}, e_{20}, e_{21}, e_{22}, e_{23}, e_{24}, e_{25}, e_{26}, e_{27}, e_{28}, e_{29}, e_{30}, e_{31}, e_{32}\}$, $x_0 = q_0$, $D = \{2,3,1,4,2,3,0,5,1,3,2,2,1,4,2,2,0,3,0,2,1,3,2,5,4,1,1,5,2,3,5,3\}$ şeklindedir. $d_u = 1$ sn olduğu için (3.2) kullanılarak olaylara ait zaman adımlarının sayısını veren küme $\tau = \{2,3,1,4,2,3,0,5,1,3,2,2,1,4,2,2,0,3,0,2,1,3,2,5,4,1,1,5,2,3,5,3\}$ şeklinde oluşturulur.



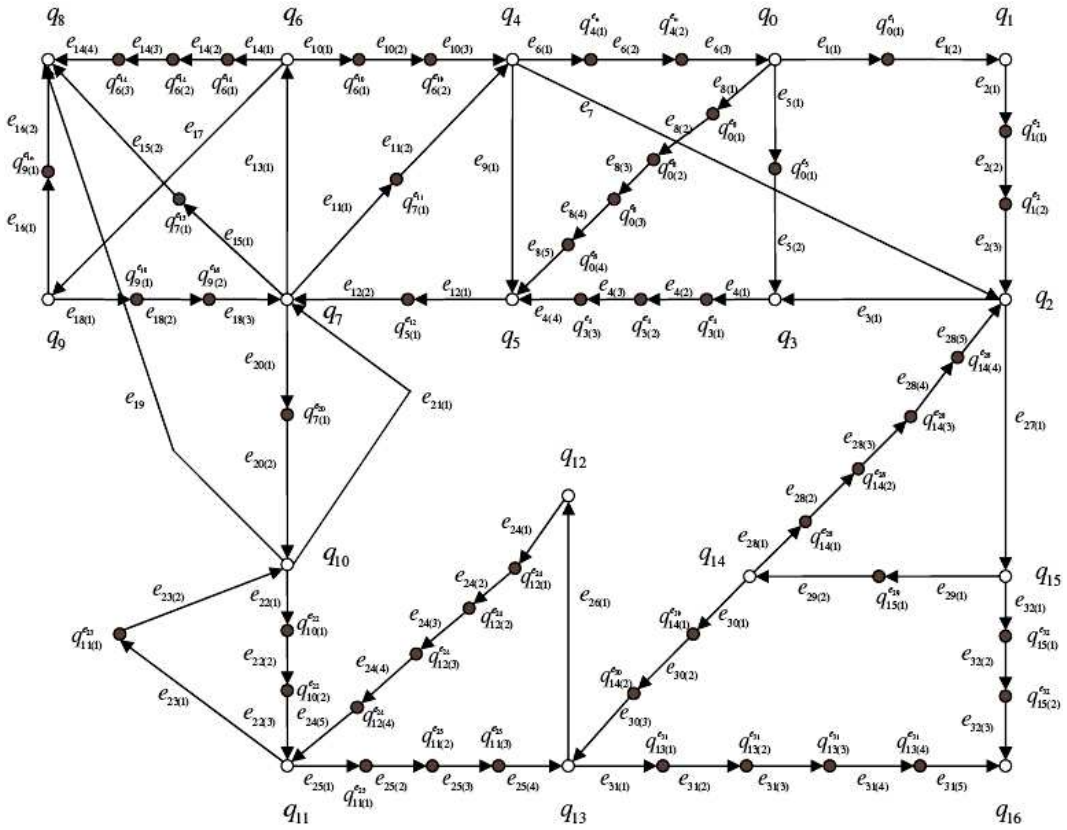
Şekil 3.4. [5]'dan alınan otomata modeli

Denklem (3.3-3.7) kullanılarak Şekil 3.5'deki gösterim oluşturulmuştur. Bu gösterimde örneğin, q_0 durumundaki geçerli olay e_1 ve bu olay için geçerli zaman adımı sayısı 2'dir. e_1 olayı meydana geldikten sonra sistem q_1 durumuna geçer. q_1, durumundaki geçerli olay da e_2 ve bu olay için geçerli zaman adımı sayısı 3'dür. Sistemin 5 saniye çalışması istenmesi halinde sistemde meydana gelen geçişler aşağıdaki gibidir:

- q_0 durumunda e_1 olayı anlık olarak meydana gelir. Bundan sonra sistem bu olaya ait olan 2 zaman adımını saymaya başlar.
- 1. saniye sonunda sistem 1. zaman adımıyla q_0 durumundan $q_{0(1)}^{e_1}$ ara durumuna geçer.
- 2. saniye sonunda sistem 2. zaman adımıyla $q_{0(1)}^{e_1}$ ara durumundan q_1 durumuna geçer.
- q_1 durumunda e_2 olayı anlık olarak meydana gelir. Bundan sonra sistem bu olaya ait olan 3 zaman adımını saymaya başlar.
- 3. saniye sonunda sistem 1. zaman adımıyla q_1 durumundan $q_{1(1)}^{e_2}$ ara durumuna geçer.

- 4. saniye sonunda sistem 2. zaman adımıyla $q_{1(1)}^{e_2}$ ara durumundan $q_{1(2)}^{e_2}$ ara durumuna geçer.
- 5. saniye sonunda sistem 3. zaman adımıyla $q_{1(2)}^{e_2}$ ara durumundan q_2 durumuna geçer.

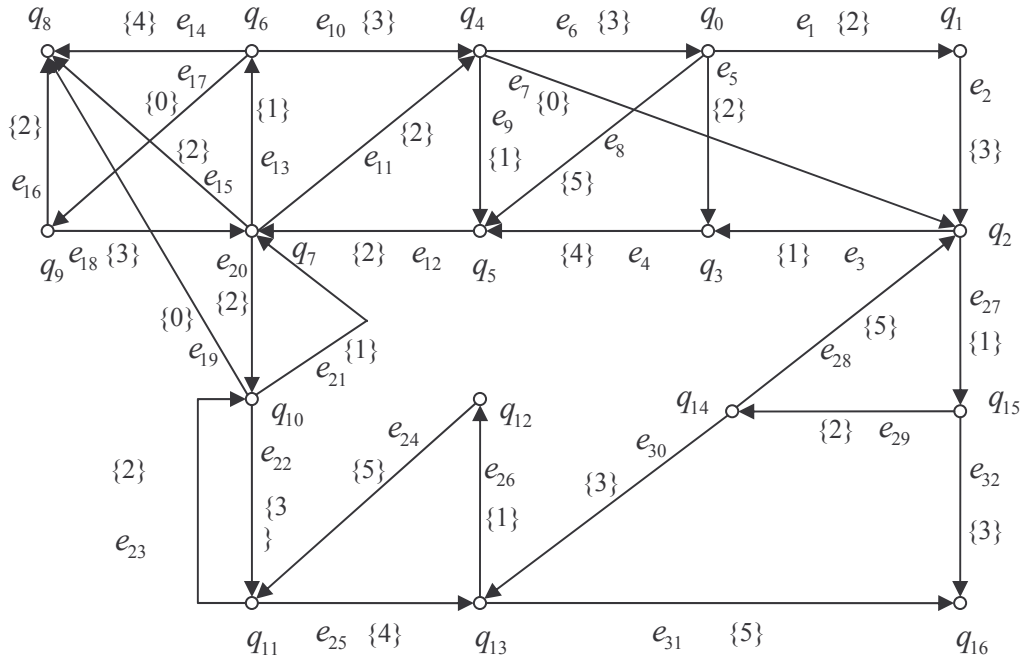
Bu sistem için birim gecikme zamanı 1 sn olduğu için, her 1 sn için bir zaman adımı sayılmıştır.



Şekil 3.5. Şekil 4.4'deki otomata için zaman adımlarından dolayı oluşan ara durumların da gösterildiği zamanlandırılmış otomata

Şekil 3.2 ve Şekil 3.5'deki gösterimler karşılaştırıldığında, sistem büyüdükçe ara durumların ve zaman adımlarının gösterilmesinin modeli daha karmaşık ve anlaşılması zor hale getirdiği açıkça görülüyor. Bu nedenle, Şekil 3.5'deki gösterimden oluşan ara durumları çıkartılır ve olaylara ait zaman

adımlarının sayısını veren τ kümesinin elemanları her bir geçişe eklenirse Şekil 3.6'daki zamanlandırılmış otomata modeli oluşturulmuş olur. Böylece anlaşılması daha kolay bir zamanlandırılmış model elde edilmiş olur.



Şekil 3.6. Zaman adımı ile zamanlandırılmış otomata modeli

3.3. Hazırlanan Algoritmalar

Bu bölümde, tanımlanan zamanlandırılmış otomata modeli için hazırlanan algoritmalar ham-kodlar (pseudo-codes) kullanılarak verilmiştir. Hazırlanan algoritmalar beş ana başlık altında incelenebilir. i) Otomataya zaman adımlarının ilave edilmesi ile oluşturulan zamanlandırılmış otomata hakkında bilgi veren algoritma ii) Belirlenen zaman aralığında otomatanın benzetimini yapan algoritma. iii) Sistem çıkmazını bulan algoritma. iv) Sistem çıkmazının meydana gelmesinin engellendiği otomatanın belirlenen zaman aralığında benzetimini yapan algoritma. v) İstenilen durumların bulunduğu en kısa durum dizisini ve en kısa sürede oluşan durum dizisini bulan algoritma. Hazırlanan tüm algoritmalar, temel olarak, A^{τ} ile gösterilen otomatanın tanımını ve Γ ile gösterilen otomata matrisini (sistemin çıkmaza girmesin engelleyerek, istenilen zaman aralığında

çalışmasının benzetimini yapan algoritma haricinde) girdi olarak kullanmaktadırlar.

3.3.1. Otomataya zaman adımlarının ilave edilmesi ile oluşturulan zamanlandırılmış otomata hakkında bilgi veren algoritma

EK-1A'da verilen algoritma, öncelikle sisteme ait birim zaman gecikmesini (d_u) bulur ve τ ile gösterilen olaylara ait zaman adımı sayılarını veren vektörü oluşturur. Daha sonra sistemde tüm durumlarda meydana gelebilecek olayları, olaylara ait zaman adımı sayılarını ve olayların meydana gelmesi ile oluşan tüm geçişleri bulur ve kullanıcının görebilmesi sırasıyla ekrana yazar. Algoritmada oluşturulan U matrisi, Γ matrisinde olayların yerlerine olaylara zaman adımı sayıları yazılarak oluşturulur. Böylece Γ matrisindeki adresleme bilgileri aynen kullanılarak zaman adımı sayılarına erişim kolaylaştırılmıştır.

3.3.2. Belirlenen zaman aralığında otomatanın benzetimini yapan algoritma

EK-1B'de verilen algoritma, A^{τ} ve Γ girdileri dışında, kullanıcının klavye aracılığı ile yazdığı ve *input*, *input*1 ve *input*2 ile adlandırılan girdileri kullanmaktadır. *input* değişkeni kullanıcı tarafından girilen sistemin çalışacağı zaman aralığını, *input*1 değişkeni sadece algoritma ilk çalıştırıldığında kullanıcının istediği başlangıç durumunu, *input*2 değişkeni ise süre bitene kadar kullanıcının girdiği olayları tutar. Algoritma kullanıcının istediği süre bitene kadar ya da sistem çıkmaza girene kadar çalışır ve her seferinde kullanıcıdan bir olay girmesini ister. Böyle çalışmaya devam ederken meydana gelen geçişleri ekrana yazar. Ayrıca geçilen durumları R vektörüne, meydana gelen olay ve zaman adımlarını T vektörüne yazar.

3.3.3. Sistem çıkmazını bulan algoritma

EK-1C'de verilen algoritma, öncelikle Γ_{new} matrisi (başlangıçta Γ matrisine eşit) oluşturur. Daha sonra sistem çıkmazı oluşan durumları, sistemi bu duruma götüren olayları ve olayların meydana gelmesi ile oluşan geçişleri bulur ve ekrana yazar. Γ_{new} matrisi, sistem çıkmazının oluştuğu durumlara götüren olayların

yerine sıfır atayarak günceller. Son olarak Γ_{new} matrisinde, sistem çıkmazı meydana gelen durumların satırları haricinde, tümü sıfır olan başka satırların varlığını araştırır. Eğer var ise, sistemi bu duruma götüren olayları ve olayların meydana gelmesi ile oluşan geçişleri ekrana yazar ve o duruma götüren olayları da, Γ_{new} matrisinde, sıfır atayarak günceller. Bu işlemi tümü sıfır olan farklı bir satır bulamayana kadar devam eder.

3.3.4. Sistem çıkmazının meydana gelmesinin engellendiği otomatanın belirlenen zaman aralığında benzetimini yapan algoritma

EK-1D'de verilen algoritma, EK1-B'de verilen algoritmadan farklı olarak Γ matrisi yerine, Γ_{new} matrisini kullanmaktadır. Böylece sistemin çıkmaza girmeden süre bitene kadar çalışması sağlanmış olur. Bunun dışında algoritmalar aynı şekilde çalışır.

3.3.5. İstenilen durumların bulunduğu en kısa durum dizisini ve en kısa sürede oluşan durum dizisini bulan algoritma

EK-1E'de verilen algoritma, A^{τ} ve Γ girdileri dışında, kullanıcının klavye aracılığı ile yazdığı ve *input*, *input*1 ile adlandırılan girdileri kullanmaktadır. *input* değişkeni kullanıcı tarafından istenilen durum sayını, *input*1 değişkeni ise kullanıcının istediği durumları sırasıyla tutar. Algoritma, istenilen durum sayısı ile doğru orantılı miktarda, başlangıç durumundan başlayarak olabilecek tüm durum dizilerini oluşturur ve bu diziler oluşurken meydana gelen zaman adımı sayılarını toplar. Oluşturulan durum dizilerini *vek* hücre yığınına (cell array), bu diziler oluşurken meydana gelen toplam zaman adımı sayılarını da *time_vek* vektörüne kaydeder. Daha sonra bu dizilerden en kısa olanını ve en kısa sürede oluşanını bulur. Eğer iki dizi de aynı dizi ise diziye ait durumları, bu durumlara götüren olayları ve olaylara ait zaman adımlarını ekrana yazar. Eğer iki dizi farklı ise dizilere ait durumları, bu durumlara götüren olayları ve olaylara ait zaman adımlarını ayrı ayrı ekrana yazar.

3.4. Hazırlanan Yazılımlar

Bir önceki bölümde ham-kod olarak sunulan algoritmalar Matlab kullanılarak yazılımlar haline dönüştürülmüştür. RUN_export.m (bkz. EK-2A) yazılımı tüm algoritmaları çalıştırabilecek şekilde tasarlanan görsel bir ara yüzdür. Yazılım ilk çalıştırıldığında Şekil 3.7'deki ara yüz açılır ve YÜKLE isimli buton haricinde bütün butonlar kullanılamaz haldedir.

Şekil 3.7. Yazılım ilk olarak çalıştırıldığı zaman açılan ara yüz

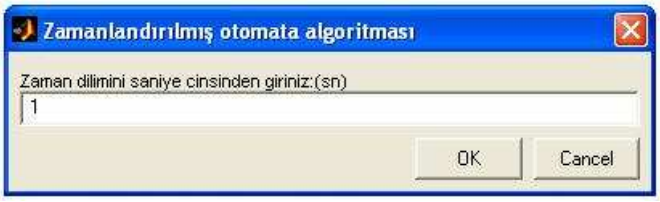
İlk olarak YÜKLE butonuna basılır. Şekil 3.8'deki ara yüz açılır ve zamanlandırılmış otomatayı tanımlayan ve kullanıcı tarafından oluşturulan dosyanın adının girilmesini kullanıcıdan ister. Bu girdi dosyası olaylara ait gerçek zaman gecikmelerini (D kümesini), Γ ile gösterilen otomata matrisini [17] olayların ve durumların isimlendirilmesinde kullanılan vektörleri kapsamaktadır (bkz. EK-2B). Bu dosya seçildiği zaman ilk olarak, EK-1A'da algoritması verilen sistem.m yazılımı (bkz. EK-2C) çalışır. Bu yazılım çalışırken ekrana yazılan tüm bilgiler sistem_bilgi.m dosyasına (bkz. EK-2C) kaydedilir ve kullanıcının görebilmesi için açılır. Bu dosya sistemde hangi durumda hangi olay veya olayların meydana gelebileceğini, bu olaylara ait zaman adımı sayısını, oluşan ara durumları ve sistem çıkmazı oluşan durumlar hakkında, kısacası sistem hakkında kullanıcıya tam bir bilgi verir. Daha sonra Şekil 3.9'deki ara yüz açılır. BENZETİM-2 butonu haricinde tüm butonlar kullanılır hale gelmiştir.

Şekil 3.8. Kullanıcıdan zamanlandırılmış otomatayı tanımlayan dosyayı açması istenilen ara yüz

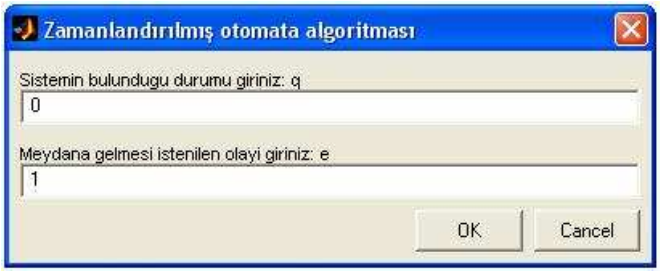
Şekil 3.9. YÜKLE butonuna basılıp çalıştırılan yazılımdan sonra açılan ara yüz

BENZETİM-1 butonu sırası ile EK-1B'de algoritması verilen algoritma.m (bkz. EK-2D) ve T ve R vektörlerinin içeriğini dosyaya kaydeden cikis.m (bkz. EK-2D) yazılımlarını çalıştırır. BENZETİM-1 butonuna basıldığında Şekil 3.10'daki ara yüz açılır. Burada kullanıcıdan sistemin çalışacağı zaman aralığı istenir. Örnek olarak zaman 10 saniye alınsın.

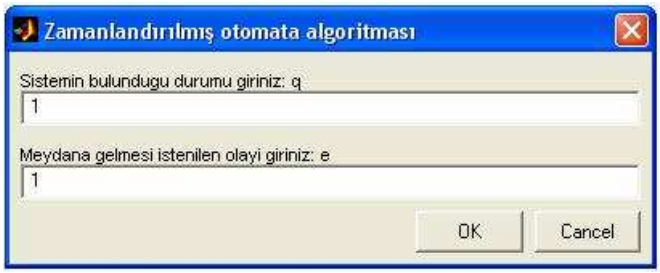
Bundan sonra Şekil 3.11'deki ara yüz açılır ve kullanıcıdan başlangıç durumunu ve meydana gelecek olayı ister. İlk değerler başlangıçta q_0 ve e_1 olarak verilmiştir. Tamam tıklandığında Şekil 3.12'daki ara yüz açılır.



Şekil 3.10. Kullanıcıdan sistemin çalışacağı zaman aralığının istenildiği ara yüz

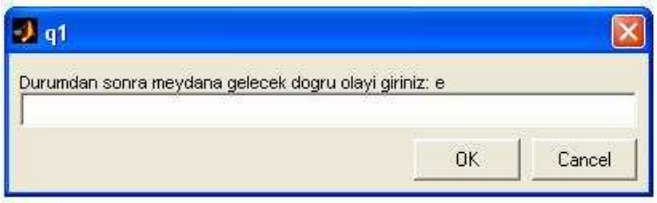


Şekil 3.11. Kullanıcıdan başlangıç durumunun ve meydana gelecek olayın istenildiği ara yüz



Şekil 3.12. Kullanıcıdan meydana gelecek olayların istenildiği ve sistemin bulunduğu durumu gösteren ara yüz

Görüldüğü gibi her seferinde durum satırı sistemin geçtiği durum atanarak güncellenir ve kullanıcıdan geçerli bir olay ister. Bundan sonra sadece meydana gelecek olaylar değiştirilmelidir. Bu döngü zaman bitene kadar veya sistem çıkmaza girene kadar devam eder. Eğer bu arada herhangi bir durumdan sonra geçersiz bir olay meydana gelmesi istenirse Şekil 3.13'deki ekran görünür ve kullanıcıdan bu durumdan sonraki geçerli olayı girmesi istenir. Tekrar geçersiz bir olay girilirse program sonlandırılır.



Şekil 3.13. Kullanıcıdan meydana gelecek doğru olayın girilmesinin istenildiği ara yüz

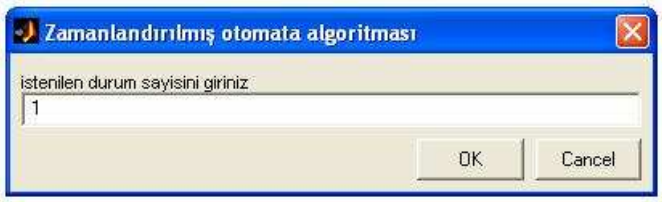
Süre bittiği zaman ekrana yazılan bilgilerin tutulduğu bilgi.m dosyası (bkz. EK-2D) ile bu süre içinde oluşan durumlar ve meydana gelen olayların kaydedildiği, T ve R vektörlerinin içeriğinin yazıldığı bilgi1.m dosyası (bkz. EK-2D) oluşturulur ve kullanıcının görebilmesi için açılır.

SİSTEM ÇIKMAZI & KONTROLÖR butonu EK-1C'de algoritması verilen deadlock.m yazılımını (bkz. EK-2E) çalıştırır. Bu yazılım da ekrana yazılan bilgilerin kaydedildiği deadlock_bilgi.m dosyasını (bkz. EK-2E) oluşturur ve kullanıcının görebilmesi için açar. Bu yazılım sistemde eğer varsa sistem çıkmazı oluşan durumları ve bu durumları engellemek için meydana gelmemesi gereken olaylar hakkında kullanıcıya bilgi verir. Bundan sonra BENZETİM-2 butonuda aktif hale gelir ve görünüm Şekil 3.14'deki gibi olur.

BENZETİM-2 butonu sırasıyla EK-1D'de algoritması verilen controlled_algoritma.m (bkz. EK-2F) ve T ve R vektörlerinin içeriğini dosyaya kaydeden controlled_cikis.m (bkz. EK-2F) yazılımlarını çalıştırır. Bu yazılımlar algoritma.m ve cikis.m yazılımları gibi çalışırlar (BENZETİM-1 butonuna basıldığı gibi). Fakat kullanılan otomata matrisi değiştirilmiştir. girdi.m dosyasındaki Gama matrisi yerine artık Gama_new matrisi (bkz. EK-2F) kullanılmaktadır. Bu matriste sistemi çıkmaza götüren olaylar kullanılamaz hale getirilmiştir. Böylece sistemin çıkmaza girmeden verilen süre içinde çalışması garanti altına alınmıştır. Gama_new matrisi dışında yazılım aynı şekilde çalışır ve aynı ara yüzler görünür (Şekil 3.10–3.13). Verilen süre bittiği zaman ekrana yazılan bilgilerin tutulduğu controlled_bilgi.m dosyası (bkz. EK-2F) ile bu süre içinde oluşan durumlar ve meydana gelen olayların kaydedildiği, T ve R vektörlerinin içeriğinin yazıldığı controlled_bilgi1.m dosyası (bkz. EK-2F) oluşturulur ve kullanıcının görebilmesi için açılır.

Şekil 3.14. SİSTEM ÇIKMAZI & KONTROLÖR butonuna basılıp çalıştırılan yazılımdan sonra açılan ara yüz

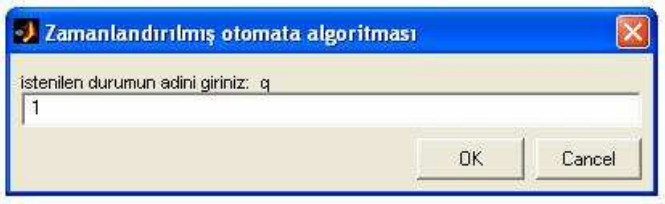
İSTENİLEN DURUM KONTROLÖRÜ butonu EK-1E'de algoritması verilen desired.m yazılımını (bkz. EK-2G) çalıştırır. Butona basıldığı zaman Şekil 3.15'deki ara yüz açılır.



Şekil 3.15. Kullanıcıdan durum sayısının istenildiği ara yüz

Şekil 3.15'deki ara yüzde kullanıcıdan durumların sayısını ister. Örnek olarak alınan sistem için durum sayısı 5 seçilsin. 5 yazılıp tamam tıklandığında Şekil 3.16'daki ekran açılır. Bu ara yüze kullanıcı istediği durumları girer. Örnek olarak alınan sistem için istenilen durumlar q13, q9, q7, q4 ve q15 olsun. Şekil 3.16'daki ekrana 13 yazılıp tamam tıklansın. Bundan sonra Şekil 3.16'daki ara yüz dört kere daha açılacaktır. Açılan her yeni ara yüze istenilen durumlar yazılır. Bu işlem yapılırken sıralı yapılması önemli değildir. En son durum yazıldıktan sonra yazılım bu istenilen durumların bulunduğu en kısa durum dizisini ve en kısa sürede oluşan durum dizisini bulur. Ayrıca bu işlem yapılırken meydana gelen olay dizisini ve geçen zamanı (saniye cinsinden ve zaman adımı sayısı cinsinden)

bulur. Tüm bu bilgileri desired_bilgi.m dosyasına (bkz. EK-2G) yazar ve kullanıcının görebilmesi için açar.



Şekil 3.16. Kullanıcıdan durumların istenildiği ara yüz

6. SONUÇLAR

Bu çalışmada, öncelikle kesikli olay sistemlerinin modelleme yöntemlerinden otomata hakkında bilgi verilmiştir. Matematiksel gösterimleri, örnekler verilerek sunulmuştur. Daha sonra, zaman gecikmelerinin olayların meydana gelmesi ile ilişkilendirildiği yeni bir matematiksel model sunulmuştur.

Bu modelde, verilen diğer çalışmalardan farklı olarak, olayların meydana geliş sürelerine önem verilmiş ve durumların anlık olarak oluştuğu kabullenilmiştir. Zaman gecikmelerinin en büyük ortak böleni bulunup, zaman gecikmeleri ölçeklenerek sayma sayıları haline dönüştürülmektedir. Elde edilen her bir sayma sayısı ilgili olay için zaman adımı olarak adlandırılmaktadır. Zaman adımı kullanılıp, yeni durum ve olaylar ilave edilerek, zamanlandırılmış otomata gösterimi, klasik (zamanlandırılmamış) otomata gösterimi haline getirilmektedir. Böylece, ele alınan zamanlandırılmış otomata daha anlaşılabilir olmaktadır.

Bu modelleme yönteminin herhangi bir otomataya uygulanabilmesi için, sistemin gözlenebilir olması ve tüm olaylara ait zaman gecikmelerinin tam değerinin bilinmesi gerekmektedir.

Zaman adımı yaklaşımı kullanılarak elde edilen yeni model için sistem çıkmazını bulan, meydana gelmesini önleyen ve istenilen durumların en kısa sürede meydana gelmesini sağlayan algoritmalar da bu tezde ele alınmıştır. Bu algoritmalar, Matlab kullanılarak yazılımlar haline getirilmiştir.

Gelecekteki çalışmalarda, aynı yöntem kullanılarak, sistemin durumlarda beklediği süre de modellenebilir. Hem olayların gecikmeleri hem de durumlarda beklediği süre modellenerek daha gerçeğe yakın bir zamanlandırılmış model oluşturulabilir.

KAYNAKLAR

[1] Aybar, A., *Petri Ağlarda Örtüşmeli Ayrıştırma ve Genleştirme Kullanılarak Kontrolör Tasarımı*, Doktora Tezi, Anadolu Üniversitesi, Fen Bilimleri Enstitüsü, Eskişehir, 2001.

[2] Thiele, L., *Discrete Event Systems: Introduction*, 2005. http://www.tik.ee.ethz.ch/tik/education/lectures/DES/Book/des_book_intro.pdf

[3] Ramadge, P.J.G. ve Wonham, M., "The Control of Discrete Event System," *Proceedings of the IEEE*, **77**, 81-98, 1989.

[4] Ramadge, P.J.G. ve Wonham, M., "Supervision of discrete event systems," *Proceedings of the IEEE*, **37**, 1692-1708, 1982.

[5] Aybar, A. ve İftar, A., "Decentralized Supervisory Controller Design for Discrete-Event Systems using Overlapping Decompositions and Expansions," *Dynamics of Continuous, Discrete and Impulsive Systems (Series B)*, **11**, 553-568, 2004.

[6] Sudkamp, T.A., *Languages and Machines: An Introduction to the Theory of Computer Science*, Addison-Wesley Publishing Co., Boston, A.B.D., 1997.

[7] Wonham, W.M., *Notes on Control of Discrete Event Systems*, Systems Control Group, University of Toronto, Dept. of Electrical and Computer Engineering, Kanada, 2003.

[8] Hopcroft, J., Motwani R., ve Ullman, J.D., *Introduction to Automata Theory: Languages, and Computation*, Addison-Wesley Publishing Co., Massachusetts, A.B.D., 2001.

[9] Willner, Y. ve Heymann, M., "On Supervisory Control of Concurrent Discrete-Event Systems," *International J. on Control*, **54**, 1119-1142, 1991.

[10] Chandra, V. ve Kumar, R., "A New Formalism and Automata Model Generator for a Class of Discrete Event Systems," *2001 American Control Conference*, VA, 4562-4567, 2001.

[11] Alur, R. ve Dill, D.L., "Automata for Modeling Real-time Systems," *Proceedings of the ICALP'90*, 443, 322-335, 1990.

[12] Alur, R. ve Dill, D.L., "A Theory of Timed Automata," *Theoretical Computer Science*, **126**, 183–235, 1994.

[13] Brandin, B.A. ve Wonham, W.M., "Supervisory Control of Timed Discrete Event Systems," *IEEE Transactions on Automatic Control*, **39**, 329-341, 1994.

[14] Zad, S.H., Kwong, R.H. ve Wonham, W.M., "Fault Diagnosis in Timed Discrete-Event Systems," *Proceedings of the 38th Conference on Decision and Control*, FM03, 1756-1761 , 1999

[15] Bouyer, P., *Timed Automata: From Theory to Implementation*, 2005. http://www.lsv.ens-cachan.fr/~bouyer/files/bouyer_chennai.pdf

[16] Yovine, S., "Compiling Timed Algebras into Timed Automata." *Proceedings of the XVIII Conf. Latinoamericana de Informática*, PANEL'92, 1243-1250, 1992.

[17] Aybar A., Polat Ç. ve Atasoy F., "Algorithms for Deadlock Avoidance and Reversibility Enforcement in Discrete Event Systems," *Dynamics of Continuous, Discrete and Impulsive Systems*, Series A, (Basım aşamasında)

EK-1A Otomataya zaman adımlarının ilave edilmesi ile oluşturulan zamanlandırılmış otomata hakkında bilgi veren algoritma

Load A^{τ}, Γ
$d_u = e.b.o.b(D)$
$\tau = D / d_u$
For $i = 1$ to $|X|$
  For $j = 1$ to $|X|$
    If $\Gamma(i, j) \neq 0$ Then
      $U(i, j) = [\tau]_{\Gamma(i,j)}$
    End
  End
End
E-yaz ("birim zaman gecikmesi d_u saniyedir")
For $i = 1$ to $|X|$
  If $\Gamma(i,:) = \phi$ Then
    E-yaz ("$[X]_i$ durumunda meydana gelebilecek geçerli bir olay yoktur")
  Else
    For $j = 1$ to $|X|$
      If $\Gamma(i, j) \neq 0$ Then
        $\tau_e = U(i, j)$
      End
      $e = [\Sigma]_{\Gamma(i,j)}$
      E-yaz ("$[X]_i$ durumunda iken e olayı meydana gelince oluşan geçişler aşağıdadır")
      E-yaz ("e olayı geçerli olan zaman adımı τ_e 'dir")
      If τ_e =0 Then
        E-yaz ("$[X]_i$ --(e)--> $[X]_j$")
      ElseIf τ_e =1 Then
        E-yaz ("$[X]_i$ --(za)--> $[X]_j$")
      Else
        E-yaz ("$[X]_i$ --(za)--> $[\lambda(e)]_1$")
        For $k = 1$ to (τ_e -2)
          E-yaz ("$[\lambda(e)]_k$ --(za)--> $[\lambda(e)]_{k+1}$")
        End
        E-yaz ("$[\lambda(e)]_{k+1}$ --(za)--> $[X]_j$")
      End
    End
  End
End

EK-1B Belirlenen zaman aralığında otomatanın benzetimini yapan algoritma

Load A^{τ}, Γ
$sure = input$
$sure1 = sure / d_u$
$time = 0$
$a = 1$
$b = 1$
$x = q_0$
$e = e_1$
$k = input1$
$x = [X]_k$
$[R]_a = x$
$a = a + 1$
ATLA: Devam Et
Do Loop Döngü-1
$x = [X]_k$
$e = [\Sigma]_{input2}$
If $\Gamma(k,:) = \phi$ Then
    E-yaz ("x durumundan sonra e olayı ateşlenemez")
    Exit Loop Döngü-1
End
If $x \notin Q(e)$ Then
    $e = [\Sigma]_{input2}$ (kullanıcıdan geçerli bir olay iste)
    If $x \notin Q(e)$ Then
        Exit Loop Döngü-1
    Else
        If $\tau_e = 0$ Then
            $[T]_b = e$
            $b = b + 1$
            For $i = 1$ to $|X|$
                If $\Gamma(k,i) = input2$ Then
                    E-yaz ("sistem $[X]_k$ durumundan sonra e olayı meydana gelince $[X]_i$ durumuna gecikme olmadan geçer")
                    E-yaz ("$[X]_k$ --(e)--> $[X]_i$")
                    $k = i$
                    $[R]_a = [X]_i$
                    $a = a + 1$
                Else
                    Devam Et
                End
            End

ElseIf $sure1 > 0$ & $\tau_e = 1$ Then

If $t = sure1$ Then

E-yaz (“Süre bitti”)

Exit Loop Döngü-1

Else

$[T]_b = e$

$b = b + 1$

If $\Gamma(k,i) = input2$ Then

E-yaz (“sistem $[X]_k$ durumundan sonra 1 zaman adımıyla $[X]_i$ durumuna geçer”)

E-yaz (“$[X]_k$ --(za)--> $[X]_i$ ”)

$k = i$

$[R]_a = [X]_i$

$a = a + 1$

$[T]_b = \text{za}$

$b = b + 1$

$t = t + 1$

End

ElseIf $\tau_e > 1$ Then

If $t = sure1$ Then

E-yaz (“Süre bitti”)

Exit Loop Döngü-1

Else

$[T]_b = e$

$b = b + 1$

For $i = 1$ to $|X|$

If $\Gamma(k,i) = input2$ Then

E-yaz (“sistem $[X]_k$ durumundan sonra 1 zaman adımıyla $[\lambda(e)]_l$ durumuna geçer”)

E-yaz (“$[X]_k$ --(za)--> $[\lambda(e)]_l$ ”)

$[R]_a = [\lambda(e)]_l$

$a = a + 1$

$[T]_b = \text{za}$

$b = b + 1$

For $k = 1$ to (τ_e -2)

If $t = sure1$ Then

E-yaz (“Süre bitti”)

Exit Loop Döngü-1

Else

E-yaz (“sistem $[\lambda(e)]_k$ durumundan sonra k+1 zaman adımıyla $[\lambda(e)]_{k+1}$ durumuna geçer”)

E-yaz (“$[\lambda(e)]_k$ --(za)--> $[\lambda(e)]_{k+1}$ ”)

$[R]_a = [\lambda(e)]_{k+1}$
$a = a + 1$
$[T]_b = \text{za}$
$b = b + 1$
End
End
E-yaz ("sistem $[\lambda(e)]_{k+1}$ durumundan
sonra τ_e zaman adımıyla $[X]_i$ durumuna geçer")
E-yaz ("$[\lambda(e)]_{k+1}$ --(za)--> $[X]_i$ ")
$k = i$
End
End
Go To ATLA
End
End
End
End Loop Döngü-1

EK-1C Sistem çıkmazını bulan algoritma

Load A^{τ}, Γ
$\Gamma_{new} = \Gamma$
$line = [\ ]$
For $i = 1$ to $|X|$
  If $\Gamma(i,:) = \phi$ Then
    Ekrana yaz ("$[X]_i$ durumunda sistem çıkmazı oluşuyor")
    $[line]_i = i$
    For $j = 1$ to $|X|$
      If $\Gamma(j,i) \neq 0$ Then
        $event = \Gamma(j,i)$
        $e = [\Sigma]_{event}$
        Ekrana yaz("$[X]_j$ durumunda e olayının meydana gelmesi engellenmeli")
        If τ_e =0 Then
          Ekrana yaz ("$[X]_j$ --(e)--> $[X]_i$ ")
        ElseIf τ_e =1 Then
          Ekrana yaz ("$[X]_j$ --(za)--> $[X]_i$ ")
        Else
          Ekrana yaz ("$[X]_j$ --(za)--> $[\lambda(e)]_1$ ")
          For $k = 1$ to (τ_e -2)
            Ekrana yaz ("$[\lambda(e)]_k$ --(za)--> $[\lambda(e)]_{k+1}$ ")
          End
          Ekrana yaz ("$[\lambda(e)]_{k+1}$ --(za)--> $[X]_i$ ")
        End
        $\Gamma_{new}(j,i) = 0$
      End
    End
  End
End
For $m = 1$ to $|\Sigma|$
  For $i = 1$ to $|X|$
    If $\Gamma(i,:) = \phi$ Then
      $[line\_new]_i = i$
    Else
      Go To ATLA
    End
  End

If $line \neq line\_new$ Then
$z = line\_new - line$
For $i = 1$ to $|X|$
If $[z]_i \neq 0$ Then
For $j = 1$ to $|X|$
If $\Gamma(j,i) \neq 0$ Then
$event = \Gamma(j,i)$
$e = [\Sigma]_{event}$
Ekrana yaz("$[X]_j$ durumunda e olayının meydana gelmesi engellenmeli")
$tick = [\tau]_{event}$
Ekrana yaz ("$[X]_j$ --(za)--> $[\lambda(e)]_1$ ")
For $k = 1$ to (*tick*-2)
Ekrana yaz ("$[\lambda(e)]_k$ --(za)--> $[\lambda(e)]_{k+1}$ ")
End
Ekrana yaz ("$[\lambda(e)]_{k+1}$ --(za)--> $[X]_i$ ")
$\Gamma_{new}(j,i) = 0$
End
End
End
End
$line = line\_new$
ATLA: Devam Et
End

EK-1D Sistem çıkmazının meydana gelmesinin engellendiği otomatanın belirlenen zaman aralığında benzetimini yapan algoritma

Load A^{τ}, Γ_{new}

$sure = input$
$sure1 = sure / d_u$
$time = 0$
$a = 1$
$b = 1$
$x = q_0$
$e = e_1$
$k = input1$
$x = [X]_k$
$[R]_a = x$
$a = a + 1$
ATLA: Devam Et
Do Loop Döngü-1
$x = [X]_k$
$e = [\Sigma]_{input2}$
If $x \notin Q(e)$ Then
  $e = [\Sigma]_{input2}$ (kullanıcıdan geçerli bir olay iste)
  If $x \notin Q(e)$ Then
    Exit Loop Döngü-1
  Else
    If $\tau_e = 0$ Then
      $[T]_b = e$
      $b = b + 1$
      For $i = 1$ to $|X|$
        If $\Gamma_{new}(k, i) = input2$ Then
          E-yaz ("sistem $[X]_k$ durumundan sonra e olayı meydana gelince $[X]_i$ durumuna gecikme olmadan geçer")
          E-yaz ("$[X]_k$ --(e)--> $[X]_i$ ")
          $k = i$
          $[R]_a = [X]_i$
          $a = a + 1$
        Else
          Devam Et
        End
      End
    ElseIf $sure1 > 0$ & $\tau_e = 1$ Then

If $t = sure1$ Then
E-yaz (“Süre bitti”)
Exit Loop Döngü-1
Else
$[T]_b = e$
$b = b + 1$
If $\Gamma_{new}(k,i) = input2$ Then
E-yaz (“sistem $[X]_k$ durumundan sonra 1 zaman adımıyla $[X]_i$ durumuna geçer”)
E-yaz (“$[X]_k$ --(za)--> $[X]_i$ ”)
$k = i$
$[R]_a = [X]_i$
$a = a + 1$
$[T]_b = \text{za}$
$b = b + 1$
$t = t + 1$
End
ElseIf $\tau_e > 1$ Then
If $t = sure1$ Then
E-yaz (“Süre bitti”)
Exit Loop Döngü-1
Else
$[T]_b = e$
$b = b + 1$
For $i = 1$ to $|X|$
If $\Gamma_{new}(k,i) = input2$ Then
E-yaz (“sistem $[X]_k$ durumundan sonra 1 zaman adımıyla $[\lambda(e)]_l$ durumuna geçer”)
E-yaz (“$[X]_k$ --(za)--> $[\lambda(e)]_l$ ”)
$[R]_a = [\lambda(e)]_l$
$a = a + 1$
$[T]_b = \text{za}$
$b = b + 1$
For $k = 1$ to (τ_e -2)
If $t = sure1$ Then
E-yaz (“Süre bitti”)
Exit Loop Döngü-1
Else
E-yaz (“sistem $[\lambda(e)]_k$ durumundan sonra k+1 zaman adımıyla $[\lambda(e)]_{k+1}$ durumuna geçer”)
E-yaz (“$[\lambda(e)]_k$ --(za)--> $[\lambda(e)]_{k+1}$ ”)

$[R]_a = [\lambda(e)]_{k+1}$
$a = a + 1$
$[T]_b = \text{za}$
$b = b + 1$
End
End
E-yaz ("sistem $[\lambda(e)]_{k+1}$ durumundan
sonra τ_e zaman adımıyla $[X]_i$ durumuna geçer")
E-yaz ("$[\lambda(e)]_{k+1}$ --(za)--> $[X]_i$ ")
$k = i$
End
End
Go To ATLA
End
End
End
End Loop Döngü-1

EK-1E İstenilen durumların bulunduğu en kısa durum dizisini ve en kısa sürede oluşan durum dizisini bulan algoritma

Load A^{τ}, Γ
durum sayısı = input
For $i = 1$ to *durum sayısı*
    $[desired\_states]_i = input1$
End
For $i = 1$ to $|X|$
    $a = 0$
    For $j = 1$ to $|X|$
        If $\Gamma(i, j) \neq 0$ Then
            $a = a + 1$
        End
    End
    $[event\_number]_i = a$
End
state=$[\ ]_{|X| x \max[event\_number]}$
For $i = 1$ to $|X|$
    $k = 0$
    For $j = 1$ to $|X|$
        If $\Gamma(i, j) \neq 0$ Then
            $state(i,k) = j$
            $k = k + 1$
        End
    End
End
times=$[\ ]_{|X| x \max[event\_number]}$
For $i = 1$ to $|X|$
    $k = 0$
    For $j = 1$ to $|X|$
        If $\Gamma(i, j) \neq 0$ Then
            $times(i,k) = [\tau]_{\Gamma(i,j)}$
            $k = k + 1$
        End
    End
End
$n = 1$
$time\_vek = [\ ]$
$vek = \{\ \}$

$[time\_vek]_1 = 0$
$vek\{1,n\} = [1]$
$n = n + 1$
For $i = 1$ to $[event\_number]_1$
$m = state(1,i)$
$k = times(1,i)$
$vek1 = [1]$
$vek2 = [\ ]_{1\times 2}$
$[vek2]_1 = [vek1]_1$
$[vek2]_2$ =m
$vek\{1,n\} = [vek2]$
$[time\_vek]_n = [\tau]_k$
$n = n + 1$
End
$o = |vek| + 1$
$t = 2$
For $i = 1$ to (1000 × durum sayısı)
$vek1 = vek\{1,t\}$
$last\_state = [vek1]_{|vek1|}$
If $[event\_number]_{last\_state} = 0$ Then
$t = t + 1$
Devam Et
Else
For $k = 1$ to $[event\_number]_{last\_state}$
$vek2 = [\ ]_{1\times(|vek1|+1)}$
For $j = 1$ to $|vek1|$
$[vek2]_j = [vek1]_j$
End
$m = state(last\_state,k)$
$k = times(last\_state,k)$
$[vek2]_{|vek2|} = m$
$vek\{1,o\} = vek2$
$[time\_vek]_o = [time\_vek]_t + [\tau]_k$
$o = o + 1$
End
End
$t = t + 1$
End
For $k = 1$ to t
$vek\{1,t\} = vek1$

For $j = 1$ to $|desired\_state|$

$[desird\_state]_j \notin vek1$ Then

$vek\{1,t\} = [\ ]$

End

End

n=1

For $k = 1$ to t

If $vek\{1,t\} \neq [\ ]$ Then

$desired\_vek\{1,n\} = vek\{1,t\}$

$[boyut]_n = |vek\{1,t\}|$

$[desired\_time\_vek]_n = [time\_vek]_t$

$n = n + 1$

End

End

If $[boyut]_n$ =min($[boyut]$) Then

I1= $desired\_vek\{1,n\}$

End

If $[desired\_time\_vek]_k$ =min($[desired\_time\_vek]$)

I2= $desired\_vek\{1,k\}$

End

Ekrana yaz (“q0 durumundan sonra ulaşılması istenilen durumlar”)

For $i = 1$ to durum sayısı

Ekrana yaz (“$[X]_{[desired\_states]_i}$ ”)

End

If I1=I2 Then

Ekrana yaz(“istenilen durumların oluştuğu en kısa durum dizisi”)

For i =1 to |I1|

Ekrana yaz (“$[X]_{[I1]_i}$ ”)

End

Ekrana yaz (“istenilen en kısa durum dizisi oluşurken meydana gelen olaylar”)

For i =1 to $[boyut]_n$ -1

$addr1 = [I1]_i$

$addr2 = [I1]_{i+1}$

Ekrana yaz (“$[\Sigma]_{\Gamma(addr1,addr2)}$ ”)

$tick = [\tau]_{\Gamma(addr1,addr2)}$

For i =1 to tick

Ekrana yaz (“za”)

End

Ekrana yaz (“istenilen durum dizisi oluşurken geçen minimum $[desired\_time\_vek]_n \times d_u$ zaman saniyedir”)

Ekrana yaz (“istenilen durum dizisi oluşurken ateşlenen zaman adımı sayısı $[desired\_time\_vek]_n$ tanedir”)

Else

Ekrana yaz(“istenilen durumların oluştuğu en kısa durum dizisi”)

For i =1 to |I1|

Ekrana yaz (“$[X]_{[I1]_i}$ ”)

End

Ekrana yaz (“istenilen en kısa durum dizisi oluşurken meydana gelen olaylar”)

For i =1 to $[boyut]_n$ -1

$addr1 = [I1]_i$

$addr2 = [I1]_{i+1}$

Ekrana yaz (“$[\Sigma]_{\Gamma(addr1,addr2)}$ ”)

$tick = [\tau]_{\Gamma(addr1,addr2)}$

For i =1 to tick

Ekrana yaz (“za”)

End

Ekrana yaz (“istenilen durum dizisi oluşurken geçen minimum $[desired\_time\_vek]_n \times d_u$ zaman saniyedir”)

Ekrana yaz (“istenilen durum dizisi oluşurken ateşlenen zaman adımı sayısı $[desired\_time\_vek]_n$ tanedir”)

Ekrana yaz(“istenilen durumların en kısa sürede oluştuğu durum dizisi”)

For i =1 to |I2|

Ekrana yaz (“$[X]_{[I2]_i}$ ”)

End

Ekrana yaz (“istenilen en kısa durum dizisi oluşurken meydana gelen olaylar”)

For i =1 to $[boyut]_k$ -1

$addr1 = [I2]_i$

$addr2 = [I2]_{i+1}$

Ekrana yaz (“$[\Sigma]_{\Gamma(addr1,addr2)}$ ”)

$tick = [\tau]_{\Gamma(addr1,addr2)}$

For i =1 to tick

Ekrana yaz (“za”)

End

End

Ekrana yaz (“istenilen durum dizisi oluşurken geçen minimum $[desired\_time\_vek]_k \times d_u$ zaman saniyedir”)

Ekrana yaz (“istenilen durum dizisi oluşurken ateşlenen zaman adımı sayısı $[desired\_time\_vek]_k$ tanedir”)

End

EK-2A Yazılımları çalıştıran arayüz (RUN_export.m)

```
function varargout = RUN_export(varargin)
% RUN_export M-file for RUN_export.fig
%      RUN_export, by itself, creates a new RUN_export or raises the existing
%      singleton*.
%
%      H = RUN_export returns the handle to a new RUN_export or the handle to
%      the existing singleton*.
%
%      RUN_export('Property','Value',...) creates a new RUN_export using the
%      given property value pairs. Unrecognized properties are passed via
%      varargin to RUN_export_OpeningFcn.  This calling syntax produces a
%      warning when there is an existing singleton*.
%
%      RUN_export('CALLBACK') and RUN_export('CALLBACK',hObject,...)
call the
%      local function named CALLBACK in RUN_export.M with the given input
%      arguments.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run_export (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help RUN_export

% Last Modified by GUIDE v2.5 05-Jun-2006 15:29:03

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @RUN_export_OpeningFcn, ...
                   'gui_OutputFcn',  @RUN_export_OutputFcn, ...
                   'gui_LayoutFcn',  @RUN_export_LayoutFcn, ...
                   'gui_Callback',   []);
if nargin & isstr(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
% --- Executes just before RUN_export is made visible.
function RUN_export_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   unrecognized PropertyName/PropertyValue pairs from the
%            command line (see VARARGIN)

% Choose default command line output for RUN_export
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes RUN_export wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = RUN_export_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

[filename, pathname] = uigetfile( ...
    {'*.m', 'All M-Files (*.m)'; ...
                              }, ...
    'Zamanlandırılmış Otomata Algoritması');

if isequal([filename,pathname],[0,0]) % Eger Dosya seçilmemisse islem yapma
    return

else
    File = fullfile(pathname,filename);
    % Eger .m dosyasi geçerli degilse, adi kaydetme
    %run(filename
    run(File)
```

```
    sistem
    A=findobj('Enable','off');
    set(A(4),'Enable','on')  %RUN butonunu kullanılır hale getirir
    set(A(3),'Enable','on')  %DEADLOCK butonunu kullanılır hale getirir
    set(A(2),'Enable','on')  %DESIRED CONTROLLER butonunu kullanılır hale
getirir
end

% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

algoritma          %algoritma.m yazılımını çalıştırır
winopen('bilgi.m')  %bilgi.m dosyasını kullanıcının görebilmesi için açar
cikis              %cikis.m yazılımını çalıştırır
winopen('bilgi1.m') %bilgi.m dosyasını kullanıcının görebilmesi için açar

% --- Executes on button press in pushbutton8.
function pushbutton8_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

desired                    %desired.m yazılımını çalıştırır
winopen('desired_bilgi.m')  %desired_bilgi.m dosyasını kullanıcının görebilmesi
için açar

% --- Executes on button press in pushbutton9.
function pushbutton9_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

deadlock                   %deadlock.m yazılımını çalıştırır
winopen('deadlock_bilgi.m') %deadlock_bilgi.m dosyasını kullanıcının
görebilmesi için açar
A=findobj('Enable','off');
set(A(1),'Enable','on')     %DEADLOCK CONTROLLER butonunu kullanılır hale
getirir.

% --- Executes on button press in pushbutton10.
function pushbutton10_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
controlled_algoritma          %controlled_algoritma.m yazılımını çalıştırır
winopen('controlled_bilgi.m')  %controlled_bilgi.m dosyasını kullanıcının
görebilmesi için açar
controlled_cikis              %controlled_cikis.m yazılımını çalıştırır
winopen('controlled_bilgi1.m') %controlled_bilgi1.m dosyasını kullanıcının
görebilmesi için açar

% --- Creates and returns a handle to the GUI figure.
function h1 = RUN_export_LayoutFcn(policy)

% policy - create a new figure or use a singleton. 'new' or 'reuse'.
persistent hsingleton;

if strcmpi(policy, 'reuse') & ishandle(hsingleton)
   h1 = hsingleton;
   return;
end

h1 = figure(...
'Units','characters',...
'PaperUnits',get(0,'defaultfigurePaperUnits'),...
'Color',[0.925490196078431 0.913725490196078 0.847058823529412],...

'Colormap',[0 0 0.5625;0 0 0.625;0 0 0.6875;0 0 0.75;0 0 0.8125;0 0 0.875;0 0
0.9375;0 0 1;0 0.0625 1;0 0.125 1;0 0.1875 1;0 0.25 1;0 0.3125 1;0 0.375 1;0
0.4375 1;0 0.5 1;0 0.5625 1;0 0.625 1;0 0.6875 1;0 0.75 1;0 0.8125 1;0 0.875 1;0
0.9375 1;0 1 1;0.0625 1 1;0.125 1 0.9375;0.1875 1 0.875;0.25 1 0.8125;0.3125 1
0.75;0.375 1 0.6875;0.4375 1 0.625;0.5 1 0.5625;0.5625 1 0.5;0.625 1
0.4375;0.6875 1 0.375;0.75 1 0.3125;0.8125 1 0.25;0.875 1 0.1875;0.9375 1
0.125;1 1 0.0625;1 1 0;1 0.9375 0;1 0.875 0;1 0.8125 0;1 0.75 0;1 0.6875 0;1
0.625 0;1 0.5625 0;1 0.5 0;1 0.4375 0;1 0.375 0;1 0.3125 0;1 0.25 0;1 0.1875 0;1
0.125 0;1 0.0625 0;1 0 0;0.9375 0 0;0.875 0 0;0.8125 0 0;0.75 0 0;0.6875 0
0;0.625 0 0;0.5625 0 0],...

'IntegerHandle','off',...
'InvertHardcopy',get(0,'defaultfigureInvertHardcopy'),...
'MenuBar','none',...
'Name','RUN',...
'NumberTitle','off',...
'PaperPositionMode','auto',...
'PaperSize',[20.98404194812 29.67743169791],...
'PaperType',get(0,'defaultfigurePaperType'),...
'Position',[103.8 35.8461538461539 97.6 16.6153846153846],...
'Renderer',get(0,'defaultfigureRenderer'),...
'RendererMode','manual',...
'HandleVisibility','callback',...
'Tag','figure1',...
```

```
'UserData',zeros(1,0));

setappdata(h1, 'GUIDEOptions', struct(...
'active_h', 1.130006e+002, ...
'taginfo', struct(...
'figure', 2, ...
'pushbutton', 13), ...
'override', 0, ...
'release', 13, ...
'resize', 'none', ...
'accessibility', 'callback', ...
'mfile', 1, ...
'callbacks', 1, ...
'singleton', 1, ...
'syscolorfig', 1, ...
'lastSavedFile', 'C:\RUN.m'));


h2 = uicontrol(...
'Parent',h1,...
'Units','characters',...
'Callback','RUN_export("pushbutton6_Callback",gcbo,[],guidata(gcbo))',...
'FontWeight','bold',...
'ListboxTop',0,...
'Position',[2 11.6153846153846 92.8 3.92307692307692],...
'String','YÜKLE',...
'Tag','pushbutton6');


h3 = uicontrol(...
'Parent',h1,...
'Units','characters',...
'Callback','RUN_export("pushbutton7_Callback",gcbo,[],guidata(gcbo))',...
'CData',zeros(1,0),...
'Enable','off',...
'FontWeight','bold',...
'ListboxTop',0,...
'Position',[2 6.61538461538462 45.2 3.92307692307692],...
'String','BENZETİM-1',...
'Tag','pushbutton7',...
'UserData',zeros(1,0));


h4 = uicontrol(...
'Parent',h1,...
'Units','characters',...
'Callback','RUN_export("pushbutton8_Callback",gcbo,[],guidata(gcbo))',...
'CData',zeros(1,0),...
```

```
'Enable','off',...
'FontWeight','bold',...
'ListboxTop',0,...
'Position',[49.8 6.61538461538462 45.2 3.92307692307692],...
'String','İSTENİLEN DURUM KONTROLÖRÜ',...
'Tag','pushbutton8',...
'UserData',zeros(1,0));


h5 = uicontrol(...
'Parent',h1,...
'Units','characters',...
'Callback','RUN_export("pushbutton9_Callback",gcbo,[],guidata(gcbo))',...
'CData',zeros(1,0),...
'Enable','off',...
'FontWeight','bold',...
'ListboxTop',0,...
'Position',[2.2 1.07692307692308 45.2 3.92307692307692],...
'String','SİSTEM ÇIKMAZI & KONTROLÖR',...
'Tag','pushbutton9',...
'UserData',zeros(1,0));


h6 = uicontrol(...
'Parent',h1,...
'Units','characters',...
'Callback','RUN_export("pushbutton10_Callback",gcbo,[],guidata(gcbo))',...
'CData',zeros(1,0),...
'Enable','off',...
'FontWeight','bold',...
'ListboxTop',0,...
'Position',[49.8 1.07692307692308 45.2 3.92307692307692],...
'String','BENZETİM -2',...
'Tag','pushbutton10',...
'UserData',zeros(1,0));


hsingleton = h1;
% --- Handles default GUIDE GUI creation and callback dispatch
function varargout = gui_mainfcn(gui_State, varargin)

%   GUI_MAINFCN provides these command line APIs for dealing with GUIs
%
%      RUN_EXPORT, by itself, creates a new RUN_EXPORT or raises the
existing
%      singleton*.
%
```

```
%      H = RUN_EXPORT returns the handle to a new RUN_EXPORT or the
handle to
%      the existing singleton*.
%
%      RUN_EXPORT('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in RUN_EXPORT.M with the given input
arguments.
%
%      RUN_EXPORT('Property','Value',...) creates a new RUN_EXPORT or
raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before untitled_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to untitled_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".

%   Copyright 1984-2002 The MathWorks, Inc.
%   $Revision: 1.4 $ $Date: 2002/05/31 21:44:31 $

gui_StateFields =  {'gui_Name'
                    'gui_Singleton'
                    'gui_OpeningFcn'
                    'gui_OutputFcn'
                    'gui_LayoutFcn'
                    'gui_Callback'};

gui_Mfile = '';
for i=1:length(gui_StateFields)
   if ~isfield(gui_State, gui_StateFields{i})
      error('Could not find field %s in the gui_State struct in GUI M-file %s',
gui_StateFields{i}, gui_Mfile);
   elseif isequal(gui_StateFields{i}, 'gui_Name')
      gui_Mfile = [getfield(gui_State, gui_StateFields{i}), '.m'];
   end
end

numargin = length(varargin);

if numargin == 0
   % RUN_EXPORT
   % create the GUI
   gui_Create = 1;
elseif numargin > 3 & ischar(varargin{1}) & ishandle(varargin{2})
   % RUN_EXPORT('CALLBACK',hObject,eventData,handles,...)
```

```
    gui_Create = 0;
else
    % RUN_EXPORT(...)
    % create the GUI and hand varargin to the openingfcn
    gui_Create = 1;
end

if gui_Create == 0
    varargin{1} = gui_State.gui_Callback;
    if nargout
        [varargout{1:nargout}] = feval(varargin{:});
    else
        feval(varargin{:});
    end
else
    if gui_State.gui_Singleton
        gui_SingletonOpt = 'reuse';
    else
        gui_SingletonOpt = 'new';
    end

    % Open fig file with stored settings.  Note: This executes all component
    % specific CreateFunctions with an empty HANDLES structure.

   % Do feval on layout code in m-file if it exists
    if ~isempty(gui_State.gui_LayoutFcn)
        gui_hFigure = feval(gui_State.gui_LayoutFcn, gui_SingletonOpt);
    else
        gui_hFigure = local_openfig(gui_State.gui_Name, gui_SingletonOpt);
        % If the figure has InGUIInitialization it was not completely created
        % on the last pass.  Delete this handle and try again.
        if isappdata(gui_hFigure, 'InGUIInitialization')
            delete(gui_hFigure);
            gui_hFigure = local_openfig(gui_State.gui_Name, gui_SingletonOpt);
        end
    end

    % Set flag to indicate starting GUI initialization
    setappdata(gui_hFigure,'InGUIInitialization',1);
    % Fetch GUIDE Application options
    gui_Options = getappdata(gui_hFigure,'GUIDEOptions');
    if ~isappdata(gui_hFigure,'GUIOnScreen')
        % Adjust background color
        if gui_Options.syscolorfig
            set(gui_hFigure,'Color', get(0,'DefaultUicontrolBackgroundColor'));

        end
        % Generate HANDLES structure and store with GUIDATA
```

```
    guidata(gui_hFigure, guihandles(gui_hFigure));
end

% If user specified 'Visible','off' in p/v pairs, don't make the figure
% visible.
gui_MakeVisible = 1;
for ind=1:2:length(varargin)
    if length(varargin) == ind
        break;
    end

    len1 = min(length('visible'),length(varargin{ind}));
    len2 = min(length('off'),length(varargin{ind+1}));
    if ischar(varargin{ind}) & ischar(varargin{ind+1}) & ...
            strncmpi(varargin{ind},'visible',len1) & len2 > 1

        if strncmpi(varargin{ind+1},'off',len2)
            gui_MakeVisible = 0;

        elseif strncmpi(varargin{ind+1},'on',len2)
            gui_MakeVisible = 1;
        end
    end
end

% Check for figure param value pairs
for index=1:2:length(varargin)

    if length(varargin) == index
        break;
    end

    try, set(gui_hFigure, varargin{index}, varargin{index+1}), catch, break, end
end
% If handle visibility is set to 'callback', turn it on until finished
% with OpeningFcn
gui_HandleVisibility = get(gui_hFigure,'HandleVisibility');
if strcmp(gui_HandleVisibility, 'callback')
    set(gui_hFigure,'HandleVisibility', 'on');
end

feval(gui_State.gui_OpeningFcn, gui_hFigure, [], guidata(gui_hFigure),
varargin{:});
if ishandle(gui_hFigure)
    % Update handle visibility
     set(gui_hFigure,'HandleVisibility', gui_HandleVisibility);
    % Make figure visible
    if gui_MakeVisible
```

```
            set(gui_hFigure, 'Visible', 'on')
            if gui_Options.singleton
                setappdata(gui_hFigure,'GUIOnScreen', 1);
            end

        end
        % Done with GUI initialization
        rmappdata(gui_hFigure,'InGUIInitialization');
    end

    % If handle visibility is set to 'callback', turn it on until finished with
    % OutputFcn
    if ishandle(gui_hFigure)
        gui_HandleVisibility = get(gui_hFigure,'HandleVisibility');
        if strcmp(gui_HandleVisibility, 'callback')
            set(gui_hFigure,'HandleVisibility', 'on');
        end
        gui_Handles = guidata(gui_hFigure);
    else
        gui_Handles = [];
    end

    if nargout
        [varargout{1:nargout}] = feval(gui_State.gui_OutputFcn, gui_hFigure, [],
gui_Handles);
    else
        feval(gui_State.gui_OutputFcn, gui_hFigure, [], gui_Handles);
    end
    if ishandle(gui_hFigure)
        set(gui_hFigure,'HandleVisibility', gui_HandleVisibility);
    end
end

function gui_hFigure = local_openfig(name, singleton)
if nargin('openfig') == 3
    gui_hFigure = openfig(name, singleton, 'auto');
else
    % OPENFIG did not accept 3rd input argument until R13,
    % toggle default figure visible to prevent the figure
    % from showing up too soon.
    gui_OldDefaultVisible = get(0,'defaultFigureVisible');
    set(0,'defaultFigureVisible','off');
    gui_hFigure = openfig(name, singleton);
    set(0,'defaultFigureVisible',gui_OldDefaultVisible);
end
```

EK-2B Örnek Otomata için Girdi Dosyası (girdi.m)

```
%Zamanlandırılmış otomatayı tanımlayan veriler bu dosyada verilir
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all;

D=[2 3 1 4 2 3 0 5 1 3 2 2 1 4 2 2 0 3 0 2 1 3 2 5 4 1 1 5 2 3 5 3];   %olayların
sırasıyla saniye cinsinden
                                                      %gecikmelerini
gösteren vektör
                                                      %örnek olarak
birinci kolondaki 2, e1 olayının
                                                      %saniye cinsinden
gecikmesine eşittir

Gama=[0 1 0 5 0 8 0 0 0 0 0 0 0 0 0 0 0;          %otomata matrisi
    0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0;            %durumlar arasındaki
geçerli olaylar burada tanımlanır
    0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 27 0;
    0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0;
    6 0 7 0 0 9 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 12 0 0 0 0 0 0 0 0 0;
    0 0 0 0 10 0 0 0 14 17 0 0 0 0 0 0 0;
    0 0 0 0 11 0 13 0 15 0 20 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 18 16 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 21 19 0 0 22 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 23 0 0 25 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 24 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0 26 0 0 0 31;
    0 0 28 0 0 0 0 0 0 0 0 0 0 30 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 29 0 32;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];


Sigma=['e1  ';'e2  ';'e3  ';'e4  ';'e5  ';'e6  ';'e7  ';'e8  ';'e9
';'e10';'e11';'e12';'e13';'e14';'e15';'e16';  %olaylara verilen isimler

'e17';'e18';'e19';'e20';'e21';'e22';'e23';'e24';'e25';'e26';'e27';'e28';'e29';'e30';'e31';'e32'
];

Q=['q0  ';'q1  ';'q2  ';'q3  ';'q4  ';'q5  ';'q6  ';'q7  ';'q8  ';'q9
';'q10';'q11';'q12';'q13';'q14';'q15';'q16']; %durumlara verilen isimler

save all;
```

EK-2C Otomataya zaman adımlarının ilave edilmesi ile oluşturulan zamanlandırılmış otomata hakkında bilgi veren yazılım ve çalışması sonucu oluşan çıktı dosyası

sistem.m

```
load all;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%
Sigma1 = cellstr(Sigma);
k=size(Sigma1);
size_sigma=k(1,1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%
Q1=cellstr(Q);
l=size(Q1);
size_Q=l(1,1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%

k1=0;                    %tüm zaman gecikmelerinin tamsayı olmasını sağlama
for k=1:100000           %gcd() fonksiyonu sadece tamsayılar ile ebob bulduğu
için bu işlem gerekir
  Dx=D*(10^k1);
  E=fix(Dx)-Dx;
  if all(E==0)
    break
  else
    k1=k1+1;
  end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
s=size(Dx);
b=s(:,2);
a=Dx(:,1);
ebob=a;
for i=2:b    % zaman gecikmelerinin ebob'unu bulma
   ebob=gcd(Dx(:,i),ebob);
end
du=ebob/(10^k1);              %bulunan ebobun birim zaman gecikmesi olarak
atanması
Tau=D/du;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
U=zeros(size_Q,size_Q);              %bir durumdan bir duruma geçerken oluşan
zaman adımı sayısı
for i=1:size_Q
   for j=1:size_Q
      if Gama(i,j)~=0
         c=Gama(i,j);
         U(i,j)=(Tau(1,c));
      end
   end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fid=fopen('sistem_bilgi.m','w');
fprintf(fid,'birim zaman gecikmesi %g saniyedir.\n\n',du);
for i=1:size_Q                 %her durumdan sonra meydana gelebilecek olay
sayısının bulunması
   [x,y,z]=find(Gama(i,:));
   initial_state_name=Q(i,:); % sistemin o anda bulunduğu durumun isminin
atanması
   if length(z)==0
      fprintf(fid,'%s durumunda meydana gelebilecek geçerli bir olay
yoktur..!\n\n',initial_state_name);
   else
      for j=1:length(z)
         tick=fix(U(i,y(j)));
%          tick=str2num(num2str(tick));
         next_state_name=Q(y(j),:);  %sistemin gececeği bir sonraki durumun
isminin atanması
         initial_event_name=Sigma(z(j),:); % şimdiki durumdan sonra geçerli
olayın isminin atanması
         fprintf(fid,'sistem %s durumunda iken %s olayı meydana gelince oluşan
geçişler\n',initial_state_name,initial_event_name);
         fprintf(fid,'%s olayı icin gecerli olan zaman adımı sayısı %d
tanedir.\n',initial_event_name,tick);
         % zaman gecikmesi 0 sn olan olayların meydana gelmesi
         if tick==0
           fprintf(fid,'%s --(%s)-->
%s\n\n',initial_state_name,initial_event_name,next_state_name);

         % 1 tane zaman adımına sahip olayların meydana gelmesi
         elseif tick==1
              fprintf(fid,'%s --(za)--> %s\n\n',initial_state_name,next_state_name);

         %zaman adımı 1 taneden fazla olan olayların meydana gelmesi
```

```
            elseif tick>0
                  initial_state_name1=initial_state_name;
                  m=tick-1;
                  for n=1:m

next_state_name1=[initial_state_name,initial_event_name,'(',num2str(n),')'];
                        fprintf(fid,'%s --(za)-->
%s\n',initial_state_name1,next_state_name1);
                        initial_state_name1=next_state_name1;

               end
               fprintf(fid,'%s --(za)--> %s\n\n',initial_state_name1,next_state_name);
        end
      end
    end
end
save all;
fclose(fid);
winopen('sistem_bilgi.m');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

sistem_bilgi.m

birim zaman gecikmesi 1 saniyedir.

sistem q0 durumunda iken e1 olayı meydana gelince oluşan geçişler
e1 olayı icin gecerli olan zaman adımı sayısı 2 tanedir.
q0 --(za)--> q0 e1 (1)
q0 e1 (1) --(za)--> q1

sistem q0 durumunda iken e5 olayı meydana gelince oluşan geçişler
e5 olayı icin gecerli olan zaman adımı sayısı 2 tanedir.
q0 --(za)--> q0 e5 (1)
q0 e5 (1) --(za)--> q3

sistem q0 durumunda iken e8 olayı meydana gelince oluşan geçişler
e8 olayı icin gecerli olan zaman adımı sayısı 5 tanedir.
q0 --(za)--> q0 e8 (1)
q0 e8 (1) --(za)--> q0 e8 (2)
q0 e8 (2) --(za)--> q0 e8 (3)
q0 e8 (3) --(za)--> q0 e8 (4)
q0 e8 (4) --(za)--> q5

sistem q1 durumunda iken e2 olayı meydana gelince oluşan geçişler
e2 olayı icin gecerli olan zaman adımı sayısı 3 tanedir.
q1 --(za)--> q1 e2 (1)
q1 e2 (1) --(za)--> q1 e2 (2)

q1 e2 (2) --(za)--> q2

sistem q2 durumunda iken e3 olayı meydana gelince oluşan geçişler
e3 olayı icin gecerli olan zaman adımı sayısı 1 tanedir.
q2 --(za)--> q3

sistem q2 durumunda iken e27 olayı meydana gelince oluşan geçişler
e27 olayı icin gecerli olan zaman adımı sayısı 1 tanedir.
q2 --(za)--> q15

sistem q3 durumunda iken e4 olayı meydana gelince oluşan geçişler
e4 olayı icin gecerli olan zaman adımı sayısı 4 tanedir.
q3 --(za)--> q3 e4 (1)
q3 e4 (1) --(za)--> q3 e4 (2)
q3 e4 (2) --(za)--> q3 e4 (3)
q3 e4 (3) --(za)--> q5

sistem q4 durumunda iken e6 olayı meydana gelince oluşan geçişler
e6 olayı icin gecerli olan zaman adımı sayısı 3 tanedir.
q4 --(za)--> q4 e6 (1)
q4 e6 (1) --(za)--> q4 e6 (2)
q4 e6 (2) --(za)--> q0

sistem q4 durumunda iken e7 olayı meydana gelince oluşan geçişler
e7 olayı icin gecerli olan zaman adımı sayısı 0 tanedir.
q4 --(e7)--> q2

sistem q4 durumunda iken e9 olayı meydana gelince oluşan geçişler
e9 olayı icin gecerli olan zaman adımı sayısı 1 tanedir.
q4 --(za)--> q5

sistem q5 durumunda iken e12 olayı meydana gelince oluşan geçişler
e12 olayı icin gecerli olan zaman adımı sayısı 2 tanedir.
q5 --(za)--> q5 e12(1)
q5 e12(1) --(za)--> q7

sistem q6 durumunda iken e10 olayı meydana gelince oluşan geçişler
e10 olayı icin gecerli olan zaman adımı sayısı 3 tanedir.
q6 --(za)--> q6 e10(1)
q6 e10(1) --(za)--> q6 e10(2)
q6 e10(2) --(za)--> q4

sistem q6 durumunda iken e14 olayı meydana gelince oluşan geçişler
e14 olayı icin gecerli olan zaman adımı sayısı 4 tanedir.
q6 --(za)--> q6 e14(1)
q6 e14(1) --(za)--> q6 e14(2)
q6 e14(2) --(za)--> q6 e14(3)
q6 e14(3) --(za)--> q8

sistem q6 durumunda iken e17 olayı meydana gelince oluşan geçişler
e17 olayı icin gecerli olan zaman adımı sayısı 0 tanedir.
q6 --(e17)--> q9

sistem q7 durumunda iken e11 olayı meydana gelince oluşan geçişler
e11 olayı icin gecerli olan zaman adımı sayısı 2 tanedir.
q7 --(za)--> q7 e11(1)
q7 e11(1) --(za)--> q4

sistem q7 durumunda iken e13 olayı meydana gelince oluşan geçişler
e13 olayı icin gecerli olan zaman adımı sayısı 1 tanedir.
q7 --(za)--> q6

sistem q7 durumunda iken e15 olayı meydana gelince oluşan geçişler
e15 olayı icin gecerli olan zaman adımı sayısı 2 tanedir.
q7 --(za)--> q7 e15(1)
q7 e15(1) --(za)--> q8

sistem q7 durumunda iken e20 olayı meydana gelince oluşan geçişler
e20 olayı icin gecerli olan zaman adımı sayısı 2 tanedir.
q7 --(za)--> q7 e20(1)
q7 e20(1) --(za)--> q10

q8 durumunda meydana gelebilecek geçerli bir olay yoktur..!

sistem q9 durumunda iken e18 olayı meydana gelince oluşan geçişler
e18 olayı icin gecerli olan zaman adımı sayısı 3 tanedir.
q9 --(za)--> q9 e18(1)
q9 e18(1) --(za)--> q9 e18(2)
q9 e18(2) --(za)--> q7

sistem q9 durumunda iken e16 olayı meydana gelince oluşan geçişler
e16 olayı icin gecerli olan zaman adımı sayısı 2 tanedir.
q9 --(za)--> q9 e16(1)
q9 e16(1) --(za)--> q8

sistem q10 durumunda iken e21 olayı meydana gelince oluşan geçişler
e21 olayı icin gecerli olan zaman adımı sayısı 1 tanedir.
q10 --(za)--> q7

sistem q10 durumunda iken e19 olayı meydana gelince oluşan geçişler
e19 olayı icin gecerli olan zaman adımı sayısı 0 tanedir.
q10 --(e19)--> q8

sistem q10 durumunda iken e22 olayı meydana gelince oluşan geçişler
e22 olayı icin gecerli olan zaman adımı sayısı 3 tanedir.
q10 --(za)--> q10e22(1)

q10e22(1) --(za)--> q10e22(2)
q10e22(2) --(za)--> q11

sistem q11 durumunda iken e23 olayı meydana gelince oluşan geçişler
e23 olayı icin gecerli olan zaman adımı sayısı 2 tanedir.
q11 --(za)--> q11e23(1)
q11e23(1) --(za)--> q10

sistem q11 durumunda iken e25 olayı meydana gelince oluşan geçişler
e25 olayı icin gecerli olan zaman adımı sayısı 4 tanedir.
q11 --(za)--> q11e25(1)
q11e25(1) --(za)--> q11e25(2)
q11e25(2) --(za)--> q11e25(3)
q11e25(3) --(za)--> q13

sistem q12 durumunda iken e24 olayı meydana gelince oluşan geçişler
e24 olayı icin gecerli olan zaman adımı sayısı 5 tanedir.
q12 --(za)--> q12e24(1)
q12e24(1) --(za)--> q12e24(2)
q12e24(2) --(za)--> q12e24(3)
q12e24(3) --(za)--> q12e24(4)
q12e24(4) --(za)--> q11

sistem q13 durumunda iken e26 olayı meydana gelince oluşan geçişler
e26 olayı icin gecerli olan zaman adımı sayısı 1 tanedir.
q13 --(za)--> q12

sistem q13 durumunda iken e31 olayı meydana gelince oluşan geçişler
e31 olayı icin gecerli olan zaman adımı sayısı 5 tanedir.
q13 --(za)--> q13e31(1)
q13e31(1) --(za)--> q13e31(2)
q13e31(2) --(za)--> q13e31(3)
q13e31(3) --(za)--> q13e31(4)
q13e31(4) --(za)--> q16

sistem q14 durumunda iken e28 olayı meydana gelince oluşan geçişler
e28 olayı icin gecerli olan zaman adımı sayısı 5 tanedir.
q14 --(za)--> q14e28(1)
q14e28(1) --(za)--> q14e28(2)
q14e28(2) --(za)--> q14e28(3)
q14e28(3) --(za)--> q14e28(4)
q14e28(4) --(za)--> q2

sistem q14 durumunda iken e30 olayı meydana gelince oluşan geçişler
e30 olayı icin gecerli olan zaman adımı sayısı 3 tanedir.
q14 --(za)--> q14e30(1)
q14e30(1) --(za)--> q14e30(2)
q14e30(2) --(za)--> q13

sistem q15 durumunda iken e29 olayı meydana gelince oluşan geçişler
e29 olayı icin gecerli olan zaman adımı sayısı 2 tanedir.
q15 --(za)--> q15e29(1)
q15e29(1) --(za)--> q14

sistem q15 durumunda iken e32 olayı meydana gelince oluşan geçişler
e32 olayı icin gecerli olan zaman adımı sayısı 3 tanedir.
q15 --(za)--> q15e32(1)
q15e32(1) --(za)--> q15e32(2)
q15e32(2) --(za)--> q16

q16 durumunda meydana gelebilecek geçerli bir olay yoktur..!

EK-2D Belirlenen zaman aralığında otomatanın benzetimini yapan yazılımlar ve çalışması sonucu oluşan çıktı dosyaları

algoritma.m

```
load all;
% Bu fonksiyon ana program tarafından çağırılır.
% Bu yazılım ilk olarak sistemdeki geçerli olan tüm geçerli geçişleri bulur
% ve bunları zaman adımları ile birlikte sistem_bilgi.m dosyasına
% yazar. Daha sonra aşağıdaki işlemleri yapar.

% 1)
%Zaman gecikmelerinin ebobu bulunarak ve birim gecikme zamanı olarak
atanarak durumlar
%arasında meydana gelen zaman adımı sayısını gösteren U matrisi oluşturulur.

% 2)
% Kullanıcıdan girdileri isteyen ekranlar burada yaratılır

% 3)
% Oluşan durumları zaman adımına göre ekrana yazar. Ayrıca uyarıları
% kullanıcıya ekrana yazdırarak bildirir.

% 4)
% Ekrana yazdırılan tüm yazıları bilgi.m dosyasında saklar

% 5)
% Oluşan durumları ve meydana gelen olayları sırası ile ayrı karakter
% yığınlarında saklar.

load all;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%

fid=fopen('bilgi.m','w');     %ekrana yazılan yazıları dosyada saklamak için bilgi.m
dosyasını açar

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%

set(0,'defaultfigurePosition',[300,300,50,50]);
prompt= {
        'Zaman dilimini saniye cinsinden giriniz:(sn)',...
        };
     baslik='Zamanlandırılmış otomata algoritması';
   lineNo=1;
```

```
                                        %kullanıcıdan zaman diliminin istenmesi
def={
    '1'...

    };

answer=inputdlg(prompt,baslik,lineNo,def);

  if length(answer)==0,
    return;
  end


sure=str2num(answer{1,1});
sure2=fix(sure/du);       % kullanın girdiği zamanın tamsayı olması
garantilemek
sure1=sure2(1,1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
x=1;                %durum dizisinin yazılacağı R arrayini oluşturma
Rc=[];
Re=char(Rc);
R=cellstr(Re);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
y=1;                %olay dizisinin yazılacağı T arrayinin oluşturma
Tc=[];
Te=char(Tc);
T=cellstr(Te);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
t=0;   % meydana gelen zaman adımı sayısını tutmak için
state=0; % kullanıcı ekranındaki ilk açılıştaki olay ve durum için default değerleri
atamak
event=1;
for n=1:1000000
  state1=num2str(state);
  event1=num2str(event);
set(0,'defaultfigurePosition',[300,300,50,50]);
prompt= {
      'Sistemin bulundugu durumu giriniz: q',...
      'Meydana gelmesi istenilen olayi giriniz: e',...
      };
    baslik='Zamanlandırılmış otomata algoritması';
```

```
  lineNo=1;

                        % Kullanıcıdan olayların ve durumların girilmesinin istendiği
arabirimi oluşturma
def={
    state1...
    event1...
    };

answer=inputdlg(prompt,baslik,lineNo,def);

if length(answer)==0,
  break;
end


state=str2num(answer{1,1});
event=str2num(answer{2,1});
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%

C=Gama((state+1),:); %olayın verilen durumdan sonra geçerli olup olmadığının
belirlenmesi
e=find(C==event);   %olayı otomata matrisinde duruma ait satırda arar

if n==1                     %başlangıç durumu R arrayine yazma
  R(x,:)=Q1((state+1),:);
  x=x+1;
end

if all(C==0)          %eğer tüm satır 0 ise yani sistem çıkmazı oluşuyorsa
  sprintf('%s durumunda sistem cikmazi olusuyor',Q((state+1),:))
  fprintf(fid,'%s durumunda sistem cikmazi olusuyor\n',Q((state+1),:));
  fclose(fid)
  return

elseif  isempty(e)    %eğer satırda olay yok ise yani geçerli durumdan sonra bu
olay tanımlı değil ise
   sprintf('%s durumundan sonra %s olayı meydana
gelemez..!',Q((state+1),:),Sigma(event,:))
   fprintf(fid,'%s durumundan sonra %s meydana
gelemez..!\n',Q((state+1),:),Sigma(event,:));

   set(0,'defaultfigurePosition',[300,300,50,50]);    %kullanıcıdan tekrar bir olay
istemek
prompt= {
```

```
        'Durumdan sonra meydana gelecek dogru olayi giriniz: e',...
        };
      baslik=Q((state+1),:);
   lineNo=1;


def={
      ' ',...
      };

answer=inputdlg(prompt,baslik,lineNo,def);

if length(answer)==0,
   break;
end

   event=str2num(answer{1,1});
   e=find(C==event);

    if  isempty(e)    %eğer istenilen tanımlı değil ise
       sprintf('%s durumundan sonra %s olayı meydana
gelemez..!',Q((state+1),:),Sigma(event,:))
       fprintf(fid,'%s durumundan sonra %s olayı meydana
gelemez..!\n',Q((state+1),:),Sigma(event,:));
       fclose(fid)
       return
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


e=find(C==event); % kullanıcının verdiği tanımlı olayın otomata matrisinde
yerinin bulunması
tick=fix(U((state+1),e)); % bu olaya karşılık gelen zaman adımı sayısının
bulunması
initial_state_name=Q((state+1),:); % sistemin o anda bulunduğu durumun isminin
atanması
next_state_name=Q(e,:);  %sistemin gececeği bir sonraki durumun isminin
atanması
initial_event_name=Sigma(event,:); % şimdiki durumdan sonra geçerli olayın
isminin atanması

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%başta kullanıcının verdiği zaman aralığının birim zaman gecikmesinden
%küçük olduğu durumda sadece zaman gecikmesi 0 sn olan olayların
%meydana gelmesine izin verilmesi

if sure1==0 && tick==0
   sprintf('sistem %s durumundan sonra %s olayı meydana gelince %s durumuna
gecikme olmadan gecer',initial_state_name,initial_event_name,next_state_name)
   fprintf(fid,'sistem %s durumundan sonra %s olayı meydana gelince %s
durumuna gecikme olmadan
gecer\n',initial_state_name,initial_event_name,next_state_name);
   fprintf(fid,'%s --(%s)-->
%s\n\n',initial_state_name,initial_event_name,next_state_name);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% zaman gecikmesi 0 sn olan olayların meydana gelmesi
elseif tick==0 && sure1>0

   T(y,:)=Sigma1(event,:);
   y=y+1;

   sprintf('sistem %s durumundan sonra %s olayı meydana gelince %s durumuna
gecikme olmadan gecer',initial_state_name,initial_event_name,next_state_name)
   fprintf(fid,'sistem %s durumundan sonra %s meydana gelince %s durumuna
gecikme olmadan
gecer\n',initial_state_name,initial_event_name,next_state_name);
   fprintf(fid,'%s --(%s)-->
%s\n\n',initial_state_name,initial_event_name,next_state_name);

   R(x,:)=cellstr(next_state_name);
   x=x+1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%

% 1 tane zaman adımına sahip olayların meydana gelmesi
elseif tick==1 && sure1>0

   T(y,:)=Sigma1(event,:);
   y=y+1;

   if t==sure1                          % geçen zamanın kontrol edilmesi
      sprintf('%d saniye bitti..!',sure)
      fprintf(fid,'%d saniye bitti..!\n',sure);
      fclose(fid)
      return
   end
```

```
   sprintf('sistem %s durumundan sonra 1 zaman adımıyla %s durumuna
geçer',initial_state_name,next_state_name)
   sprintf('Ara durum olusmaz...!')
   fprintf(fid,'sistem %s durumundan sonra 1 zaman adımıyla %s durumuna geçer.
Ara durum olusmaz...!\n',initial_state_name,next_state_name);
   fprintf(fid,'%s --(za)--> %s\n\n',initial_state_name,next_state_name);

   R(x,:)=cellstr(next_state_name);
   x=x+1;

   T(y,:)=cellstr('za');
   y=y+1;


   t=t+1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 %zaman adımı 1 taneden fazla olan olayların meydana gelmesi

elseif sure1>0

   if t<sure1
     T(y,:)=Sigma1(event,:);
     y=y+1;
   end

   initial_state_name1=initial_state_name;
   for i=1:(tick-1)

      if t==sure1              % geçen zamanın kontrol edilmesi
         sprintf('%d saniye bitti..!',sure)
         fprintf(fid,'%d saniye bitti..!\n',sure);
         fclose(fid)
         return
      end

      next_state_name1=[initial_state_name,initial_event_name,'(',num2str(i),')'];
      sprintf('sistem %s durumundan sonra %d. zaman adımıyla %s durumuna
geçer',initial_state_name1,i,next_state_name1)
      fprintf(fid,'sistem %s durumundan sonra %d. zaman adımıyla %s durumuna
geçer\n',initial_state_name1,i,next_state_name1);
      fprintf(fid,'%s --(za)--> %s\n\n',initial_state_name1,next_state_name1);
      initial_state_name1=next_state_name1;

      T(y,:)=cellstr('za');
```

```
      y=y+1;

      R(x,:)=cellstr(next_state_name1);
      x=x+1;

      t=t+1;
   end

   i=tick;

   if t==sure1              % geçen zamanın kontrol edilmesi
       sprintf('%d saniye bitti..!',sure)
       fprintf(fid,'%d saniye bitti..!\n',sure);
       fclose(fid)
       return
   end

   sprintf('sistem %s durumundan sonra %d. zaman adımıyla %s durumuna
geçer',initial_state_name1,i,next_state_name)
   fprintf(fid,'sistem %s durumundan sonra %d. zaman adımıyla %s durumuna
geçer\n',initial_state_name1,i,next_state_name);
   fprintf(fid,'%s --(za)--> %s\n\n',initial_state_name1,next_state_name);

   R(x,:)=cellstr(next_state_name);
   x=x+1;

   T(y,:)=cellstr('za');
   y=y+1;


   t=t+1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
else  % verilen zaman aralığı 0 sn iken zaman gecikmesi 0 sn'den farklı olan
olayların meydana gelmesini engellemek
   sprintf('%d saniye geçerli bir süre değil',sure)
   return;
end

state=e-1;
end
```

cikis.m

```
fid=fopen('bilgi1.m','w'); % oluşan durumları ve meydana gelen olayları yazmak
için bilgi1.m dosyasını açar
```

```
m=size(R);
size_R=m(1,1);
n=size(T);
size_T=n(1,1);

fprintf(fid,'verilen sürede oluşan durumlar\n');

for i=1:size_R
   Rc=char(R(i,:));
   fprintf(fid,'%s\n',Rc);
end

fprintf(fid,'verilen sürede meydana gelen olaylar\n');

for i=1:size_T
   Tc=char(T(i,:));
   fprintf(fid,'%s\n',Tc);
end

fclose(fid);
```

bilgi.m

```
sistem q0  durumundan sonra 1. zaman adımıyla q0 e1 (1) durumuna geçer
q0  --(za)--> q0 e1 (1)

sistem q0 e1 (1) durumundan sonra 2. zaman adımıyla q1  durumuna geçer
q0 e1 (1) --(za)--> q1

sistem q1  durumundan sonra 1. zaman adımıyla q1 e2 (1) durumuna geçer
q1  --(za)--> q1 e2 (1)

sistem q1 e2 (1) durumundan sonra 2. zaman adımıyla q1 e2 (2) durumuna geçer
q1 e2 (1) --(za)--> q1 e2 (2)

sistem q1 e2 (2) durumundan sonra 3. zaman adımıyla q2  durumuna geçer
q1 e2 (2) --(za)--> q2

sistem q2  durumundan sonra 1 zaman adımıyla q3  durumuna geçer. Ara durum
olusmaz...!
q2  --(za)--> q3

sistem q3  durumundan sonra 1. zaman adımıyla q3 e4 (1) durumuna geçer
q3  --(za)--> q3 e4 (1)

sistem q3 e4 (1) durumundan sonra 2. zaman adımıyla q3 e4 (2) durumuna geçer
q3 e4 (1) --(za)--> q3 e4 (2)
```

sistem q3 e4 (2) durumundan sonra 3. zaman adımıyla q3 e4 (3) durumuna geçer
q3 e4 (2) --(za)--> q3 e4 (3)

sistem q3 e4 (3) durumundan sonra 4. zaman adımıyla q5 durumuna geçer
q3 e4 (3) --(za)--> q5

10 saniye bitti..!

bilgi1.m
verilen sürede oluşan durumlar
q0
q0 e1 (1)
q1
q1 e2 (1)
q1 e2 (2)
q2
q3
q3 e4 (1)
q3 e4 (2)
q3 e4 (3)
q5
verilen sürede meydana gelen olaylar
e1
za
za
e2
za
za
za
e3
za
e4
za
za
za
za
e12

bilgi.m (Süre bitmediği halde sistem çıkmaza girer ise)
sistem q0 durumundan sonra 1. zaman adımıyla q0 e1 (1) durumuna geçer
q0 --(za)--> q0 e1 (1)

sistem q0 e1 (1) durumundan sonra 2. zaman adımıyla q1 durumuna geçer
q0 e1 (1) --(za)--> q1

sistem q1 durumundan sonra 1. zaman adımıyla q1 e2 (1) durumuna geçer
q1 --(za)--> q1 e2 (1)

sistem q1 e2 (1) durumundan sonra 2. zaman adımıyla q1 e2 (2) durumuna geçer
q1 e2 (1) --(za)--> q1 e2 (2)

sistem q1 e2 (2) durumundan sonra 3. zaman adımıyla q2 durumuna geçer
q1 e2 (2) --(za)--> q2

sistem q2 durumundan sonra 1 zaman adımıyla q15 durumuna geçer. Ara durum olusmaz...!
q2 --(za)--> q15

sistem q15 durumundan sonra 1. zaman adımıyla q15e32(1) durumuna geçer
q15 --(za)--> q15e32(1)

sistem q15e32(1) durumundan sonra 2. zaman adımıyla q15e32(2) durumuna geçer
q15e32(1) --(za)--> q15e32(2)

sistem q15e32(2) durumundan sonra 3. zaman adımıyla q16 durumuna geçer
q15e32(2) --(za)--> q16

q16 durumunda sistem cikmazi olusuyor

bilgi1.m (Süre bitmediği halde sistem çıkmaza girer ise)
verilen sürede oluşan durumlar
q0
q0 e1 (1)
q1
q1 e2 (1)
q1 e2 (2)
q2
q15
q15e32(1)
q15e32(2)
q16
verilen sürede meydana gelen olaylar
e1
za
za
e2
za
za
za
e27
za
e32
za
za
za

EK-2E Sistem çıkmazını bulan yazılım ve çalışması sonucu oluşan çıktı dosyası

deadlock.m

```
load all;

fid=fopen('deadlock_bilgi.m','w');     % sistem çıkmazı bilgilerini kaydetmek için
deadlock_bilgi.m dosyasını açar

Gama_new=Gama;                         % otomata matrisini kopyalar

line=[];                               % otomata matrisinde boş satır numaralarını tutmak
için oluşturulur

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i=1:size_Q                         % sistem çıkmazına neden olan durumları bulmak
için döngü
   C=Gama(i,:);

   if all(C==0)          %eğer tüm satır 0 ise yani sistem çıkmazı oluşuyorsa

     line(i)=i;          %eğer tüm satır 0 ise o satıra satır numarasını yaz

     sprintf('%s durumunda sistem cikmazi olusuyor',Q(i,:))
     fprintf(fid,'%s durumunda sistem cikmazi olusuyor\n',Q(i,:));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     F=Gama(:,i);        %sistem çıkmazının olduğu durumun otomata matrisindeki
sutununa bakar
     d=find(F);          %sistem çıkmazına neden olan durmuma götüren olayların
hangi durumdan sonra meydana geldiğini belirlemek için
     e=length(d);        %kaç tane durumdan sonra sistemin çıkmaza gittiğini
bulmak için
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
          for j=1:e      %sistem çıkmazına götüren olayları bulmak için ve dosyaya
durumları ve olayları yazırmak için kullanılan döngü
             f=F(d(j,:),:);
             sprintf('%s durumundan sonra %s olayının meydana gelmesi
engellenmeli ',Q(d(j,:),:),Sigma(f,:))
             fprintf(fid,'%s durumundan sonra %s olayının meydana gelmesi
engellenmeli \n',Q(d(j,:),:),Sigma(f,:));
```

```
            Gama_new(d(j,:),i)=0;  %kopyalanan otomata matrisi çıkmaza götüren
olay disable edilerek güncellenir
            tick=fix(Tau(f));
            if tick==0
              fprintf(fid,'%s --(%s)--> %s\n',Q(d(j,:),:),Sigma(f,:),Q(i,:));
            elseif tick==1
              fprintf(fid,'%s --(za)--> %s\n',Q(d(j,:),:),Q(i,:));
            else
              initial_state_name=Q(d(j,:),:);
              initial_event_name=Sigma(f,:);
              initial_state_name1=initial_state_name;
              for r=1:tick-1

next_state_name=[initial_state_name,initial_event_name,'(',num2str(r),')'];
                fprintf(fid,'%s --(za)-->
%s\n',initial_state_name1,next_state_name);
                initial_state_name1=next_state_name;
              end
              fprintf(fid,'%s --(za)--> %s\n',initial_state_name,Q(i,:));
            end
         end
      else
         continue

      end
      fprintf(fid,'\n\n',initial_state_name1,next_state_name);
   end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%
 line_new=[];         % otomata matrisinde boş satır numaralarını tutmak için
oluşturulur

for o=1:size_sigma         % maksimum olay sayısı kadar engellenecek olay olduğu
için güncellenen otomata matrisine bakmak için kullanılan döngü

   for i=1:size_Q     % sadece çıkmazın olduğu durumdan önceki duruma değil
daha önceki durumlarada bakmak için kullanılan döngü

      C_new=Gama_new(i,:);
      if all(C_new==0)    %eğer tüm satır 0 ise
         line_new(i)=i;  %eğer tüm satır 0 ise o satıra satır numarasını yaz

      end
   end

      if line==line_new  %güncellenen otomata matrisi ile orjinal otomata
matrisindeki tümü 0 olan satırlar eşit ise
```

```
            continue
        else
            z=find(line_new-line); % güncellenen otomata matrisi ile orjinal otomata
matrisindeki tümü 0 olan satırlar eşit değil ise
                              % sistem çıkmazına götüren durumdan önceki durumda
sistemi çıkmaza götüren olaydan başka olay meydana gelemiyorsa

            for n=1:length(z);
              F_new=Gama(:,z(n));       %sistem çıkmazından önceki durumun
otomata matrisindeki sutununa bakar
              d_new=find(F_new);         %sistem çıkmazından önceki duruma  neden
olan olayların hangi durumdan sonra meydana geldiğini belirlemek için
              e_new=length(d_new);       %kaç tane durumdan sonra sistemin
çıkmazdan önceki duruma gittiğini bulmak için
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
              for m=1:e_new     % sistem çıkmazını engellemek için engellenmesi
gerekli olan ekstra olayları ve durumları dosyaya yazmak
                f_new=F_new(d_new(m,:),:);
                sprintf('%s durumundan sonra %s olayının meydana gelmesi
engellenmeli ',Q(d_new(m,:),:),Sigma(f_new,:))
                fprintf(fid,'%s durumundan sonra %s olayının meydana gelmesi
engellenmeli \n',Q(d_new(m,:),:),Sigma(f_new,:));
                Gama_new(d_new(m,:),z(n))=0;
                tick=fix(Tau(f_new));

                if tick==0
                  fprintf(fid,'%s --(%s)-->
%s\n',Q(d_new(m,:),:),Sigma(f_new,:),Q(i,:));
                elseif tick==1
                  fprintf(fid,'%s --(za)--> %s\n',Q(d_new(m,:),:),Q(i,:));
                else
                  initial_state_name=Q(d_new(m,:),:);
                  initial_event_name=Sigma(f_new,:);
                  initial_state_name1=initial_state_name;
                  for r=1:tick-1

next_state_name=[initial_state_name,initial_event_name,'(',num2str(r),')'];
                    fprintf(fid,'%s --(za)-->
%s\n',initial_state_name1,next_state_name);
                    initial_state_name1=next_state_name;
                  end
                  fprintf(fid,'%s --(za)--> %s\n',initial_state_name,Q(i,:));

                end
              end
            end
```

```
      end
     line=line_new;
end
 fclose(fid);
save all;
```

deadlock_bilgi.m

```
q8  durumunda sistem cikmazi olusuyor
q6  durumundan sonra e14 olayının meydana gelmesi engellenmeli
q6  --(za)--> q6 e14(1)
q6 e14(1) --(za)--> q6 e14(2)
q6 e14(2) --(za)--> q6 e14(3)
q6  --(za)--> q8
q7  durumundan sonra e15 olayının meydana gelmesi engellenmeli
q7  --(za)--> q7 e15(1)
q7  --(za)--> q8
q9  durumundan sonra e16 olayının meydana gelmesi engellenmeli
q9  --(za)--> q9 e16(1)
q9  --(za)--> q8
q10 durumundan sonra e19 olayının meydana gelmesi engellenmeli
q10 --(e19)--> q8


q16 durumunda sistem cikmazi olusuyor
q13 durumundan sonra e31 olayının meydana gelmesi engellenmeli
q13 --(za)--> q13e31(1)
q13e31(1) --(za)--> q13e31(2)
q13e31(2) --(za)--> q13e31(3)
q13e31(3) --(za)--> q13e31(4)
q13 --(za)--> q16
q15 durumundan sonra e32 olayının meydana gelmesi engellenmeli
q15 --(za)--> q15e32(1)
q15e32(1) --(za)--> q15e32(2)
q15 --(za)--> q16
```

EK-2F Sistem çıkmazının meydana gelmesinin engellendiği otomatanın belirlenen zaman aralığında benzetimini yapan yazılımlar ve çalışması sonucu oluşan çıktı dosyaları

controlled_algoritma.m

```
% Bu fonksiyon ana program tarafından çağırılır.
% 1)
%Zamangecikmelerinin ebobu bulunarak ve birim gecikme zamanı olarak atanarak durumlar
%arasında meydana gelen zaman adımı sayısını gösteren U matrisi oluşturulur.

% 2)
% Kullanıcıdan girdileri isteyen ekranlar burada yaratılır

% 3)
% Oluşan durumları zaman adımlarına göre ekran yazar. Ayrıca uyarıları
% kullanıcıya ekrana yazdırarak bildirir.

% 4)
% Ekrana yazdırılan tüm yazıları bilgi.m dosyasında saklar

% 5)
% En son olarak oluşan durumları ve meydana gelen olayları sırası ile ayrı karakter
% yığınlarında saklar.

load all;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fid=fopen('controlled_bilgi.m','w');     %ekrana yazılan yazıları dosyada saklamak için bilgi.m dosyasını açar

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

set(0,'defaultfigurePosition',[300,300,50,50]);
prompt= {
        'Zaman dilimini saniye cinsinden giriniz:(sn)',...
        };
     baslik='Kontrol edilmiş zamanlandırılmış otomata algoritması';
  lineNo=1;

                                         %kullanıcıdan zaman diliminin istenmesi
def={
     '1'...
```

```
    };

answer=inputdlg(prompt,baslik,lineNo,def);

   if length(answer)==0,
     return;
   end

sure=str2num(answer{1,1});
sure2=fix(sure/du);                    % kullanın girdiği zamanın tamsayı olması
garantilemek
sure1=sure2(1,1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
x=1;                   %durum dizisinin yazılacağı R arrayini oluşturma
Rc=[];
Re=char(Rc);
R=cellstr(Re);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
y=1;                   %olay dizisinin yazılacağı T arrayinin oluşturma
Tc=[];
Te=char(Tc);
T=cellstr(Te);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
t=0;    % meydana gelen zaman adımı sayısını tutmak için
state=0; % kullanıcı ekranındaki ilk açılıştaki olay ve durum için default değerleri
atamak
event=1;
for n=1:1000000
   state1=num2str(state);
   event1=num2str(event);
set(0,'defaultfigurePosition',[300,300,50,50]);
prompt= {
       'Sistemin bulundugu durumu giriniz: q',...
       'Ateslenecek olayi giriniz: e',...
       };
     baslik='Kontrol edilmiş zamanlandırılmış otomata algoritması';
  lineNo=1;

                       % Kullanıcıdan olayların ve durmların girilmesinin istendiği
arabirimi oluşturma
def={
     state1...
     event1...
     };
```

```
answer=inputdlg(prompt,baslik,lineNo,def);

if length(answer)==0,
  break;
end

state=str2num(answer{1,1});
event=str2num(answer{2,1});
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

C=Gama_new((state+1),:); %olayın verilen durumdan sonra geçerli olup
olmadığının belirlenmesi
e=find(C==event);   %olayı otomata matrisinde duruma ait satırda arar

if n==1                  %başlangıç durumu R arrayine yazma
  R(x,:)=Q1((state+1),:);
  x=x+1;
end

if all(C==0)        %eğer tüm satır 0 ise yani sistem çıkmazı oluşuyorsa
  sprintf('%s durumunda sistem cikmazi olusuyor',Q((state+1),:))
  fprintf(fid,'%s durumunda sistem cikmazi olusuyor\n',Q((state+1),:));
  fclose(fid)
  return

elseif  isempty(e)    %eğer satırda olay yok ise yani geçerli durumdan sonra bu
olay tanımlı değil ise
   sprintf('%s      durumundan      sonra      %s      olayı      meydana
gelemez..!\n',Q((state+1),:),Sigma(event,:))
   fprintf(fid,'%s      durumundan      sonra      %s      olayımeydana
gelemez..!\n',Q((state+1),:),Sigma(event,:));

   set(0,'defaultfigurePosition',[300,300,50,50]);    %kullanıcıdan tekrar bir olay
istemek
prompt= {

      'Durumdan sonra ateslenecek dogru olayi giriniz: e',...
      };
    baslik=Q((state+1),:);
  lineNo=1;


def={
    ' ',...
    };

answer=inputdlg(prompt,baslik,lineNo,def);
```

```
if length(answer)==0,
  break;
end

  event=str2num(answer{1,1});
  e=find(C==event);

   if  isempty(e)    %eğer istenilen tanımlı değil ise
     sprintf('%s durumundan sonra %s olayı meydana
gelemez..!',Q((state+1),:),Sigma(event,:))
     fprintf(fid,'%s durumundan sonra %s olayı meydana
gelemez..!\n',Q((state+1),:),Sigma(event,:));
     fclose(fid)
     return
   end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

e=find(C==event); % kullanıcının verdiği tanımlı olayın otomata matrisinde
yerinin bulunması
tick=fix(U((state+1),e)); % bu olaya karşılık gelen zaman adımı sayısının
bulunması
initial_state_name=Q((state+1),:); % sistemin o anda bulunduğu durumun isminin
atanması
next_state_name=Q(e,:);  %sistemin gececeği bir sonraki durumun isminin
atanması
initial_event_name=Sigma(event,:); % şimdiki durumdan sonra geçerli olayın
isminin atanması

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%başta kullanıcının verdiği zaman aralığının birim zaman gecikmesinden
%küçük olduğu durumda sadece zaman gecikmesi 0 sn olan olayların
%meydana gelmesine izin verilmesi

if sure1==0 && tick==0
   sprintf('sistem %s durumundan sonra %s olayı meydana gelince %s durumuna
gecikme olmadan gecer',initial_state_name,initial_event_name,next_state_name)
   fprintf(fid,'sistem %s durumundan sonra %s olayı meydana gelince %s
durumuna gecikme olmadan
gecer\n',initial_state_name,initial_event_name,next_state_name);
   fprintf(fid,'%s --(%s)-->
%s\n\n',initial_state_name,initial_event_name,next_state_name);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% zaman gecikmesi 0 sn olan olayların meydana gelmesi
```

```
elseif tick==0 && sure1>0

   T(y,:)=Sigma1(event,:);
   y=y+1;

   sprintf('sistem %s durumundan sonra %s olayı meydana gelince %s durumuna
gecikme olmadan gecer',initial_state_name,initial_event_name,next_state_name)
   fprintf(fid,'sistem %s durumundan sonra %s olayı meydana gelince %s
durumuna gecikme olmadan
gecer\n',initial_state_name,initial_event_name,next_state_name);
   fprintf(fid,'%s --(%s)-->
%s\n\n',initial_state_name,initial_event_name,next_state_name);

   R(x,:)=cellstr(next_state_name);
   x=x+1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% 1 tane zaman adımına sahip olayların meydana gelmesi
elseif tick==1 && sure1>0

   T(y,:)=Sigma1(event,:);
   y=y+1;

   if t==sure1                    % geçen zamanın kontrol edilmesi
      sprintf('%d saniye bitti..!',sure)
      fprintf(fid,'%d saniye bitti..!\n',sure);
      fclose(fid)
      return
   end

   sprintf('sistem %s durumundan sonra 1 zaman adımıyla %s durumuna
geçer',initial_state_name,next_state_name)
   sprintf('Ara durum olusmaz...!')
   fprintf(fid,'sistem %s durumundan sonra 1 zaman adımıyla %s durumuna geçer.
Ara durum olusmaz...!\n',initial_state_name,next_state_name);
   fprintf(fid,'%s --(za)--> %s\n\n',initial_state_name,next_state_name);

   R(x,:)=cellstr(next_state_name);
   x=x+1;

   T(y,:)=cellstr('za');
   y=y+1;


   t=t+1;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 %zaman adımı 1 taneden fazla olan olayların meydana gelmesi

elseif sure1>0

   if t<sure1
     T(y,:)=Sigma1(event,:);
     y=y+1;
   end

   initial_state_name1=initial_state_name;
   for i=1:(tick-1)

     if t==sure1              % geçen zamanın kontrol edilmesi
        sprintf('%d saniye bitti..!',sure)
        fprintf(fid,'%d saniye bitti..!\n',sure);
        fclose(fid)
        return
     end

     next_state_name1=[initial_state_name,initial_event_name,'(',num2str(i),')'];
     sprintf('sistem %s durumundan sonra %d. zaman adımıyla %s durumuna
geçer',initial_state_name1,i,next_state_name1)
     fprintf(fid,'sistem %s durumundan sonra %d. zaman adımıyla %s durumuna
geçer\n',initial_state_name1,i,next_state_name1);
     fprintf(fid,'%s --(za)--> %s\n\n',initial_state_name1,next_state_name1);
     initial_state_name1=next_state_name1;

     T(y,:)=cellstr('za');
     y=y+1;

     R(x,:)=cellstr(next_state_name1);
     x=x+1;


     t=t+1;
   end

   i=tick;

   if t==sure1              % geçen zamanın kontrol edilmesi
      sprintf('%d saniye bitti..!',sure)
      fprintf(fid,'%d saniye bitti..!\n',sure);
      fclose(fid)
```

```
    return
  end

  sprintf('sistem %s durumundan sonra %d. zaman adımıyla %s durumuna geçer',initial_state_name1,i,next_state_name)
  fprintf(fid,'sistem %s durumundan sonra %d. zaman adımıyla %s durumuna geçer\n',initial_state_name1,i,next_state_name);
  fprintf(fid,'%s --(tick)--> %s\n\n',initial_state_name1,next_state_name);

  R(x,:)=cellstr(next_state_name);
  x=x+1;

  T(y,:)=cellstr('za');
  y=y+1;


  t=t+1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
else  % verilen zaman aralığı 0 sn iken zaman gecikmesi 0 sn'den farklı olan olayların meydana gelmesini engellemek
  sprintf('%d saniye geçerli bir süre değil',sure)
  return;
end

state=e-1;
end
```

controlled_cikis.m

```
fid=fopen('controlled_bilgi1.m','w'); % oluşan durumları ve meydana gelen olayları yazmak için bilgi1.m dosyasını açar

m=size(R);
size_R=m(1,1);
n=size(T);
size_T=n(1,1);

fprintf(fid,'verilen sürede oluşan durumlar\n');

for i=1:size_R
  Rc=char(R(i,:));
  fprintf(fid,'%s\n',Rc);
end

fprintf(fid,'verilen sürede meydana gelen olaylar\n');
```

```
for i=1:size_T
   Tc=char(T(i,:));
   fprintf(fid,'%s\n',Tc);
end

fclose(fid);
```

```
Gama_new=[0 1 0 5 0 8 0 0 0 0 0 0 0 0 0 0 0;
    0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 27 0;
    0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0;
    6 0 7 0 0 9 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 12 0 0 0 0 0 0 0 0 0;
    0 0 0 0 10 0 0 0 0 17 0 0 0 0 0 0 0;
    0 0 0 0 11 0 13 0 0 0 20 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 18 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 21 0 0 0 22 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 23 0 0 25 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 24 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0 26 0 0 0 0;
    0 0 28 0 0 0 0 0 0 0 0 0 0 30 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 29 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
```

```
Gama=[0 1 0 5 0 8 0 0 0 0 0 0 0 0 0 0 0;
    0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 27 0;
    0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0;
    6 0 7 0 0 9 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 12 0 0 0 0 0 0 0 0 0;
    0 0 0 0 10 0 0 0 14 17 0 0 0 0 0 0 0;
    0 0 0 0 11 0 13 0 15 0 20 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 18 16 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 21 19 0 0 22 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 23 0 0 25 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 24 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0 26 0 0 0 31;
    0 0 28 0 0 0 0 0 0 0 0 0 0 30 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 29 0 32;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
```

controlled_bilgi.m

sistem q0 durumundan sonra 1. zaman adımıyla q0 e1 (1) durumuna geçer
q0 --(za)--> q0 e1 (1)

sistem q0 e1 (1) durumundan sonra 2. zaman adımıyla q1 durumuna geçer
q0 e1 (1) --(za)--> q1

sistem q1 durumundan sonra 1. zaman adımıyla q1 e2 (1) durumuna geçer
q1 --(za)--> q1 e2 (1)

sistem q1 e2 (1) durumundan sonra 2. zaman adımıyla q1 e2 (2) durumuna geçer
q1 e2 (1) --(za)--> q1 e2 (2)

sistem q1 e2 (2) durumundan sonra 3. zaman adımıyla q2 durumuna geçer
q1 e2 (2) --(za)--> q2

sistem q2 durumundan sonra 1 zaman adımıyla q3 durumuna geçer. Ara durum olusmaz...!
q2 --(za)--> q3

q3 durumundan sonra e12 olayı meydana gelemez..!
sistem q3 durumundan sonra 1. zaman adımıyla q3 e4 (1) durumuna geçer
q3 --(za)--> q3 e4 (1)

sistem q3 e4 (1) durumundan sonra 2. zaman adımıyla q3 e4 (2) durumuna geçer
q3 e4 (1) --(za)--> q3 e4 (2)

sistem q3 e4 (2) durumundan sonra 3. zaman adımıyla q3 e4 (3) durumuna geçer
q3 e4 (2) --(za)--> q3 e4 (3)

sistem q3 e4 (3) durumundan sonra 4. zaman adımıyla q5 durumuna geçer
q3 e4 (3) --(za)--> q5

q5 durumundan sonra e9 olayı meydana gelemez..!
10 saniye bitti..!

Controlled_bilgi1.m
verilen sürede oluşan durumlar
q0
q0 e1 (1)
q1
q1 e2 (1)
q1 e2 (2)
q2
q3
q3 e4 (1)
q3 e4 (2)
q3 e4 (3)
q5
verilen sürede meydana gelen olaylar
e1
za
za
e2
za
za
za
e3
za
e4
za
za
za
za
e12

EK-2G İstenilen durumların bulunduğu en kısa durum dizisini ve en kısa sürede oluşan durum dizisini bulan yazılım ve çalışması sonucu oluşan çıktı dosyası

desired.m

```
load all;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

set(0,'defaultfigurePosition',[300,300,50,50]);
prompt= {
        'istenilen durum sayisini giriniz',...
        };
     baslik='Zamanlandırılmış otomata algoritması';
   lineNo=1;

                                                            %kullanıcıdan istenilen durum sayısını
istemek
def={
     '1'...

     };

answer=inputdlg(prompt,baslik,lineNo,def);

    if length(answer)==0,
      return;
    end


quan=str2num(answer{1,1});
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i=1:quan

    set(0,'defaultfigurePosition',[300,300,50,50]);
    prompt= {
           'istenilen durumun adini giriniz:  q',...
            };
           baslik='Zamanlandırılmış otomata algoritması';
           lineNo=1;
```

```
                                        %kullanıcıdan durumların istenmesi
        def={
        '1'...

        };

    answer=inputdlg(prompt,baslik,lineNo,def);

      if length(answer)==0,
        return;
      end


  desired_state(i)=str2num(answer{1,1})+1;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i=1:size_Q                    %her durumdan sonra meydana gelebilecek olay
sayısının bulunması
  [x,y,z]=find(Gama(i,:));
  a=size(z);
  event_number(i)=a(1,2);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
b=max(event_number);            %herhangi bir durumdan sonra sistemin
geçebileceği durum veya durumların bulunması
state=zeros(size_Q,b);

for i=1:size_Q
  for j=1:event_number(i)
  [x,y,z]=find(Gama(i,:));
  state(i,j)=y(j);
  end
end


times=zeros(size_Q,b);          %herhangi bir durumdan başka bir duruma
geçerken geçen sürenin bulunması
for i=1:size_Q
  for j=1:event_number(i)
  [x,y,z]=find(Gama(i,:));
  times(i,j)=D(z(j));
  end
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%
t=1;                %başlangıç durumu qo'dan sonra 2 elemanlı ilk durum dizisi veya
dizilerinin oluşturulması
vek(t)={[1]};
time_vek(t)=0;
for i=1:event_number(1)
   p=vek(1);
   p1=cell2mat(p);
   r=length(p1);
   m=zeros(1,(r+1));
   for j=1:r
      m(j)=p1(j);
   end
   m(r+1)=state(1,i);
   t=t+1;
   vek(t)={m};
   time_vek(t)=times(1,i);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
o=length(vek)+1;
t=2;                %sisteme ait olası tüm durum dizilerinin oluşturulması ve vek adlı
yığına kaydedilmesi
d=1;

for i=1:1000*quan
   p=vek(t);
   p1=cell2mat(p);
   r=length(p1);
   last_state=p1(r);

if event_number(last_state)==0
   t=t+1;
   continue
else
   for k=1:event_number(last_state)
      n=zeros(1,(r+1));
      for j=1:r
         n(j)=p1(j);
      end
      n(r+1)=state(last_state,k);
      vek(o)={n};
      time_vek(o)=time_vek(t)+times(last_state,k);
      o=o+1;
   end
```

```
end
   t=t+1;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%
desired_vek=vek;    %kullanıcının istediği durumların bulunduğu dizilerin
bulunması
for i=1:t
   for j=1:quan
      p=desired_vek(i);
      p1=cell2mat(p);
      k=find(p1==desired_state(j));
      if all(k==0)
         desired_vek(i)={[]};
      end
   end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%
n=1;              %istenilen durumların bulunduğu dizilerden en kısa olanının
bulunması

for i=1:t
   p=desired_vek(i);
   p1=cell2mat(p);
   if length(p1)~=0
      boyut(n)=length(p1);
      addr(n)=i;
      time_addr(n)=i;
      desired_time_vek(n)=time_vek(i);
      n=n+1;
   end
end

[C1,I1]=min(boyut);
cev1=cell2mat(desired_vek(addr(I1)));
[C2,I2]=min(desired_time_vek);
cev2=cell2mat(desired_vek(time_addr(I2)));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
fid=fopen('desired_bilgi.m','w');   %bilgilerin desired_bilgim dosyasına yazılması

fprintf(fid,'q0 durumundan sonra ulaşılması istenilen durumlar\n');
for i=1:quan
```

```
    fprintf(fid,'%s\n',Q((desired_state(i)),:));
end

if I1==I2

    fprintf(fid,'istenilen durumların oluştuğu en kısa durum dizisi\n');
    boy=length(cev1);
    for i=1:boy
        fprintf(fid,'%s\n',Q((cev1(i)),:));
    end

    fprintf(fid,'\n');
    fprintf(fid,'istenilen en kısa durum dizisi oluşurken meydana gelen olaylar\n');
    for i=1:(boy-1)
        event=Gama(cev1(i),cev1(i+1));
        fprintf(fid,'%s\n',Sigma(event,:));

        for j=1:(D(event)/du)
            fprintf(fid,'za\n');
        end
    end

    time=desired_time_vek(I2);
    fprintf(fid,'\n');
    fprintf(fid,'istenilen durum dizisi oluşurken geçen minimum zamam %g
saniyedir.\n',time);
    fprintf(fid,'istenilen durum dizisi oluşurken meydana gelen zaman adımı sayısı
%g tanedir.\n',(time/du));

    fclose(fid)

else

    fprintf(fid,'istenilen durumların oluştuğu en kısa durum dizisi\n');
    boy=length(cev1);
    for i=1:boy
        fprintf(fid,'%s\n',Q((cev1(i)),:));
    end

    fprintf(fid,'\n');
    fprintf(fid,'istenilen en kısa durum dizisi oluşurken meydana gelen olaylar\n');
    for i=1:(boy-1)
        event=Gama(cev1(i),cev1(i+1));
        fprintf(fid,'%s\n',Sigma(event,:));

        for j=1:(D(event)/du)
            fprintf(fid,'za\n');
        end
```

```
    end

    time1=desired_time_vek(I1);
    fprintf(fid,'\n');
    fprintf(fid,'istenilen durum dizisi oluşurken geçen zamam %g
saniyedir.\n',time1);
    fprintf(fid,'istenilen durum dizisi oluşurken meydana gelen zaman adımı sayısı
%g tanedir.\n',(time1/du));

    fprintf(fid,'\n');
    fprintf(fid,'istenilen durumların en kısa sürede oluştugu durum dizisi\n');
    boy=length(cev2);
    for i=1:boy
        fprintf(fid,'%s\n',Q((cev2(i)),:));
    end

    fprintf(fid,'\n');
    fprintf(fid,'en kısa sürede olusan durum dizisi oluşurken meydana gelen
olaylar\n');
    for i=1:(boy-1)
        event=Gama(cev2(i),cev2(i+1));
        fprintf(fid,'%s\n',Sigma(event,:));

        for j=1:(D(event)/du)
            fprintf(fid,'za\n');
        end
    end

    time2=desired_time_vek(I2);
    fprintf(fid,'\n');
    fprintf(fid,'istenilen durum dizisi oluşurken geçen zamam %g
saniyedir.\n',time2);
    fprintf(fid,'istenilen durum dizisi oluşurken meydana gelen zaman adımı sayısı
%g tanedir.\n',(time2/du));

     fclose(fid)

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

desired_bilgi.m

```
q0 durumundan sonra ulaşılması istenilen durumlar
q13
q9
q7
q4
```

q15
istenilen durumların oluştuğu en kısa durum dizisi
q0
q5
q7
q6
q9
q7
q4
q2
q15
q14
q13

istenilen en kısa durum dizisi oluşurken meydana gelen olaylar
e8
za
za
za
za
za
e12
za
za
e13
za
e17
e18
za
za
za
e11
za
za
e7
e27
za
e29
za
za
e30
za
za
za

istenilen durum dizisi oluşurken geçen minimum zamam 19 saniyedir.
istenilen durum dizisi oluşurken meydana gelen zaman adımı sayısı 19 tanedir.