İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

ÇOK FONKSİYONEL ZAMANLI RENKLİ PETRİ AĞLARI YAZILIMI

YÜKSEK LİSANS TEZİ

Hakkı KURUMAHMUT

Anabilim Dalı: Elektronik ve Haberleşme Mühendisliği

Programı: Elektronik Mühendisliği

HAZİRAN 2010

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

ÇOK FONKSİYONEL ZAMANLI RENKLİ PETRİ AĞLARI YAZILIMI

YÜKSEK LİSANS TEZİ

Hakkı KURUMAHMUT
(504071229)

Tezin Enstitüye Verildiği Tarih : 7 Mayıs 2010
Tezin Savunulduğu Tarih : 11 Haziran 2010

Tez Danışmanı : Doç. Dr. Mürvet KIRCI (İTÜ)
Diğer Jüri Üyeleri : Prof. Dr. Ali Emre HARMANCI (İTÜ)
Prof Dr. Ece Olcay GÜNEŞ (İTÜ)

HAZİRAN 2010

ÖNSÖZ

Tez çalışmamda değerli zamanını esirgemeyen, bilgi ve önerilerini benimle her zaman paylaşan ve teşvik eden danışmanım Doç. Dr. Mürvet KIRCI' ya teşekkürü bir borç bilirim. Ayrıca çalışmalarım sırasında destek ve katkılarını esirgemeyen ağabeyim Yüksek Mühendis Bayram KURUMAHMUT ve kardeşim Samet KURUMAHMUT' a ve bugünleri görebilmemde önemli katkıları olan anneme ve babama ayrıca teşekkür ederim.

Haziran 2010

Hakkı KURUMAHMUT
Elektronik ve Haberleşme Mühendisi

İÇİNDEKİLER

Sayfa

ÖNSÖZ ... **iii**
İÇİNDEKİLER ... **v**
KISALTMALAR ... **vii**
ÇİZELGE LİSTESİ ... **ix**
ŞEKİL LİSTESİ ... **xi**
ÖZET ... **xiii**
SUMMARY ... **xv**
1. GİRİŞ ... **1**
1.1 KLASİK PETRİ AĞLARI ... 6
1.1.1 Petri Ağlarına İlişkin Tanımlar ... 9
1.1.1.1 Sınırlılık (*Boundedness*) 9
1.1.1.2 Güvenlik (*Safeness*) 9
1.1.1.3 Tutarlılık (Conservation) 9
1.1.1.4 Canlılık (Liveness) 9
1.2 ZAMANLI PETRİ AĞLARI ... 10
1.2.1 Belirli Zamanlı Petri Ağları ... 13
1.2.2 Rastlantısal Zamanlı Petri Ağları ... 19
1.3 RENLİ PETRİ AĞLARI ... 23
1.4 ZAMANLI RENKLİ PETRİ AĞLARI ... 26
2. GELİŞTİRİLEN YAZILIM ... **32**
3. SONUÇ VE ÖNERİLER ... **63**
KAYNAKLAR ... **65**
EKLER ... **67**

KISALTMALAR

bz : Birim Zaman
BZPA : Belirli Zamanlı Petri Ağı
KPA : Klasik Petri Ağı
PA : Petri Ağı
PLC : Programlanabilir Mantıksal Kontrolör
RZPA : Rastlantısal Zamanlı Petri Ağı
RPA : Renkli Petri Ağı
VLSI : Çok Büyük Ölçekli Entegre Devre
ZPA : Zamanlı Petri Ağı
ZRPA : Zamanlı Renkli Petri Ağı

ÇİZELGE LİSTESİ

Sayfa

Çizelge 2.1 : Renkli Petri Ağı Örneğinin Bileşenleri ve Tanımları 33
Çizelge 2.2 : Geliştirilen Yazılımın Anahtar Kelimeleri 37
Çizelge 2.3 : ÖRNEK-1'in Rastlantısal Değişkenli Analiz Sonucu Ortalama Zamanlar 56
Çizelge 2.4 : Susturucu Egzoz Üretim Modelinin Bileşenleri ve Tanımları 59

ŞEKİL LİSTESİ

Sayfa

Şekil 1.1 : Basit Bir Petri Ağı 7
Şekil 1.2 : Basit Bir Petri Ağı Örneğinin Erişebilirlik Grafiği 8
Şekil 1.3 : Sonsuz Ateşlenmeli Basit Petri Ağı Örneğinin Erişebilirlik Grafiği.....10
Şekil 1.4 : Belirli Geçişleri Zamanlı Basit Bir Petri Ağı.....13
Şekil 1.5 : Belirli Geçişleri Zamanlı Basit Bir Petri Ağı Zaman Diyagramları.....14
Şekil 1.6 : Belirli Geçişleri e_t, f_t, w_t Zamanlı Basit Bir Petri Ağı 16
Şekil 1.7 : Belirli Geçişleri e_t, f_t, w_t Zamanlı Basit Petri Ağı Zaman Diyagramı 17
Şekil 1.8 : Belirli Yerleri Zamanlı Basit Petri Ağı.....17
Şekil 1.9 : Belirli Yerleri Zamanlı Basit Petri Ağı Analizi.....17
Şekil 1.10 : Geçişleri ve Yerleri Zamanlı Petri Ağları Arasında Geçiş.....18
Şekil 1.11 : Okları Zamanlı Basit Petri Ağı.....18
Şekil 1.12 : Okları Zamanlı Basit Petri Ağı Zaman Diyagramı.....19
Şekil 1.13 : Eksponansiyel Dağılımın Olasılık Yoğunluk Fonksiyonu(*pdf*) 20
Şekil 1.14 : Varış zamanı ile oran arasındaki ilişki.....20
Şekil 1.15 : Eksponansiyel Dağılımın Olasılık Yoğunluk Fonksiyonu Grafiği.....20
Şekil 1.16 : Eksponansiyel Dağılımın Birikimli Dağılım Fonksiyonun(*cdf*) 20
Şekil 1.17 : Eksponansiyel Dağılımın Birikimli Dağılım Fonksiyonun Grafiği 21
Şekil 1.18 : Oran ve Varış Hızı Hesaplama Örneği 21
Şekil 1.19 : Rastlantısal Bir Petri Ağının Markov Zinciri Analizi Örneği 22
Şekil 1.20 : Çift İşlemcili Üretim Sisteminin Klasik Petri Ağı Modeli.....23
Şekil 1.21 : Çift İşlemcili Üretim Sisteminin Renkli Petri Ağı Modeli 24
Şekil 1.22 : Renkli Petri Ağı Örneği 25
Şekil 1.23 : Geliştirilen Algoritma Geçişlere Ait Durum Geçiş Diyagramı.....28
Şekil 1.24 : Basit Zamanlı Renkli Petri Ağı 28
Şekil 2.1 : Erişebilirlik Ağacı Çıkarılacak Renkli Petri Ağı Örneği(ÖRNEK–1) 35
Şekil 2.2 : Erişebilirlik Ağacı Çıkarma Giriş Dosyası Örneği.....36
Şekil 2.3 : Benzer Bir Yazılımın Giriş Dosyası Örneği 42
Şekil 2.4 : ÖRNEK-1'in Zamansız Analiz Sonucu Bulunan Erişilebilir Durumları .44
Şekil 2.5 : ÖRNEK-1'in Zamansız Analiz Aranan İşaretleme Sonuçları 45
Şekil 2.6 : ÖRNEK-1'in Zamansız Analiz Erişilebilirlik Grafiği.....46
Şekil 2.7 : CPN-Tool'un Yemek Yiyen Filozof Örneği (ÖRNEK–2) 47
Şekil 2.8 : Yemek Yiyen Filozof Örneği Giriş Dosyası.....48
Şekil 2.9 : Yemek Yiyen Filozof Örneği CPN-Tool Durum Uzayı.....48
Şekil 2.10 : Yemek Yiyen Filozof Örneği Erişebilirlik Grafiği.....49
Şekil 2.11 : Yemek Yiyen Filozof Örneği Erişebilir Durumlar.....50
Şekil 2.12 : Örnek–1 Zamansız Koşullu Analiz Erişilebilir Durumlar 51
Şekil 2.13 : Örnek–1 Zamansız Koşullu Analiz Erişilebilirlik Grafiği.....52
Şekil 2.14 : Örnek–1'in Belirli Zamanlı Analizi Erişilebilir Durumları 55
Şekil 2.15 : Örnek–1'in Belirli Zamanlı Analizi Erişilebilirlik Diyagramı.....55
Şekil 2.16 : Egzoz Bileşenleri.....57

Şekil 2.17 : Susturucu Egzoz Üretim Aşamaları RPA Modeli 58
Şekil 2.18 : Susturucu Egzoz Örneği Giriş Dosyası 62

ÇOK FONKSİYONEL ZAMANLI RENKLİ PETRİ AĞLARI YAZILIMI

ÖZET

Bu çalışmada, ayrık olaylı ve eş zamanlı sistemleri modellemede ve analiz etmede kullanılan Petri Ağı sistemlerinin gelişmiş tipleri olan Renkli Petri Ağları, Zamanlı Petri Ağları ve bunların bileşimi olan Zamanlı Renkli Petri Ağları ele alınmıştır. Ayrıca Rastlantısal Zamanlı Renkli Petri Ağları sistemleri de incelenmiştir. Petri Ağların incelenmesi ve analizi için bir algoritma önerilmiştir. Ayrıca bu algoritmayı kullanan çok fonksiyonel bir yazılım geliştirilmiştir.

Giriş bölümünde ilk olarak Petri Ağlarının ne olduğu ve bu alanda yapılan çalışmaların neler olduğuna değinilmiştir. Petri Ağların temel yapıları olan yer, geçiş, oklar ve jetonlardan, Petri Ağlarının matematiksel gösterimleri ve ateşleme kavramından bahsedilmiştir. İşaretleme, Erişebilirlik Ağacı ve Erişebilirlik Grafiği gibi kavramlar açıklanmıştır. Ayrıca Zamanlı Petri Ağları'nın Klasik Petri Ağlarından farkları, Zamanlı Petri Ağlarının çeşitleri ve Rastlantısal Petri Ağları'ndan bahsedilmiştir. Renkli Petri Ağlarının, Klasik Petri ağlarından farkları ve Renkli Petri Ağlarının üstünlüklerinden de bahsedilmiştir. Önerilen algoritma ve güçlü yanlarından bahsedilmiştir.

İkinci bölümde geliştirilen yazılım üzerinedir. Yazılımın giriş dosya formatı(anahtar kelimeler ve parametrelerinin açıklanması) ve çıkış dosyasının içeriği(işaretlemeler, ateşlenme dizileri, aranan ve kapsayan işaretlemelerin bulunması) açıklanmıştır. Daha sonra yazılım kullanarak analiz edilmiş Petri Ağı örnekleri verilmiştir. Var olan benzeri yazılımlarla karşılaştırılması yapılmıştır.

Son bölüm olan üçüncü bölümde yapılan çalışmanın sonuçları ve gelecekte yapılabilecek çalışmalar için algoritma ve yazılımla ilgili önerilerde bulunulmuştur.

HIGLY CUSTOMIZABLE PETRI NET TOOL

SUMMARY

In this thesis, Petri Nets (PN) used in modeling/analyzing discrete event and concurrent systems are focused on. The following PN types are investigated in this work: Colored Petri Nets (CPN), Timed Petri Nets (TPN), Timed Colored Petri Nets (TCPN), and Stochastic Timed Petri Net (STPN). The main goal of the thesis is to design an algorithm to analyze PNs and implement this algorithm by writing a tool.

In the first chapter, PNs are described and available background works are given. Also, the basic components (place, transition, arrow, and tokens) of a PN, PN terminology (firing, marking, reachability graph, reachability tree), and PN related mathematical expressions are detailed. Furthermore, the differences in descriptions between basic PNs and TPNs, the types of TPNs, SPNs are pointed out. Moreover, the followings are also explained: CPNs, the differences in descriptions between basic PNs and CPNs, and strengths of CPNs over basic PNs. Finally, the proposed algorithm and its strengths are stated.

The second chapter focuses on the implemented tool. Input file details (keywords and their parameters, etc.) and output file contents (markings, firing groups, searched and found markings, etc.) are described. Then, example PNs are analyzed by using the tool. The results from this written new tool are compared with the ones, got from the existing similar tools, in terms of strengths and weakness.

In the last chapter, the obtained results of the thesis and the possible future works for the thesis (algorithm and tool) are listed.

1. GİRİŞ

Bilgisayar teknolojisinde yaşanan hızlı gelişmeler, imalat sistemlerinde, haberleşme ağlarında, robotlar üzerine alanlarda birçok uygulamaların ortaya çıkmasına katkı sağlamıştır. Bu alanlar, ayrık olay sistemleri (*discrete event systems*) olarak adlandırılmaktadır. Ayrık olay sistemlerinin içerdiği paralellik, asenkron çalışma, belirsizlik (*non-determinism*) gibi özelliklerin, ayrık eşitliklerle modellenen sürekli veya senkron ayrık değişkenlerle ilgilenen geleneksel kontrol teorisi ile tanımlanmasının zorluğu sebebiyle, ayrık olay sistemleri, modern endüstriyel sistemlerin kontrol problemleriyle ilgili yeni bir disiplin olarak ortaya çıkmıştır. Başlarda bu tür sistemler çok karmaşık olmadıkları için kontrolleri deneme yanılmayla ve kontrol mühendisinin becerisi sayesinde gerçekleştirilebilmekteydi. Fakat geliştirilen ayrık olay sistemleri daha karmaşık hale geldikçe, bu sistemlerin daha etkin bir şekilde formal olarak tasarlanması ve gerçekleştirilmesi daha fazla önem kazandı. İlk ciddi çalışmaların sonucunda, sonlu durum makineleri (*finite state machine*) ortaya atılmıştır. Fakat sistemlerin büyümesiyle birlikte, kullanılması gereken durum sayısının çok fazla artması ve sonlu durum makineleriyle modellenen bir sistemin grafik olarak anlaşılmasının güçlüğü gibi sebeplerden dolayı, bu yöntemler hala kullanılmakla birlikte pratik olmaktan uzaktır. Alternatif bir yöntemse Petri Ağı kullanımıdır. Kolay ve grafiksel gösterimi ve bu grafiksel gösterimin güçlü matematiksel altyapısıyla öne çıkan kontrol yöntemidir. İlk kez bir Alman matematikçi **Carl A. Petri** [1] tarafından 1962'de ortaya atılmıştır.

Carl Petri'nin teorisi 1968 ve 1976 yılları arasında araştırmacılar tarafından geliştirilmiştir. Birçok alt bileşen sistemlerine sahip kombine bir sistemin modellemesi amacıyla kullanılabilen Petri Ağlarının kullanım alanları üretim tesisi otomasyonu, sistem performans ölçümleri, haberleşme protokolleri, dağılmış yazılım sistemlerinin özelliklerinin belirlenmesi ve tasarımı, iş akış modellemelerin, PLC sistemlerin ve VLSI devrelerinin performans değerlendirmesi gibi alanlar olabilmektedir.[2]

Petri Ağların temel elemanları yerler (*places*), geçişler (*transitions*), oklar (*arcs*) ve jeton(*tokens*)lardır. Yerler durumları ifade ederken geçişlerde olayları ifade etmektedir. Oklar, durumlar ve olaylar arası ilişkiyi ifade eder. Ateşlenme koşulu sağlandığında yani geçişlerin giriş yerlerinde gereken sayıdan daha fazla jeton bulunduğunda geçiş, çıkış yerlerine yeni jetonlar üretir. Giriş yerlerinden jetonların eksilmesi ve çıkış yerlerine jetonların eklenmesi aynı anda olmaktadır. Bu ateşlenme tipine atomik ateşlenme (*atomik firing*) denilmektedir.

Carl Petri'nin zamandan bağımsız olarak kurduğu Petri Ağı modeli, performans analizinde yetersiz kalmıştır.[3] Bu nedenle 1974 yılında **Ramchandani** [4] olayların temsil edildiği geçişlere zaman gecikmesi vererek Zamanlanmış Petri Ağı kavramını ortaya atmıştır. Atanan negatif olmayan zaman değeri, geçişin ateşlenme gecikmesi (*firing delay*) olarak adlandırılır. Bu ateşlenme gecikmesinin başında giriş yerlerinden jetonlar hemen eksilirken bu sürenin sonunda çıkış yerlerine jetonlar hemen üretilir.

Coolahan ve **Roussopoulos** [5] tarafından yapılan çalışmalarda ise geçişlere değil yerlere zaman gecikmesi verilmiştir. Bu atama yapılırken işlemin, hareketin yerler üzerinde gerçekleştiği ve zamanın burada harcanması gerektiği kabulüyle hareket etmişlerdir. Bu yaklaşımla jetonlar elverişli ve elverişsiz olarak ikiye ayrılmıştır. Bir yere gelen jetonlar başlangıçta elverişsiz iken zaman gecikmesi kadar süre sonra elverişli hale gelmektedir. Elverişsiz jetonlar o yer tarafından kullanılamayıp sadece elverişli jetonlar kullanılabilmektedir. Geçişleri Zamanlandırılmış Petri Ağı'yla kurulan modellerden, Yerleri Zamanlandırılmış Petri Ağı'yla kurulan modellere ek yerler eklenerek modellemeler arası geçişler yapılabilinmektedir. [3]

Zue ve **Denton** [6] tarafından yapılan çalışmada ise oklara zaman gecikmesi atanmıştır. Bu yaklaşımda geçişe ait her giriş okuna ve her çıkış okuna belirli bir zaman etiketi atanmıştır. Zaman etiketi atanmayan okların zaman etiketlerinin sıfır olacağı varsayılmıştır. Bu yaklaşımın daha genel olduğunu ve daha karmaşık sistemleri modellemede kullanılabileceğinden bahsedilmiştir. Zhu ve Denton bu çalışmayı su şekilde yorumlamışlardır: geçişe ait tüm giriş oklarını kazılacak tünel gibi, çıkış oklarını ise giriş tünellerinin işi tamamlandıktan sonra jetonların kendi üzerinden geçeceği tünel gibi düşünülmüştür. Bir yer bir jeton kazandığı zaman, çıkış geçişine doğru olacak şekilde bir tünel oluşturmaya başlar ve bu tünelin tamamlanma süresi ise oklara atanan zaman gecikmesi kadardır. Eğer tünel oluşturması biterse, onun giriş yerinde jeton olduğu süre boyunca açık olarak kalacaktır. Geçiş sadece bütün giriş yerlerinden oluşturulmaya başlanmış olan tüneller bittiğinde mümkün olacaktır.[7]

Merlin [8] tarafından önerilen çalışmada ise ağdaki geçişlere bir zaman aralığı atanmıştır. [α^S, β^S] şeklinde bir zaman aralığı atanmış olup α^S zamanı, geçisin mümkün olduktan sonra, ateşlenmeden önce bekleyeceği minimum zamanı β^S zamanı ise, geçisin mümkün olduktan sonra, ateşlenmeden önce bekleyeceği maksimum zaman olarak değerlendirilmiştir. Geçiş bu iki zaman aralığı içindeki herhangi bir zamanda ateşlenebilmektedir. Ateşleme için herhangi bir gecikmeye yer verilmemiştir. Eğer geçiş, β^S zamanından önce ateşlenmemiş ise geçiş β^S zamanında ateşlenmeye zorlanır. [7]

Şimdiye kadar anlatılan çalışmalarda zaman ifadeleri belirli(*deterministic*) olarak atanmıştır. Ancak olayların oluş süresi her zaman aynı olmayabilir. Rastgele olarak Zamanlanmış Petri Ağları, Klasik Petri Ağı yaklaşımından geçişlere rastgele ateşlenme zamanları atanarak elde edilmiştir.[7]

Natkin [9] ve **Molloy** [10] tarafından Rastlantısal Petri Ağları üzerine yapılmış çalışmada, ağda yer alan her geçişe belirli olmayan her bir geçiş için ayrı ayrı olacak şekilde üstel olarak dağıtılmış ateşlenme oranı adı verilen bir değer atanmıştır. Atanan bu değer ile Petri Ağı yaklaşımının sürekli zamanlı Markov zinciri (*Continuous Time Markov Chain*) ile eşdeğer olduğu gösterilmiştir. Rastlantısal Petri Ağı' nın Erişilebilirlik Ağacı' ndaki her durumun, geçişlere üstel olarak atanan ateşlenme oranlarıyla nasıl bulunacağı ve hangi olasılıklar içinde sistemin durumlarına erişileceği gösterilmiştir.[7]

Ajmone, Balbo Conte [11] tarafından yapılan çalışmada ise Rastlantısal Petri Ağı'ndaki geçişlere üstel dağıtılmış ateşlenme süresinin yanında belirli olarak sıfıra eşit olan zaman gecikmesinin atandığı Genelleştirilmiş Rastlantısal Petri Ağları tanımlanmıştır (*Generalized Stochastic Petri Nets-GSPNs*). Bu yaklaşımla, Erişilebilirlik Ağacında bulunan her durumun hangi olasılıklar içerisinde erişilebilineceği ve erişilen her durum üzerinden sistem performansı belirlemek üzerine çeşitli yaklaşım kıstasları ortaya konmuştur.

Bowden Fred [12] tarafından Tutma Süresine Sahip Petri Ağları(*Holding Time Petri Nets*) ortaya atılmıştır. Coolahan ve Roussopoulos yaptıkları çalışmalara benzer şekilde yerlere zaman gecikmesi verip jetonları elverişli ve elverişsiz olarak ayırmıştır. Tutma süresi(*Holding Time*) sonunda elverişsiz olan jetonlar elverişli hale gelmektedir.

Üç tip ateşlenme vardır. Birincisi tek ateşlenme(*single server*), ikincisi k-kez ateşlenme(*multible server*) ve üçüncüsü sonsuz ateşlenme(*infinite server*). Bir geçişin ateşlenmesi bitimine kadar paralel olarak yeniden ateşlenemiyorsa tek ateşlenme, k-kez paralel ateşlenmesine izin veriliyorsa k-kez ateşlenme, eğer sonsuz kez paralel ateşlenebiliyorsa sonsuz ateşlenme denir.

Zafer Hüseyin Ergan [13] tarafından yapılan çalışmada Ramchandani'nin modeline uygun olarak zaman gecikmeleri geçişlere sabit olarak vermiştir. Petri Ağının sınırlı olduğunu kabul ederek yaptığı çalışmasında geçişlerin çoklu ateşlenmesine izin verilmiştir. Mevcut ateşleme devam ederken eğer ateşlenme koşulunu sağlayacak giriş yerlerinde hala jeton varsa paralel olarak yeni bir ateşlenme daha başlatılır.

Oktay Yurttakal [7] tarafından yapılan çalışmada Zafer Hüseyin Ergan modeli geliştirilmiştir. Kurulan yeni modelde ok ağırlıklarının birden fazla olmasına izin verilmiştir. Ateşlenme derecesi hesaplamaları yapılaraktan çelişki durumlarına çözümler sunmuştur.

Ahmet Yılmaz [3] tarafından yapılan çalışmada Belirli Zamanlı Petri Ağlarını algoritmaya dayalı olarak analiz etmek için ok ağırlıklarının birden fazla olduğu çelişki durumlarına öncelik atamasıyla çözüm bulunduğu bir model sunulmuştur ve bu modelli kullanan C yazılımı hazırlanmıştır.

Buraya kadar Zamanlı Petri Ağlarından bahsedilmiştir. Ayrıca ilk kez 1970 yıllarının başlarında **Kurt Jensen** [14] tarafından jetonlara ve geçişlere renkler vermiştir. Böylece büyük sistemlerin modellenmesinde karşılaşılan Klasik Petri Ağları' nındaki karmaşık azaltılmıştır. Daha az yer ve geçiş ile aynı sistemler modellenebilmektedir.

Serkan Yorulmaz [2] tarafından yapılan çalışmada Renkli Petri Ağlarını analiz etmek için algoritma önerilmiştir. Bu algoritma kullanılarak geliştirilen C++ yazılımı sayesinde Renkli Petri Ağlarının analizi yapılmaktadır. Ayrıca P-değişmezleri analizi de yapılmaktadır.

Bu çalışmada Klasik Petri Ağlarını, Zamanlı Petri Ağlarını ve Renkli Petri Ağlarını ya da bileşimi olan Zamanlı Renkli Petri Ağlarını zamansız, belirli veya rastlantılı zamanlı analiz eden çok fonksiyonel algoritma verilmiştir. Ayrıca bu algoritmayı kullanan C++ dilinde yazılmış "Çok Fonksiyonel Zamanlı Renkli Petri Net" yazılımı da sunulmuştur. Bu algoritmada ok ağırlıklarının birden fazla olmasına izin verilmiştir. Çelişki durumlarında öncelik atamasına göre çözümler sunulmuştur. Hazırlanan yazılım ile çelişki durumları kullanıcıya bildirilmiştir. Daha önce yapılan çalışmalarda giriş dosyalarının formatı ikili sayılardan(0'lar ve 1'ler) oluşuyordu. Bu yapı çok karmaşık olduğundan kolaylıkla hata yapılabiliniyor ve anlaşılması zordur. Bu nedenle geliştirilen yazılıma özgü bir Modelleme Dili(*Modelling Language*) oluşturulmuştur. Bu dil sayesinde kullanıcıya yakın(*user friendly*) bir giriş dosyası biçimi sağlanmıştır.

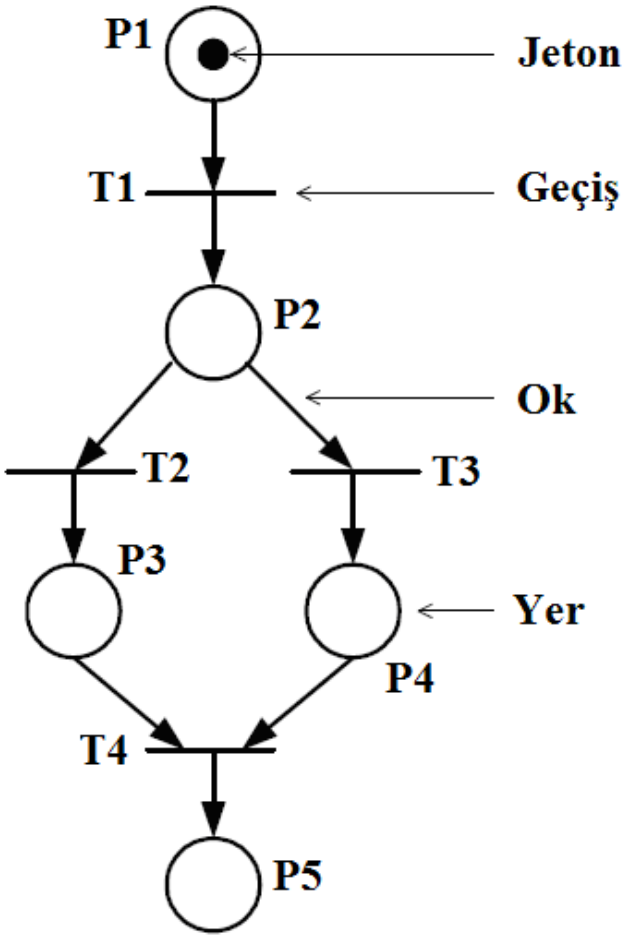
1.1 KLASİK PETRİ AĞLARI

Klasik Petri Ağları ayrık olay sistemlerinin modellenmesinde, analizinde ve yönetilmesinde kullanılan grafiksel ve matematiksel bir araçtır. Birçok uygulama alanı bulunmaktadır. Petri ağları kullanılarak ayrık olaylı ve eş zamanlı her alandan sistemler modellenebilir. Modellenen sistemler arasında eşzamanlı ve asenkron devreler, haberleşme sistemleri, haberleşme protokolleri, bilgi sistemleri, sebep sonuç ilişkisine dayanan sistemler, çok işlemcili sistemler, iş akışları, biyolojik sistemler, kimyasal reaksiyonlar, çevrebilimle ilgili sistemler, rastlantısal zamanlı sistemler, otomatik üretim sistemleri, gerçek zamanlı sistemler gösterilebilir. [15]

Petri ağları sistemlerdeki nedensel ilişkileri, yarış durumlarını ve seçim durumlarını kolay ve doğru modelleyip çeşitli analizler yapılmasını mümkün kılar. Bu analizler arasında canlılık, sınırlılık, güvenlik, tutarlılık vardır. Kilitlenmesiz bir elektronik sistem tasarlanmak istendiğinde bu durumda modellenen sisteme ait Petri Ağı' nın her durumda canlı kalması gerekir, bazı durumlarda canlılığın bozulması durumunda model üzerinde gerekli müdahaleler yapılarak sistemin her zaman canlı kalması dolayısıyla kilitlenmesiz olması sağlanabilir. Petri ağı değişmezleri sistemlerin tasarımında bu tarz yapısal özelliklerini ortaya çıkarmak amacıyla kullanılabilir. [15]

Petri Ağların temel elemanları yerler (*places*), geçişler (*transitions*), oklar (*arcs*) ve jetonlar(*tokens*)dır. Yerler durumları ifade ederken geçişlerde olayları ifade etmektedir. Oklar, durumlar ve olaylar arası ilişkiyi ifade eder. Ateşlenme koşulu sağlandığında yani geçişlerin giriş yerlerinde gereken sayıdan daha fazla jeton bulunduğunda geçiş, çıkış yerlerine yeni jetonlar üretir. Giriş yerlerinden jetonların eksilmesi ve çıkış yerlerine jetonların eklenmesi aynı anda olmaktadır. Bu ateşlenme tipine atomik ateşlenme (*atomik firing*) denilmektedir.

Şekil 1.1 de basit bir Petri Ağı örneği gösterilmiştir. Bu ağda tüm ok ağırlıkları 1'dir. Eğer ok ağırlığı 1 ise ok üzerine değer yazılmasına gerek yoktur. Jeton bulunduran yerlere işaretli yer denir. Geçişe doğru yönlendirilmiş okların başlangıç yerindeki yerlere giriş yerleri denir. Geçişten yerlere doğru yönlendirilmiş okların gösterdiği yerlere çıkış yerleri denir.



Şekil 1.1 : Basit Bir Petri Ağı

Klasik Petri Ağı PN = (P, T, I, O, W, M_0) gibi tanımlanır. P: Ağdaki yerlerin kümesi, T: Ağdaki geçişlerin kümesi, I: Yerlerden Geçişlere yönelen okların kümesi, O: Geçişlerden yerlere yönelen okların kümesi. M_0 ilk işaretleme(*initial marking*) matrisidir. Şekil 1.1 deki örnek üzerinden açıklarsak

P = { P1, P2, P3, P4, P5 }

T = { T1, T2, T3, T4 }

I = { (P1,T1), (P2,T2), (P2,T3), (P3,T4), (P4,T4) }

O = { (T1,P2), (T2,P3), (T3,P4), (T4,P5) }

W(P1,T1) = W(P2,T2) = W(P2,T3) = W(P3,T4) = W(P4,T4) = W(T1,P2) = W(T2,P3) = W(T3,P4) = W(T4,P5) = 1

M_0= [1 0 0 0 0] ^{T}

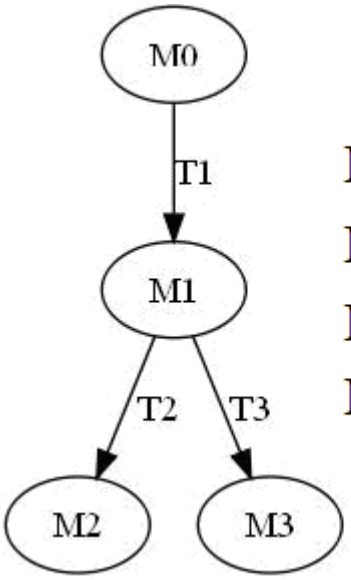
Klasik Petri Ağlarında ateşlenme şartı şu şekildedir. Bir geçişin giriş yerlerinin hepsinde ayrı ayrı en az o yerden geçişe yönlendirilmiş okun ağırlığı kadar jeton varsa bu geçiş ateşlenebilirdir ve hemen giriş yerlerinden jetonlar eksiltilerek çıkış yerlerine çıkış oklarındaki ağırlık kadar jetonlar eklenir.

Petri Ağının ilk işaretlemesinden sonra yapılan ardışık ateşlemeler sonucunda gidilen yeni işaretlemeye gidene kadar ateşlenen geçişlerin oluşturduğu dizeye ateşlenme dizisi denir. $M_1 \ldots M_{n+1}$ işaretlemeleri için ateşleme dizisi gösterimi:

$M_i\ [t_i > M_{i+1}\ \forall\ i=1 \ldots n$ şeklindedir.

Ateşleme dizisi $\sigma = \{T_1, T_2, T_3 \ldots T_n\}$ için M_i işaretlemesinden elde edilen en son işaretleme kısacası $M_i\ [\sigma > M_{n+1}$ ile gösterilir. Başlangıç durumundan erişilebilir tüm işaretlemeler kümesi $R(M_0)$ ile gösterilir.

Erişilebilir işaretlemelerin bir bütün olarak oklarla birbiriyle olan ilişkilerinin gösterildiği grafiğe Erişilebilirlik Grafiği denir. Eğer tüm ilişkiler akış olarak gösteriliyorsa yani aynı işaretlemeler de gösteriliyorsa Erişilebilirlik Ağacı denir.



$\mathbf{M_0 = [\ 1\ 0\ 0\ 0\ 0\ ]^T}$

$\mathbf{M_1 = [\ 0\ 1\ 0\ 0\ 0\ ]^T}$

$\mathbf{M_2 = [\ 0\ 0\ 1\ 0\ 0\ ]^T}$

$\mathbf{M_3 = [\ 0\ 0\ 0\ 1\ 0\ ]^T}$

Şekil 1.2 : Basit Bir Petri Ağı Örneğinin Erişebilirlik Grafiği

Şekil 1.1 de verilen Basit Petri Ağı'nda M_0 işaretlemesinden M_2 işaretlemesine ulaşabilmek için ateşleme dizisi $\sigma_1 = \{T_1, T_2\}$ 'dir. Kısaca $M_0\ [\sigma_1 > M_2$ şeklinde gösterilir. M_0 işaretlemesinden M_3 işaretlemesine ulaşabilmek için ateşleme dizisi $\sigma_2 = \{T_1, T_3\}$ 'dür. Kısaca $M_0\ [\sigma_2 > M_3$ şeklinde gösterilir.

Kompleks sistemleri kolaylıkla açıklamak için sıradan Petri net yapılarına birkaç ekleme yapılmıştır. Bunlar: Ağırlıklı oklu Petri net, Yasaklama oklu Petri net, Yetkileme oklu Petri net, Sonlu kapasiteli Petri net gibidir. [16]

1.1.1 Petri Ağlarına İlişkin Tanımlar

1.1.1.1 Sınırlılık (*Boundedness*)

Bir Petri Ağında her hangi bir yerde bulunabilecek maksimum jeton sayısı şeklinde ağa sınırlama getirilirse bu ağa k-sınırlı ağ denir. Bir ağın k-sınırlı olması, o ağın yerlerinde k'dan daha fazla jeton bulunamayacağı anlamına gelir.

Sınırlı bir Petri ağı donanımsal olarak gerçekleştirilebilir iken, sınırsız bir Petri ağı donanımsal olarak gerçekleştirilemezdir. Buna karşın sınırsız olan her Petri ağı yardımcı yerler yardımıyla sınırlı hale getirilebilir. [15]

1.1.1.2 Güvenlik (*Safeness*)

Gerçek donanımları modellemeyi amaçlayan bir Petri ağında, en önemli özelliklerden biri güvenliktir. Bir Petri ağındaki herhangi bir işaretlemede bir yerde bulunabilecek jeton sayısı biri aşmıyorsa o yer güvenlidir. Sadece güvenli yerler barındıran bir Petri ağına güvenli denir.[15] Güvenlik, sınırlılık özelliğinin özel bir halidir. 1-sınırlı Petri Ağına Güvenli Petri Ağı denir.

1.1.1.3 Tutarlılık (Conservation)

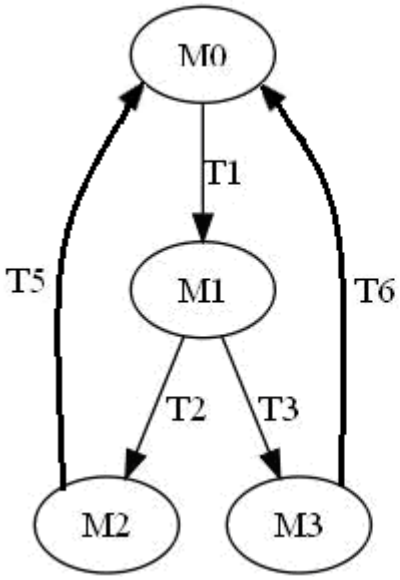
Petri Ağlarında geçişler aktif elemanlardır. Yani olayları temsil ederler. Sisteme yeni jeton katabilme ya da sistemden jeton eksiltebilme özellikleri vardır. Yani sistemde toplam jeton sayısı değişkenlik gösterebilir. Eğer sistemdeki toplam jeton sayısı değişmeyip sabit kalıyorsa bu ağa tutarlı ağ denir. Bir ağın tutarlı olabilmesi için her geçişin giriş yerlerindeki ok ağırlıklarının toplamı çıkış yerlerindeki ok ağırlıklarının toplamına eşit olmalıdır. Böylece ağ tutarlı kalacaktır.

1.1.1.4 Canlılık (Liveness)

Kilitlenmeler(*deadlock*), kaynak dağılımından kaynaklanabilecek problemlerdir. Ağ, başlangıç işaretlemesi olan M_0'dan ateşleme dizisi sonucunda daha ateşlenemiyorsa sistem kilitlenmiş demektir ve varılan işaretleme ölü işaretlemedir (*dead marking*).

Şekil 1.1 de verilen basit Petri ağı iki tane ölü işaretlemeye ulaşmaktadır. Ölü işaretlemeler olan M2 ve M3 işaretlemelerine sırasıyla M_0 [σ_1 > M_2 ve M_0 [σ_2 > M_3 yollarından ulaşılır. σ_1 = {T_1, T_2} σ_2 = {T_1, T_3}' dir. Görüldüğü gibi T_1 geçişi ateşlendikten sonra T_2 ya da T_3 geçişi ateşlenebilmekte ve sonra sistemde ateşlenebilir geçiş kalmamaktadır.

Eğer bu sistemi canlı yapmak istiyorsak P3 yerinden P1 yerine doğru ok ağırlığı bir olan yol ve P4 yerinden P1 yerine doğru ok ağırlığı 1 olan yol çizilmelidir. Böylece sistem sonsuza kadar ateşlenebilir kalacaktır. Bu ek geçiş koyularak yapılan erişilebilirlik analizinde daha önceden ölü işaretleme olan M_2 ve M_3 artık eski işaretleme(*old marking*) olarak adlandırılır. Çünkü bu işaretlemeden yapılan ateşlenmeler sonucunda daha önce ağda bulunan M_0 işaretlemesine ulaşılmaktadır. Değiştirilmiş ağa ait erişilebilirlik grafiği Şekil 1.3 de gösterilmiştir.



$$M_0 = [\,1\,0\,0\,0\,0\,]^T$$
$$M_1 = [\,0\,1\,0\,0\,0\,]^T$$
$$M_2 = [\,0\,0\,1\,0\,0\,]^T$$
$$M_3 = [\,0\,0\,0\,1\,0\,]^T$$

Şekil 1.3 : Sonsuz Ateşlenmeli Basit Petri Ağı Örneğinin Erişebilirlik Grafiği

1.2 ZAMANLI PETRİ AĞLARI

Klasik Petri Ağları ayrık olaylı sistemleri davranışsal olarak incelemeye olanak vermektedir. Yapılan analizlerde sistemin hangi durumlara varacağı ve muhtemel kilitlenme senaryoları görülebilinmektedir. Böylece hazırlanan modelin kusurları görülebilinir. Ancak gerçek zamanlı sistemlerde bu düzeyde inceleme yetersiz kalmaktadır. Bir sistemin performansı sadece o işi yapabilmesi değil işi yapabileceği optimum sürede yapması önemlidir. Bu nedenle modeller incelenirken zamanın

değişimi de göz önüne alınmalıdır. Aynı işi bir model daha kısa zamanda yapabiliyorsa uzun sürede yapan modelin kullanılma ihtiyacı tartışılmalıdır.

Zamanın Petri Ağlarına nasıl katılacağı konusunda değişik yaklaşımlar vardır. Zaman bilgisinin sistemde nerelere atanacağı hatta zamanın nasıl değişeceği soruları sorulduğunda değişik yaklaşımlar bulunmuştur.

Zamanın sisteme nasıl katılacağı konusunda dört tip yaklaşım vardır.

- Zamanın geçişlere atanması
- Zamanın yerlere atanması
- Zamanın oklara atanması
- Zamanın jetonlara atanması (*Zamanlı Renkli Petri Ağlarında anlatılacak*)

Zamanın geçişlere atanması yaklaşımı daha çok benimsenmiş olmasına rağmen diğer metotlarda kullanılmaktadır.

Zamanın nasıl değişeceği konusunda da üç tip yaklaşım bulunmaktadır.

- Zamanın belirli olarak değişmesi
- Zamanın rastlantısal olarak değişmesi
- Zamanın aralıklı olarak değişmesi

Bu iki gruplama düşünülerekten Zamanlı Petri Ağları birçok başlığa ayrılabilir. Bunlar şu şekildedir.

- Zamanlı Petri Ağları (*Timed Petri Nets*)
    - Belirli Zamanlı Petri Ağları (*Deterministic Timed Petri Nets*)
        - Geçişleri Zamanlandırılmış Belirli Zamanlı Petri Ağları
        - Yerleri Zamanlandırılmış Belirli Zamanlı Petri Ağları
        - Okları Zamanlandırılmış Belirli Zamanlı Petri Ağları
        - Jetonları Zamanlandırılmış Belirli Zamanlı Petri Ağları
    - Rastlantısal Zamanlı Petri Ağları (*Stochastic Timed Petri Nets*)
        - Rastlantısal Petri Ağları
        - Genelleştirilmiş Rastlantısal Petri Ağları

- Yüksek Seviyeli Rastlantısal Petri Ağları
    - Renkli Rastlantısal Petri Ağları
    - Rastlantısal Yüksek Seviyeli Petri Ağları
- Markovian Rastlantısal Petri Ağları
    - Genişletilmiş Rastlantısal Petri Ağları
    - Belirli-Rastlantısal Petri Ağları

Belirli ve Rastlantısal Zamanlı Petri Ağları ayrık olay sistemlerinin verim hesabında(*performance evaluation*) geniş bir uygulama alanına sahiptir. Özellikle, Belirli Zamanlı Petri Ağları, sistemin çevrim süresi(*cycle time*) bulmada, sistemin tıkanma noktalarını(*bottleneck workstation*) bulmada, sistem için beklenen zaman sınırlamalarına uyup uymadığı incelemede yararlıdır. Rastlantısal Petri Ağları, oran(*rates*), üretim miktarı (*throughtput*), ortalama gecikme(*average delays*), kritik kaynak kullanımı(*critical resource utilization*), sistem güvenilirliğini ölçme(*reliability measures*) araştırmalarında yararlıdır. [17]

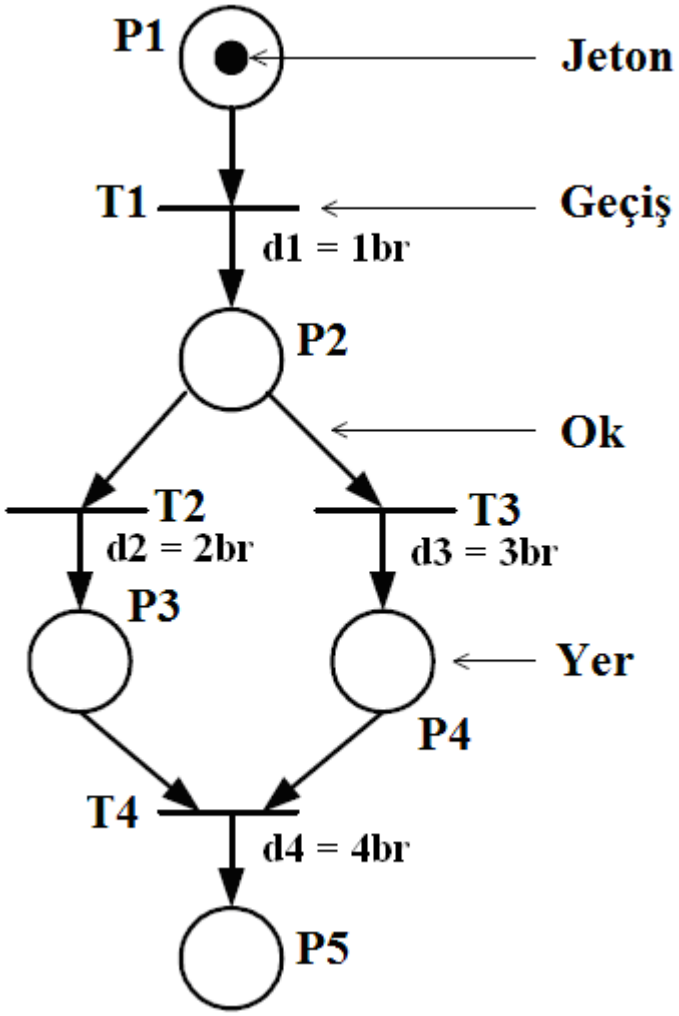
Zamanlı Petri Ağlarının üç tip bellek özelliği vardır. Bunlar;

- Yeniden Örnekleme(*Resampling*)
- Mümkün Olma Belleği(*Enabling Age Memory*)
- Bellek Yaşı Modeli(*Age Memory*)

İki geçişin(T_1, T_2) ortak giriş yerindeki jeton miktarı her iki geçişi aynı anda ateşleyemiyor fakat ayrı ayrı geçişlerden birini ateşleyebiliyor olsun. Geçişlerin ateşlenme gecikmeleri arasında T1 > T2 olsun. Geçişler izinli olduktan T2 süre sonra T_2 geçişi ateşlenir ve jetonu kullanılır. Tekrar giriş yerine yeni bir jeton geldiğinde her iki geçişte ateşlenme sürecine T1 ve T2 değerleriyle yeniden yüklenerek başlıyorsa bu bellek tipine yeniden örnekleme belleği denir. Eğer yarışı kaybeden geçiş ulaşılan yeni işaretlemede izinli olabiliyorsa T1-T2 değerinden devam eder, izinli olamıyorsa değeri sıfırlanır. Bu bellek tipine mümkün olma belleği denir. Bellek yaşında ise kalan değer kullanılana kadar saklanır.

1.2.1 Belirli Zamanlı Petri Ağları

Geçişleri Zamanlandırılmış Petri Ağlarında ilk başlarda olayın gerçekleştiği süre olan ateşlenme gecikmesi(*firing delay*) eklenmiştir. Geçiş, izinli(*enabled*) olduktan sonra girişlerden ok ağırlıkları kadar jetonlar düşülür ve gecikme süresi geriye doğru saymaya başlar. Süre bitiminde çıkış oklarındaki ok ağırlıkları kadar jeton çıkış yerlerine eklenir.



Şekil 1.4 : Belirli Geçişleri Zamanlı Basit Bir Petri Ağı

Belirli Geçişleri Zamanlı Petri Ağı PN = (P, T, I, O, W, M_0, D) gibi tanımlanır. P: Ağdaki yerlerin kümesi, T: Ağdaki geçişlerin kümesi, I: Yerlerden Geçişlere yönelen okların kümesi, O: Geçişlerden yerlere yönelen okların kümesi. M_0 ilk işaretleme(*marking*) matrisidir. Klasik Petri Ağları' nından farklı olarak geçişlerin ateşlenme gecikmelerin tutan D matrisi eklenmiştir. Şekil 1.4 deki örnek üzerinden açıklarsak

P = { P1, P2, P3, P4, P5 }, T = { T1, T2, T3, T4 }

I = { (P1,T1), (P2,T2), (P2,T3), (P3,T4), (P4,T4) }

O = { (T1,P2), (T2,P3), (T3,P4), (T4,P5) }

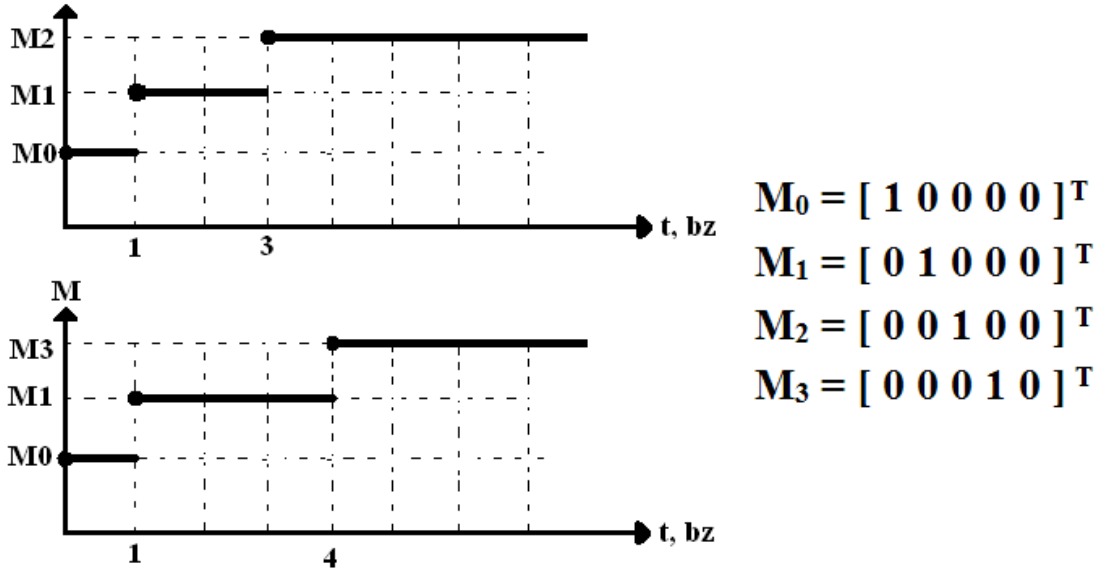
$D = [\ 1\ 2\ 3\ 4\ ]^T$

W(P1,T1) = W(P2,T2) = W(P2,T3) = W(P3,T4) = W(P4,T4) = W(T1,P2) = W(T2,P3) = W(T3,P4) = W(T4,P5) = 1

$M_0 = [\ 1\ 0\ 0\ 0\ 0\ ]^T$

Sistem t = 0 anında çalışmaya başladığında T1 geçişi hemen izinli olacaktır. Bu nedenle bu geçiş ateşlenme sürecine başlayacaktır. P1 yerindeki jeton sistemden kaybolacaktır ve geçişte bekleyecektir. T1 geçişinin ateşlenme gecikmesi olan d1 = 1br zaman dolana kadar sistemde herhangi bir değişim olmayacaktır. Çünkü T1 olayı devam etmektedir ve hala sonuç üretmiş değildir. T1 olayı gerçeklendiğinde yani t = 1 anında T1 geçişi P2 çıkış yerine jeton üretecektir.

Bu şekilde sistem incelendiğinde sistemin zamana bağlı işaretleme değişim grafiği Şekil 1.5 deki gibi olacaktır.



Şekil 1.5 : Belirli Geçişleri Zamanlı Basit Bir Petri Ağı Zaman Diyagramları

Görüldüğü gibi Petri Ağları incelenirken çelişki durumuyla karşılaşılabilinir. Çelişki durumunun oluşmasının sebebi modellemenin iyi yapılmamasından kaynaklanmaktadır. Çelişki durumu geçişler arasında ortak giriş yerlerinin bulunması durumunda görülmektedir. Eğer ortak giriş yerlerinde her iki geçişi ateşleyebilecek

kadar jeton bulunmayıp ayrı ayrı ateşleyebilecek kadar jeton varsa bu jetonun hangi geçiş tarafından kullanılacağı problemiyle karşılaşılır. Bu tasarım hatasına rağmen sistem analizinin devam edilebilmesi için öncelik ataması yapılır.

Verilen örnekte çelişki durumu olduğundan iki farklı zaman diyagramı oluşmuştur. Eğer T2 geçişi T3 geçişinden daha öncelikli seçilirse birinci zaman diyagramı elde edilecektir. Eğer tersi söz konusuysa ikinci zaman diyagramı elde edilecektir. Çelişki durumlarında ağ için tanımlanmış öncelik atama fonksiyonuna bakılır. Öncelik derecesi değeri düşük olan geçiş daha önceliklidir.

Yukarıda tanımlanan modelde geçiş izinli olur olmaz ateşlenme zamanına başladığı kabulü yapılmıştır. Gerçekte ise herhangi bir geçişin ateşlenebilirlik koşulunun sağlandığı an ile ateşlenme zamanı başladığı an arasında bir süre vardır. Eğer bu e_t izinli olma süresini(*enabled time*) modele tanımlamazsak; geçişin çıkış yerindeki jetonlar çıkış yerine eklendiği anda sonraki geçişi ateşlediğinde hemen kaybolacaktır. Bu yerde jetonlar anlık olarak gözükecektir. Bu durumu engellemek için bir izinli olma gecikme süresi konulur. Gerçek Dünya düşünüldüğünde bu kısa olan sürenin eklenmesi mantıklıdır. Çünkü bir durumdayken bir olaya başlamak için durumun algılanması süresi vardır. Örneğin, üretim bandında üretimin başlatılmasıyla üretim bandının çalışmaya başlaması arasında küçükte olsa gecikme vardır.

Ayrıca, bir sistemin ard arda çalıştırılması arasında da bir gecikme süresi vardır. Yani işin tekrar yapılmaya başlanması için dinlenme zamanı olarak da adlandırılabilecek bekleme süresi(*waiting time*) vardır.

Tanımlanan yeni parametreler örneğe eklenerek yeniden incelenirse:

Belirli Geçişleri Zamanlı Petri Ağı PN = (P, T, I, O, W, M_0, D, P_r) gibi tanımlanır.

P = { P1, P2, P3, P4, P5 }, T = { T1, T2, T3, T4 }

I = { (P1,T1), (P2,T2), (P2,T3), (P3,T4), (P4,T4) }

O = { (T1,P2), (T2,P3), (T3,P4), (T4,P5) }

D = [d1 d2 d3 d4], d = (e_t f_t w_t)

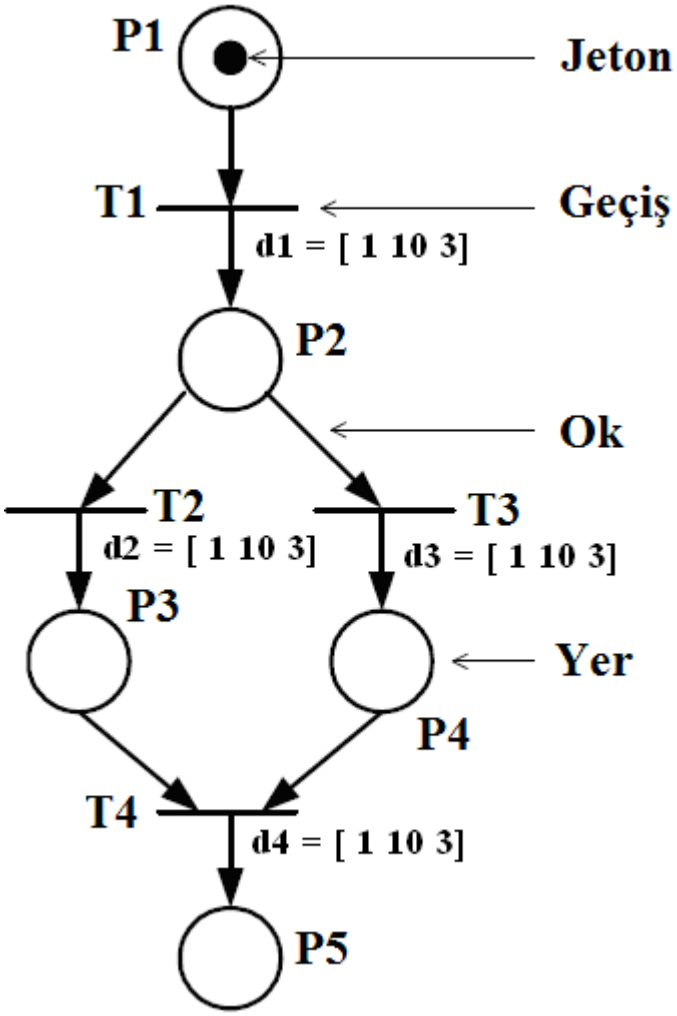
d1 = [1 10 3] , d2 = [1 10 3] ,

d3 = [1 10 3] , d4 = [1 10 3]

P_r = [0 1 2 3] şeklinde T1(0) T2(1) T3(2) T4(3) yani T1 > T2 > T3 > T4

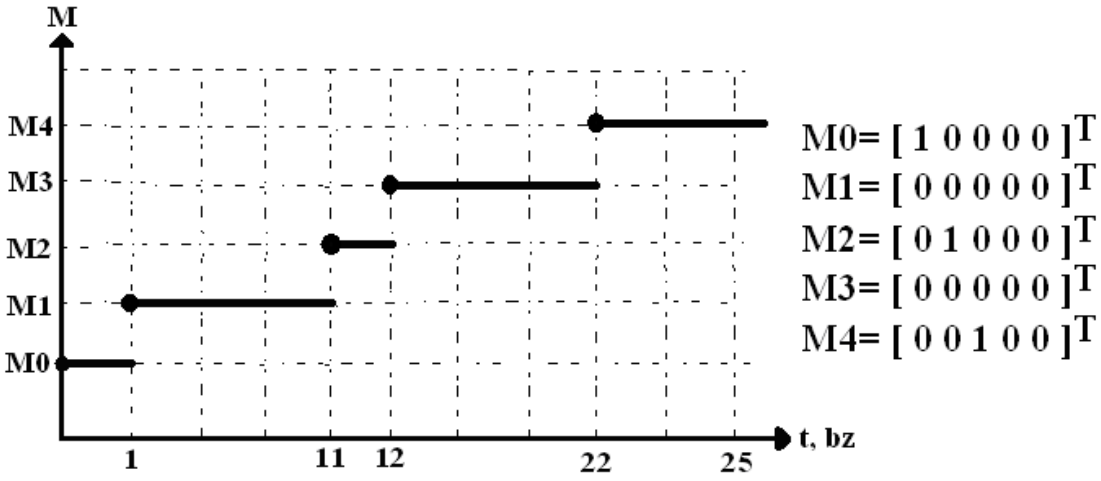
W(P1,T1) = W(P2,T2) = W(P2,T3) = W(P3,T4) = W(P4,T4) = W(T1,P2) = W(T2,P3) = W(T3,P4) = W(T4,P5) = 1

M_0= [1 0 0 0 0] ^{T}



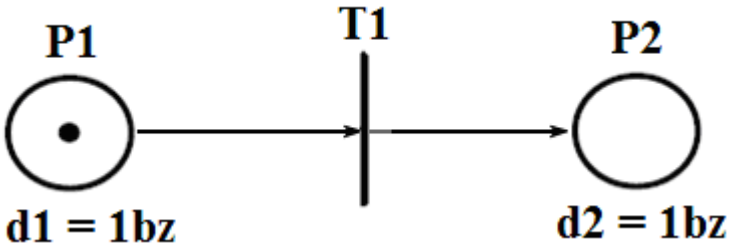
Şekil 1.6 : Belirli Geçişleri e_t, f_t, w_t Zamanlı Basit Bir Petri Ağı

Ağın zaman analizi yapıldığında sistemin zamana bağlı işaretleme değişim grafiği Şekil 1.7 deki gibi olacaktır.



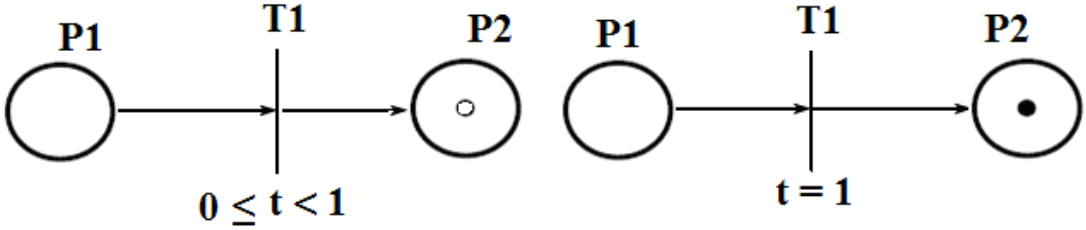
Şekil 1.7 : Belirli Geçişleri e_t, f_t, w_t Zamanlı Basit Petri Ağı Zaman Diyagramı

Yerleri Zamanlanmış Petri Ağlarında gecikmeler geçişlere değil yerlere tanımlanmıştır. Geçişlerin ateşlenmesiyle çıkış yerlerine gelen jetonlar elverişsiz jetondur. Yani var ama kullanılabilir değildir. O yer için atanan zaman gecikmesi kadar sonra jeton izinli olur. Böylece jeton artık kullanılabilirdir. Bu süreye Bowden çalışmasında bu süreyi tutma süresi (*holding time*) olarak adlandırmıştır. [12]



Şekil 1.8 : Belirli Yerleri Zamanlı Basit Petri Ağı

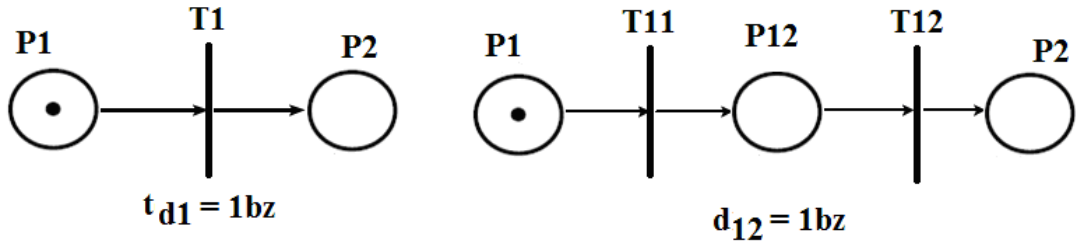
Şekil 1.8 deki Yerleri Zamanlanmış Petri Ağı incelendiğinde t = 0 anında P1 yerindeki jeton sistemden kaybolur ve hemen P2 yerine elverişsiz olarak geçer. P2 yerinin tutma süresi d2 zaman sonra jeton P2 yerinde elverişli olur ve sonraki geçişleri tetiklemek için kullanılabilir olur.



Şekil 1.9 : Belirli Yerleri Zamanlı Basit Petri Ağı Analizi

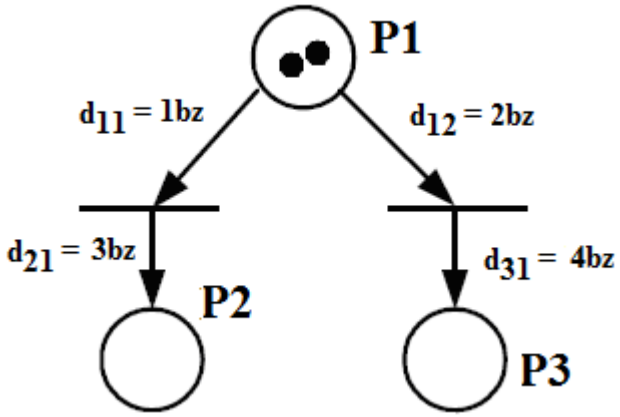
Geçişleri zamanlanmış Petri Ağlarından, Yerleri Zamanlanmış Petri Ağlarına geçiş yapılabilinir. Bu geçişi yapabilmek için Geçişleri Zamanlı Petri Ağındaki jetonun geçişte beklemesini sürecini ayrı bir durum olarak düşünmek gerekecektir. Bu durumda jeton geçişte değil, zaman ataması yapılmış ağa eklenen yeni yerde bekleyecektir ve süre dolduğunda anlık olarak geçiş gerçekleşecektir.

Şekil 1.9 deki ağlar arası geçişe ait örnek incelendiğinde T1 geçişi 1bz gecikmesine sahip olduğundan P12 yeri 1bz tutma süresine sahiptir. P1 yerindeki jeton sayesinde anlık olarak ateşlenen T11 geçişinden gelen jeton P12 yerinde elverişsiz olarak 1bz bekledikten sonra tekrar anlık olarak ateşlenecek olan T12 geçişi sayesinde P2 yerine geçecektir.



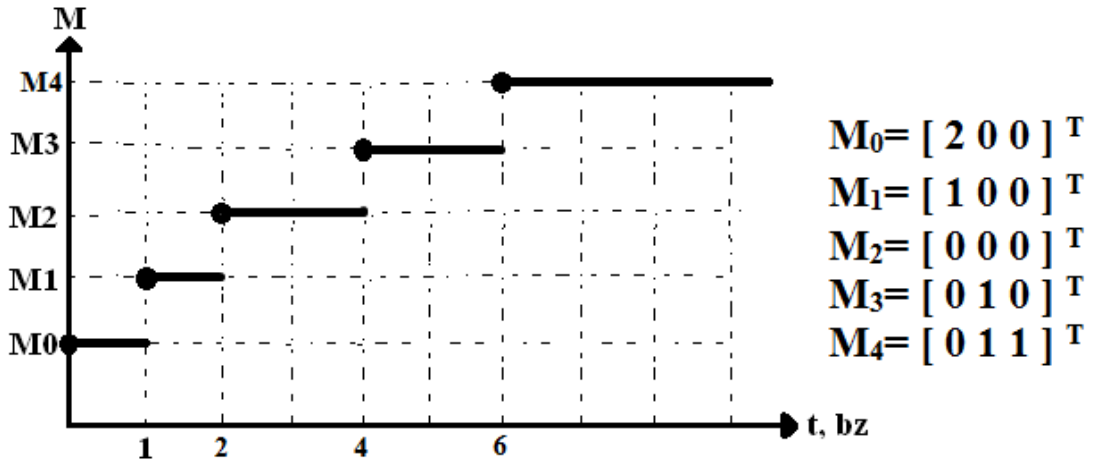
Şekil 1.10 : Geçişleri ve Yerleri Zamanlı Petri Ağları Arasında Geçiş

Okları Zamanlandırılmış Petri Ağlarında ise geçişlerin giriş ve çıkış oklarına ayrı ayrı zaman gecikmeleri atanmaktadır. Böylece sistem tasarımda daha büyük esneklik sağlanabilir.



Şekil 1.11 : Okları Zamanlı Basit Petri Ağı

Ağın zaman analizi yapıldığında sistemin zamana bağlı işaretleme değişim grafiği Şekil 1.12 deki gibi olacaktır.



Şekil 1.12 : Okları Zamanlı Basit Petri Ağı Zaman Diyagramı

1.2.2 Rastlantısal Zamanlı Petri Ağları

Rastlantısal olarak zamanlandırılmış Petri ağları, Klasik Petri Ağı yaklaşımına, geçişlere rastgele ateşlenme zamanları atanarak elde edilmiş modeldir [17]. Bu model aslında rastgele bir işlem yürüten yüksek seviyeli bir Petri ağı modelidir. Rastgele Zamanlı Petri Ağları, sistemlerin performans değerlendirmesinde daha doğru ve genel sonuçlar elde edilmesine yardımcı olur. Sistemin parametrelerine karşı ne kadar hassas olduğunun anlaşılmasında yararlı olur. Rastlantısal zaman değişimi gerçek hayata daha uygundur.

Klasik Petri Ağlarında geçişlerin ateşlenme sürelerine rastlantısal değişen zaman değerleri atanarak elde edilmiştir. Bu rastlantısal zaman olasılık yoğunluk fonksiyonu(*pdf*) ya da olasılık dağılım fonksiyon olarak seçilmektedir. Genellikle de belleksizlik özelliğinden ötürü negatif eksponansiyel olasılık dağılım fonksiyonu kullanılmaktadır. Çünkü negatif eksponansiyel pdf belleksizlik özelliğine sahip olan tek sürekli pdf'dir.

Belleksizlik özelliği şu şekilde tanımlanır: Sistemin şu anki işaretlemede bulunma süresi (*sojourn time*) ve gidilecek bir sonraki işaretleme, sistemin geçmişinde şu anki işaretlemede bulunulan *süreye* bağlı değildir, sadece ve sadece şu anki *işaretlemeye* bağlıdır. Örnek olarak, M0➔M1 işaretlemeleri arası geçiş düşünülürse, M0 da kalma süresi ve M0 dan M1 e geçişin gerçekleşmesi, M0 da geçmişte kalınan süreye bağlı değildir. M0➔M1 geçişinin gerçekleşmesi sadece M0 işaretlemesine bağlıdır.

Olasılık teorisinde ve istatistikte negatif eksponansiyel dağılım(*negative exponential distributions*) sürekli olasılık dağılımının bir alt sınıfıdır. Poisson sürecinde, olaylar

arasındaki süreyi tanımlar. Eksponansiyel dağılımın olasılık yoğunluk fonksiyonu (*pdf*) Şekil 1.13 deki gibidir. [18]

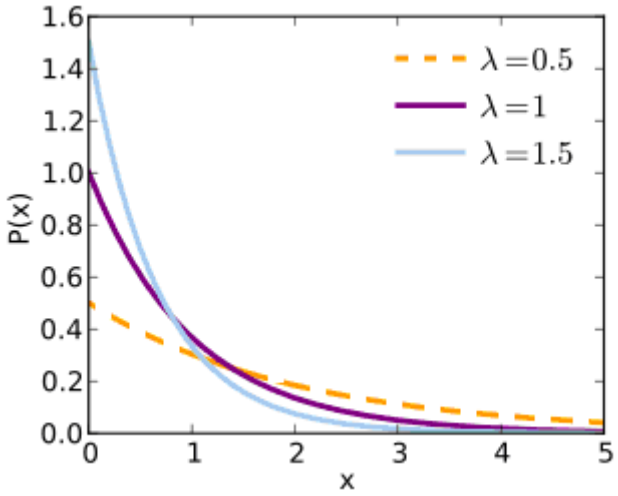
$$P(x;\lambda) = \begin{cases} \lambda e^{-\lambda x}, & x \geq 0, \\ 0, & x < 0. \end{cases}$$

Şekil 1.13 : Eksponansiyel Dağılımın Olasılık Yoğunluk Fonksiyonu(*pdf*)

Burada λ parametresi oran(*rate*) ya da varış hızı (*arrival rate*) olarak adlandırılır. Pozitif sınırlı bir değerdir. β varış zamanı (*arrival time*) parametresi oran parametresinin tersidir ve zamana karşılık gelir.

$$\beta = \frac{1}{\lambda}$$

Şekil 1.14 : Varış zamanı ile oran arasındaki ilişki

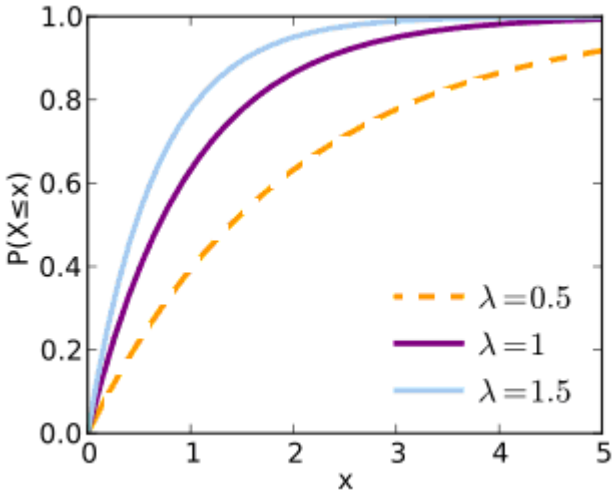


Şekil 1.15 : Eksponansiyel Dağılımın Olasılık Yoğunluk Fonksiyonu Grafiği

Eksponansiyel dağılımın birikimli *(kümülatif)* dağılım fonksiyonu (*cumulative distribution function, cdf*) Şekil 1.16 daki gibidir. [18]

$$P(x;\lambda) = \begin{cases} 1 - e^{-\lambda x}, & x \geq 0, \\ 0, & x < 0. \end{cases}$$

Şekil 1.16 : Eksponansiyel Dağılımın Birikimli Dağılım Fonksiyonun(*cdf*)

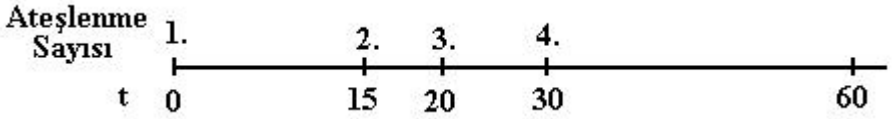


Şekil 1.17 : Eksponansiyel Dağılımın Birikimli Dağılım Fonksiyonun Grafiği

Kümülatif Olasılık dağılım fonksiyonu [0:1] aralığına normalize edildiğinden analizde tercih edilecektir.

Geçişlerin ateşlenme sürelerine rastlantısal zaman değeri üretmek için öncelikle rastgele üretilmiş bir olasılık değerine ihtiyaç vardır. Rastgele seçilen bu değer ile birlikte λ oran parametresi de kullanılaraktan birikimli dağılım fonksiyonunda yapılacak matematiksel işlemle birlikte geçiş için kullanılacak ateşlenme gecikmesi hesaplanır.

Bir örnek üzerinde açıklarsak t = 0 anından itibaren bir geçişin 4 kere ateşlendiğini ve 1. ateşlemenin 15sn, 2. ateşlenmenin 5sn, 3. ateşlenmenin 10sn 4.ateşlenmenin 30sn olduğunu düşünürsek bu geçişe ait ortalama ateşlenme tekrar süresi yani varış hızı β = (15+5+10+30) / 4 = 15sn olarak bulunur. λ oran değeri ise λ = 1/ β = 0.066 olarak bulunur. Bulunan bu değer ve rastlantısal seçilen olasılık değeri tekrar cdf kullanılarak sonraki ateşlenmenin ateşlenme süresi bulunur.



Şekil 1.18 : Oran ve Varış Hızı Hesaplama Örneği

Seksenli yıllarda Natkin ve Molloy Rastlantısal Petri Ağlarında işaretleme sürecinin sürekli zamanlı Markov Zinciri(*Markov Chain*) olduğunu göstermişlerdir.[19]

Markov Zinciri sayesinde geçişlere atanan oran değerleri sayesinde sistemin kararlı durumlarına ait olasılık değeri bulunmaktadır. Sistemin kararlı durumlarına ait

olasılık matrisi(π) ile sistemin geçiş oran matrisi(Q) arasında aşağıdaki gibi ilişki vardır.

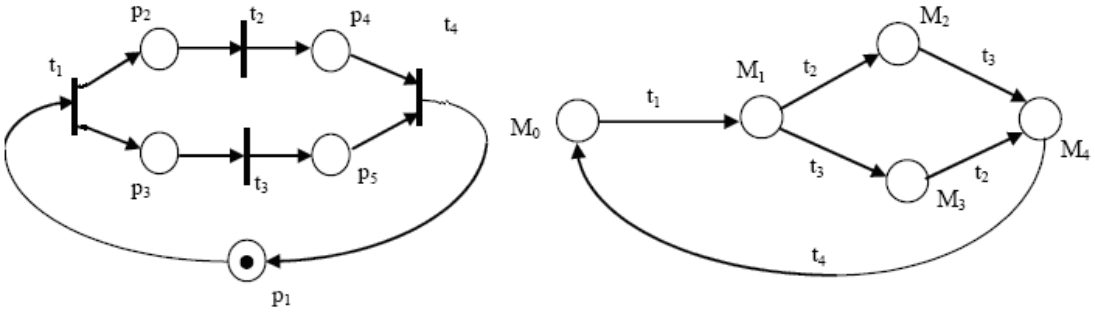
$$\pi \cdot Q = 0 \;;\; \sum_i \pi_i = 1$$

Markov Süreci(*Markov Process*), Markovian özeliği olan belleksizlik özelliğini sağlar. Belleksizlik özelliği gerçek dünyaya uygun olmasa da ayrık olaylı sistemlerin rastlantısal modelini kurmada Markov Süreci'nin geniş kullanım alanı vardır. Markov süreci, ayrık olaylı sistemlerde Markov Zinciri olarak adlandırılır.

Markov Zinciri sayesinde sistemin ulaşacağı kararlı durumlarda bulunma olasılıkları bulunur. Böylece sistemin performansı, güvenilirliği hakkında yorum yapılabilir.

Şekil 1.19 da verilen Rastlantısal Petri Ağında geçişlerin oran değeri λ1=λ2=λ3=λ4=1 olsun.[19] Sistemin kararlı işaretlemelerinin olasılıkları toplamı birdir. $\pi_0+\pi_1+\pi_2+\pi_3+\pi_4+\pi_5=1$ sistemin Markov analizi yapıldığında kararlı işaretlemelerin olasılık dağılımları aşağıdaki gibi olacaktır.

$\pi_0 = \pi_4 = 2/7$; $\pi_1 = \pi_2 = \pi_3 = 1/7$ bulunur.



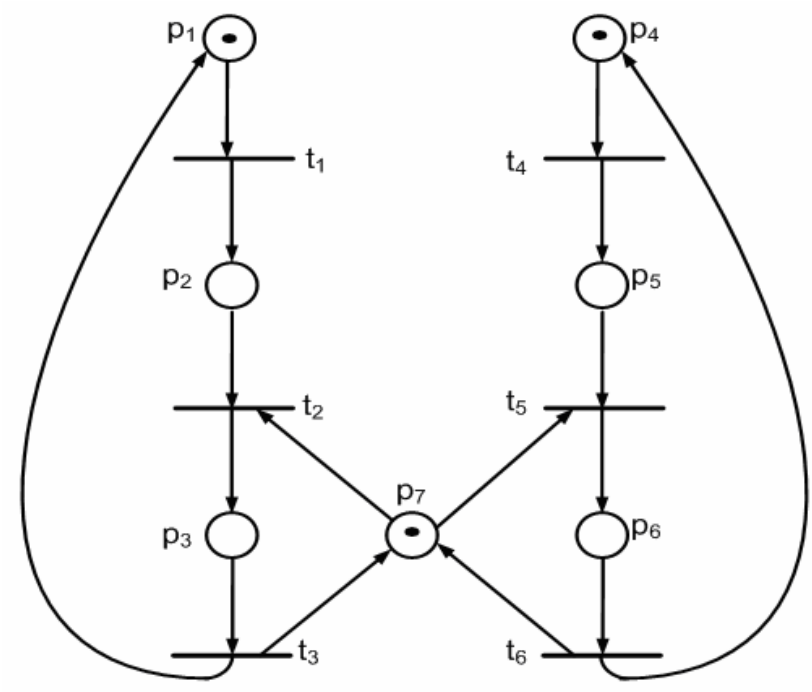
Şekil 1.19 : Rastlantısal Bir Petri Ağının Markov Zinciri Analizi Örneği

Rastlantısal Zamanlı Petri Ağlarında atomik ateşlenmeli geçişlerinde olduğu Genelleştirilmiş Rastlantısal Petri Ağları da vardır. Bu geçişler ani geçiş (*immediate transition*) denir. Zaman bilgisi içeren geçişlere zamanlı geçiş(*timed transition*) denir. Bu tip ağlarda eğer bir işaretlemede atomik ateşlenme varsa bu işaretleme sönümlü işaretleme(*vanishing marking*) olarak adlandırılır. Eğer işaretlemede hiçbir atomik ateşlenme söz konusu değilse bu işaretlemeye gerçek işaretleme(*tangible marking*) denir. Bu tip ağlarda öncelikli olarak sistemde atomik ateşlemeye sebep olan geçişler atomik ateşlenmeler bitene kadar ateşlenir. Eğer izinli olabilen ani geçiş

kalmadıysa zamanlı geçişlerin ateşlenmesine başlanarak zamanın ilerletilmesine devam edilir.

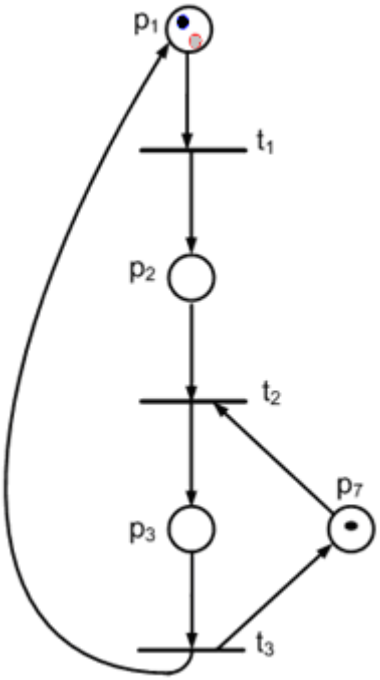
1.3 RENLİ PETRİ AĞLARI

İlk olarak K. Jensen tarafından 1970'lerin sonlarına doğru tanıtılan Renkli Petri Ağları, jetonlara ve geçişlere renk bilgisi atanaraktan Klasik Petri Ağlarıyla tanımlanması daha karmaşık olabilecek sistemleri daha basit ve anlaşılır modellemek amacıyla geliştirilmiştir. [14]



Şekil 1.20 : Çift İşlemcili Üretim Sisteminin Klasik Petri Ağı Modeli

Şekil 1.20 de verilen çift işlemcili üretim sisteminin Klasik Petri Ağı modeli verilmiştir.[2] Görüldüğü gibi birbirinin benzeri iki işi yapan sistemi modellemek için birbirinin benzeri iki Petri Ağı modeli birleştirilmiştir. Renkli Petri Ağları bu benzerliklerden yararlanarak sistemi daha basite indirger. Şekil 1.21 de sistemin Renkli Petri Ağı modeli incelendiğinde geçiş sayısının yarıya düştüğü ve yer sayısının da azaldığı görülmektedir.[2]



Şekil 1.21 : Çift İşlemcili Üretim Sisteminin Renkli Petri Ağı Modeli

Renkli Petri Ağı CPN = (P, T, N, O, C, Ω, M_0) gibi tanımlanır.

P: Ağdaki yerlerin kümesi,

T: Ağdaki geçişlerin kümesi,

N: Yerlerden Geçişlere yönelen okların kümesi,

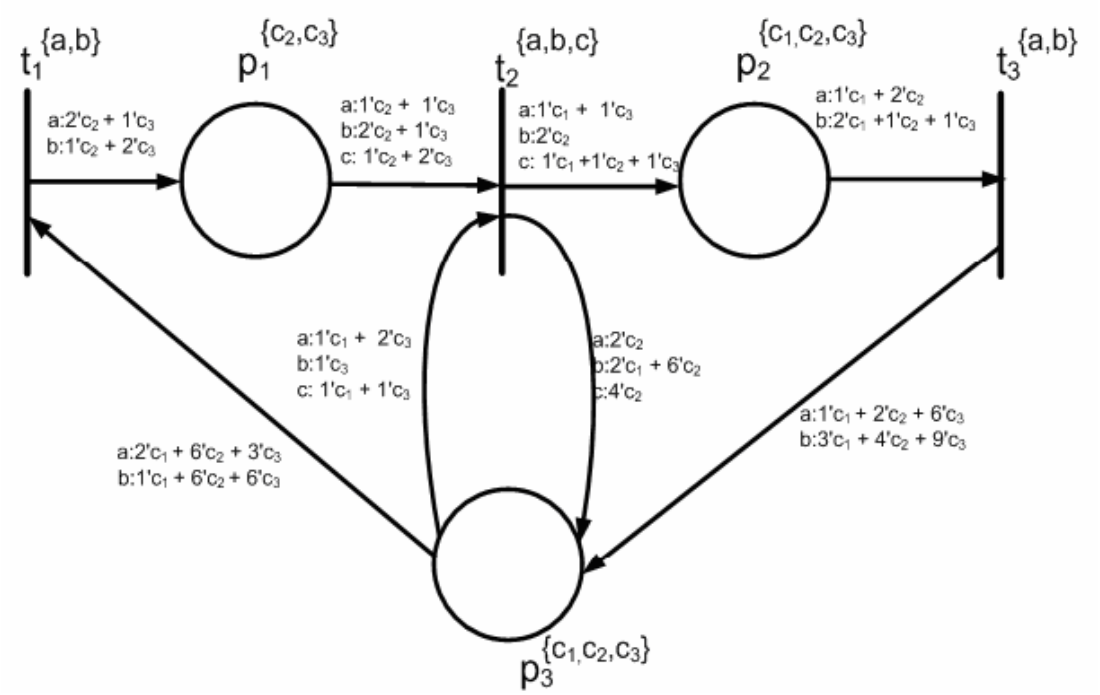
O: Geçişlerden yerlere yönelen okların kümesi.

C: Yerlerin Renk kümesi C(P) ve Geçişlerin Renk Kümesi C(T)' dir.

Ω: Sonlu sayıda renk kümesidir ve $\Omega = C(P) \cup C(T)$' dir.

M_0: İlk işaretleme(*marking*) matrisidir.

Şekil 1.22 deki örnek üzerinden Renkli Petri Ağlarının elementlerini inceleyelim.[2]



Şekil 1.22 : Renkli Petri Ağı Örneği

Şekil 4.3 deki örnek üzerinden açıklarsak

P = { P1, P2, P3 }

T = { T1, T2, T3 }

I = { (P1,T2,a), (P1,T2,b), (P1,T2,c), (P2,T3,a), (P2,T3,b), (P3,T1,a), (P3,T1,b), (P3,T2,a), (P3,T2,b), (P3,T2,c) }

O = { (T1,P1,a), (T1,P1,b), (T2,P2,a), (T2,P2,b), (T2,P2,c), (T2,P3,a), (T2,P3,b), (T2,P3,c), (T3,P3,a), (T3,P3,b) }

W(P1,T2,a, c1) = 0, W(P1,T2,a, c2) = 2, W(P1,T2,a, c3) = 1

W(P1,T2,b, c1) = 0, W(P1,T2,b, c2) = 1, W(P1,T2,b, c3) = 2

W (T2,P2,a,c1) = 1, W (T2,P2,a,c2) = 0, W (T2,P2,a,c3) = 1,

W (T2,P2,b,c1) = 0, W (T2,P2,b,c2) = 2, W (T2,P2,b,c3) = 0,

şeklinde yazılır.

Sistemin ilk işaretlemesi iki boyutlu matristir. Satırları yerler iken sütunları jeton renkleridir. Değerleri ise o yerdeki o renkte bulunan jeton sayısını verir. P1 yerinde 2 tane c2 renkli jeton ve 3 tane c3 renkli jeton olsun. M(P1,c1)=0, M(P1,c2)=2, M(P1,c3)=3 O halde M_0 ilk işaretleme matrisinin ilk satırı [0 2 3] şeklindedir.

İlk işaretleme matrisi aşağıdaki gibi yazılır.

$$M_0 = \begin{bmatrix} 0\,2\,3 \\ 9\,9\,1 \\ 7\,5\,6 \end{bmatrix}$$

Renkli Petri Ağlarında da geçişin izinli olması Klasik Petri Ağlarındaki gibidir. Geçişin herhangi bir geçiş rengi için izinli olabilmesi için o geçiş rengi için tanımlanmış geçiş koşulunu sağlayacak kadar uygun renkteki jetonların giriş yerlerinde bulunması yeterlidir. Renkli Petri Ağı örneğinde ilk işaretlemeye göre T2 geçişi a,b,c geçiş renkleri için ateşlenebilir, T3 geçişi a,b geçiş renkleri için ateşlenebilir iken T1 geçişi P3 yerinde yeterli sayıda c2 renkli jeton olmadığından ateşlenememektedir.

P1 yerindeki jetonların hangi geçiş rengi için kullanılacağı konusunda bir çelişki durumu söz konusudur. Klasik Petri Ağlarında geçişler arasında çelişki durumları söz konusu iken Renkli Petri Ağlarında geçiş renkleri arasında da çelişki oluşması durumu söz konusudur. Bu problemin de giderilebilmesi için geçiş renklerine de öncelik ataması yapılmaktadır.

Renkli Petri Ağları incelenirken çelişki durumlarında önce geçişler arası önceliğe bakılır. Eğer geçişin kendi geçiş renkleri arasında da çelişki durumu söz konusu ise geçiş renkleri arasındaki öncelik atamasına bakılır.

1.4 ZAMANLI RENKLİ PETRİ AĞLARI

Zamanlı Renkli Petri Ağları(*Timed Colored Petri Net*), Renkli Petri Ağlarının sistemleri kolay tasarlama yeteneğiyle Zamanlı Petri Ağlarının zamana dayalı analizinin getirdiği sistem hakkında daha geniş bilgi elde etme yetenekleri birleştirilerek elde edilmiş modelleme yaklaşımıdır.

Bu tez kapsamında geliştirilen algoritma ve yazılım Zamansız, Zamanlı, Renkli, Renksiz ağları ister geçişleri zamanlanmış ister yerleri zamanlanmış isterse jetonları zamanlanmış olsun belirli zamanlı ya da rastlantısal zamanlı olarak analiz edebilmektedir.

Zamanlı Renkli Petri Ağı CPN = (P, T, N, O, C, Ω, M_0, E_t, F_t, W_t, C_t, R) gibi tanımlanır.

P: Ağdaki yerlerin kümesi,

T: Ağdaki geçişlerin kümesi,

N: Yerlerden Geçişlere yönelen okların kümesi,

O: Geçişlerden yerlere yönelen okların kümesi.

C: Yerlerin renk kümesi C(P) ve Geçişlerin renk kümesi C(T)' dir.

Ω: Sonlu sayıda renk kümesidir ve $\Omega = C(P) \cup C(T)$' dir.

M_0: İlk işaretleme matrisi,

E_t: Geçişlerin izinli olma süresi kümesi,

F_t: Geçişlerin ateşlenme gecikmesi kümesi,

W_t: geçişin yeni bir ateşlenme için beklenmesi gereken süresi kümesi,

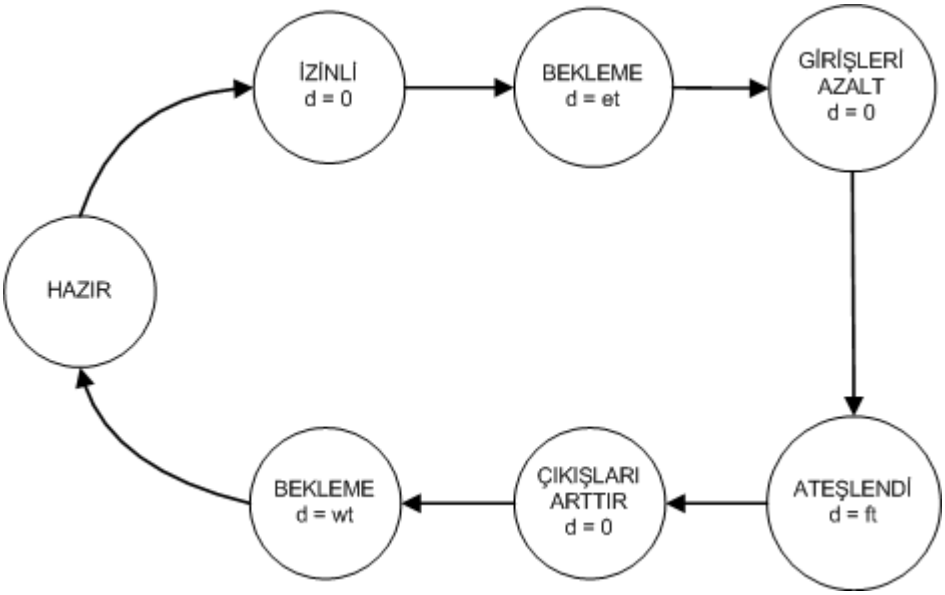
C_t: Yerleri Belirli Zamanlı Petri Ağı analizinde kullanılacak yerlerdeki jetonların izinli olabilmesi için gereken süreyi tutan küme,

R: Rastlantısal Zamanlı analizde kullanılacak oran değerleri kümesidir.

Belirtilen parametreler sayesinde birçok Petri Ağı analiz edilebilir.

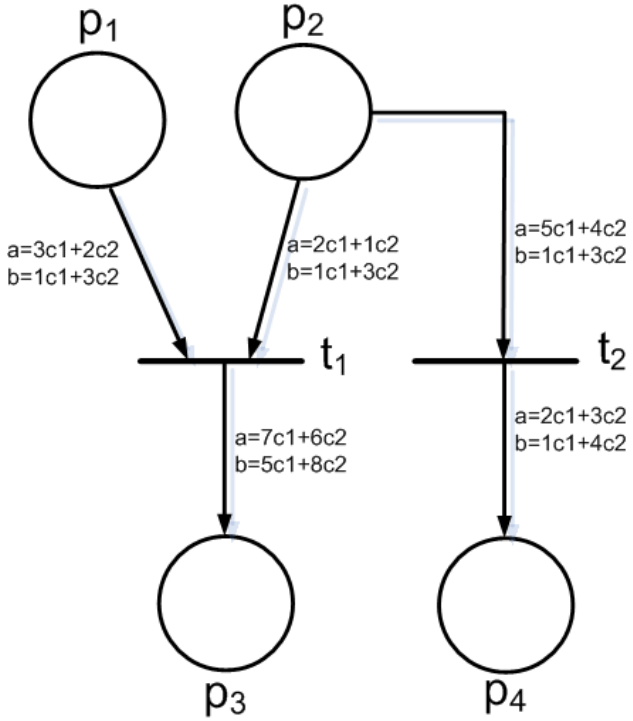
Zamanın jetonlara atanmasının zamanın yerlere atanmasından farkı Renkli Petri Ağlarına zaman özelliği katıldığında ortaya çıkmaktadır. Değişik jeton tiplerine değişik zaman bilgileri verilebilir. Hatta her yer için de değişik zaman bilgisi verilerekten ağ modellenebilir. Örneğin P1 yerinde c1 jetonu 2bz da elverişli oluyorken P2 yerinde 4bz da elverişli olabilir. Jetonların zamanlanması yerlerin zamanlandırılmasından daha esnek modelleme imkanı sunar. Renkli olmanın da getirdiği basitlik sayesinde daha karmaşık sistemleri daha rahat modelleyebiliriz.

Önceki bölümlerde Petri Ağları üzerine teorik alt yapı oluşturulmuştur. Bu bölümde ise Zamanlı ve Renkli Petri Ağların analizini yapabilmek için gerçeklenen yazılım tanıtılacaktır.



Şekil 1.23 : Geliştirilen Algoritma Geçişlere Ait Durum Geçiş Diyagramı

Algoritma anlatılıyorken Şekil 1.24 deki örnek bir Petri Ağ üzerinde açıklamalara yer verilecektir.



Şekil 1.24 : Basit Zamanlı Renkli Petri Ağı

Bu ağda a geçiş rengi b geçiş renginden T1 geçişi T2 geçişinden daha önceliklidir.

Ağın analizine en yüksek öncelikli geçiş incelenerek başlanır. Her geçişte de inceleme en yüksek öncelikli geçiş renginden başlatılır. Bu nedenle örnekte öncelikle T1 geçişinde a geçiş renginin ateşlenip ateşlenemeyeceği incelenir. a geçiş renginin ateşlenebilmesi için P1 girişinde en az 3 tane c1 ve 2 tane c2 jetonu P2 yerinde de 2 tane c1 ve 1 tane c2 jetonu olmalıdır. Eğer P1 yerinde yeterli jeton yoksa diğer giriş yerleri a geçiş rengi için incelenmeden ateşlenemez denilerek b geçiş rengi için yeterli jeton olup olmadığına bakılır. İkinci giriş oku da birinci giriş oku gibi aynı şekilde incelenir. İncelemeler sonrasında, birinci giriş oku üzerinde ve ikinci geçiş oku üzerinde ateşlenebilen aynı geçiş renkleri varsa ateşleme gerçekleşir. Örneğin, a geçiş rengi için P1 de ve P2 yeterli jetonlar var ise T1 a geçiş rengi için ateşleniyor olarak karar verilir.

Geçişler için şu durumlar mevcuttur:

Hazır: Bu durumda geçiş için ateşleme incelemesi yapılır.

İzinli: Eğer geçiş a rengi için ateşlenebiliyorsa geçiş bu renk için izinli hale gelir.

İzinliyken Bekleme: İzinli durumdan bir sonraki duruma geçmek için bir süre beklemek gerekiyorsa geçiş bu durumdadır.

Girişlerden Jeton Düşürme: İzinliyken Bekleme durumu geçtikten sonra giriş yerlerinden jeton düşürülebilir. Bu anda geçiş bu durumdadır.

Ateşleme: Girişlerden jeton düşürüldükten sonra ateşleme başlatılabilir. Geçiş ateşleme durumuna geçirilir.

Çıkışlara Jeton Verme: Ateşleme süresi geçtikten sonra çıkış yerlerine jetonlar eklenebilir. Bu anda geçiş bu durumdadır.

Sonraki Ateşlemeyi Bekleme: Çıkışlara jeton verildikten sonra, geçişin bir sonraki ateşlenebilirlik incelemesinin başlamasından önce bu durumda beklenilir.

Örneğin, T1 a geçiş rengi için ateşlenebilirse T1 a geçiş rengi için **Hazır** durumundan **İzinli** durumuna geçer. Eğer T1 a geçiş rengi için **İzinliyken Bekleme** süresi tanımlanmışsa bu duruma geçilir. Eğer bu süre yoksa hemen **Girişlerden Jeton Düşürme** durumuna geçilir.

Eğer ağda bir çelişki durumu varsa, çelişki durumu içerisinde olan bir geçişlerin **İzinliyken Bekleme** durumunda olması bu geçişin ilgili renk için ateşleneceğini garanti etmez. Çünkü çelişki durumundan ötürü daha öncelikli olan bir geçiş rengi

aynı geçişte veya bir başka geçişte herhangi bir renk ateşlenebilir hale gelebilir. Bu durumda **İzinliyken Bekleme** durumundaki ateşleme süreci iptal edilir ve geçiş ilgili renk için **Hazır** durumuna döner.

Geçişlerdeki ateşleme incelemesinin başladığı andaki işaretleme Girişlerden Jeton Düşürme ve Çıkışlara Jeton Verme durumları sonralarında değişir ve yeni işaretlemeler elde edilir. Yeni işaretlemeler elde edilmesi nedeniyle tüm geçişler tüm geçiş renkleri için tekrar ateşleme incelemesine tutulur.

Bir işaretlemeden başka bir işaretlemeye gidilemiyorsa bu noktada bu işaretleme için geçiş ateşlenebilirliliği incelemesi sonlandırılır. Bu tür işaretlemeler Ölü İşaretleme (*Dead-End*) olarak adlandırılır. Eğer bulunan bir işaretleme zaten önceden bulunmuşsa inceleme yine sonlandırılır. Bu tür işaretlemeler Eski İşaretleme (*Old-End)* olarak adlandırılır.

Önceliklerin dikkate alınması nedeni çelişki durumlarının çözümlemektir. Çelişki durumunda hangi geçiş renginin ateşleneceğine veya hangi geçişten incelemeye başlanacağına önceliklere bakılarak karar verilir. Böylece Petri Ağındaki soruna bir çözüm sunulmuş olunur.

Petri ağı tanımında Erişilebilirlik Grafiği' nın çıkarılması istenebilir. Bu durumda öncelikler, dolayısıyla çelişki durumları ve paralel çalışma ihmal edilir ve erişilebilecek tüm işaretlemeler bulunur.

Erişilebilirlik Grafiği içerisinde bazı işaretlemeler aranmak istenebilir. Bu işaretlemeler Petri Ağ tanımında belirtilir. Analiz çıkış dosyasında, bu işaretlemelerin ilk işaretlemeden oluşması için hangi geçişlerin hangi geçiş renkleri için ateşlenmesi gerektiği listelenir.

Üç farklı şekilde Petri Ağı incelemesi yapılmaktadır. Belirli zamanlı, rastlantısal zamanlı ve zamansız sistem modları mevcuttur. Belirli zamanlı modda Petri Ağ tanımında verilen İzinliyken Bekleme, Ateşleme ve Sonraki Ateşlemeyi Bekleme süreleri kullanılır. Eğer sürenin değeri ağ tanımında yoksa yani süre 0 sa, süre 1 birim zaman alınır. Zamanlı Petri ağlarında zamansız geçişlere izin verilmemiştir. Bu yazılımda olan bir kısıtlamadır.

Birim zaman, sistem moduna göre değişen bir değişkendir. Belirli zamanlı modda birim zaman girilen sürelerin OBEB i olarak alınır. Rastlantısal zamanlı modda ise birim zaman Petri Ağ tanımında ayrıca verilmelidir.

Geçişlerin ateşlenebilirlik incelemesi birim zamanda bir yapılır. Yani her inceleme sonrası sistem süresi bir birim zaman arttırılır. İnceleme yapılıyorken bulunulan işaretlemede kalınma süresi de bir arttırılır.

Rastlantısal zamanlı modda dikkate alınan bir başka değişken ise Oran dır. Bu oran değeri rastgele süre hesaplanmasında kullanılmaktadır. Bu rastgele süreler İzinliyken Bekleme, Ateşleme ve Sonraki Ateşlemeyi Bekleme süreleridir.

Geliştirilen algoritma ve yazılımın benzeri yazılımlara göre artıları

- Yerlerdeki jeton sayısı 1 ile sınırlı tutulmayıp istenilen değeri alabilmektedir. Geçişlere ve yerlere giren/çıkan okların sayısı birden fazla olabilmektedir.
- Çelişki durumlarından öncelik atamasıyla çözülmekte ve kullanıcı bilgilendirilmektedir.
- Analiz edilecek zaman aralığı gibi bir sınırlama yoktur.

Geliştirilen algoritma ve yazılımın benzeri yazılımlara göre eksileri

- Zamanlı analiz yapılırken atomik ateşlenmelere izin verilmemiştir.

2. GELİŞTİRİLEN YAZILIM

Tez çalışmaları kapsamında Klasik Petri Ağları, Zamanlı Petri Ağları, Renkli Petri Ağları ve Zamanlı Renkli Petri Ağlarını inceleyen MS-DOS tabanlı Türkçe ve İngilizce dillerini destekleyen "Çok Fonksiyonel Zamanlı Renkli Petri Ağları Yazılımı" adında yazılım geliştirilmiştir. Yazılım *petrinet* uzantılı giriş dosyasına ihtiyaç duymaktadır. Bu dosyanın içeriğinde analizi yapılacak ağın yazılıma özgü modelleme diliyle anlatıldığı kodları içerir. Bunun yanı sıra hangi tip analiz yapılacağı ve sonuçların ne kadar ayrıntılı olacağı gibi analiz sonuçlarının seviyesi seçilir.

Üç farklı yöntemle giriş dosyası programın kullanması için giriş olarak verilebilir. Birinci yöntem giriş dosyasının adı "*case.petrinet*" olarak yapılıp programın çalıştırıldığı dosya (*exe*) ile aynı dizine konularak yapılabilir. Çalıştırılabilir program aynı klasörde bulunan *case.petrinet* dosyasını görüp işleyecektir. İkinci yöntem Microsoft Windows' un "*Command.exe*" programı çalıştırılarak program çalıştırılabilir. Örneğin yazılımın adı tool.exe ve giriş dosyasının adı "*case01.petrinet*" ise komut satırına "*tool.exe case01.petrinet*" yazılıp "ENTER" tuşuna basarak analiz başlatılabilir. Üçüncü yöntemse sürükle bırak işlemidir. Giriş dosyası fare ile programın üzerine sürüklenirse de program analize başlayacaktır.

Yazılımın mantığı şu şekildedir. Sisteme girilen parametreler ışığında ilk işaretlemeye bakılaraktan ateşlenebilir geçişler tespit edilir. Eğer bu geçişler arasında çelişki varsa öncelik atamasına göre karar alınır. Eğer birkaç geçiş birlikte ateşlenebiliyorsa tüm geçişler birlikte ateşlenir. Böylece yazılım benzeri yazılımlardan farklı olarak paralel işlemi(*parallel processing*) destekler.

Girilen Petri Ağının daha verimli analizi için kullanıcının giriş olarak verdiği öncelik atamasına göre en öncelikli geçişten başlayarak sistem incelenmeye başlanır. Eğer analize daha önceliksiz bir geçişten başlansaydı çelişki durumu oluştuğunda çelişkiye sebep olan geçiş daha öncelikli ise bu geçişin analizi bırakılıp öncelikli geçişin analizi yapılması gerekirdi. Bu sırada yapılan analizler bellekte saklanması tercih edilirse sistemin en düşük öncelikli geçişinden analiz yapılmaya başlansaydı ve

çelişki dolu bir modelle çalışıyor olsaydık gereksiz bellek ve zaman kayıpları oluşacaktır. Bu nedenle öncelikli geçişten başlayarak analiz yapmak daha doğru bir çözümdür.

Geliştirilen yazılım bir ağın Erişilebilirlik Grafiğindeki durumların listesini çıkarabilir. Bu analiz tipiyle sistemin muhtemel gidebileceği tüm durumlar bulunabilir. Erişilebilirlik Analizi yapılırken çelişki durumlarına ve paralel işleme bakılmaz. Herhangi bir işaretlemede tüm olası ateşlemeler yapılır. Bulunan işaretlemeler daha önce ağaçta yok ise yeni işaretleme olarak listeye eklenir. Bu şekilde tüm durumlar bulunurken bulunan her yeni işaretlemeye gelinen işaretleme bilgisi ve bu işaretlemeye gelinmesine sebep olan ateşleme dizisi bilgi olarak sunulur. Böylece ağacın üzerinde nasıl gezinileceğine hakkında kullanıcı bilgilendirilmiş olunur.

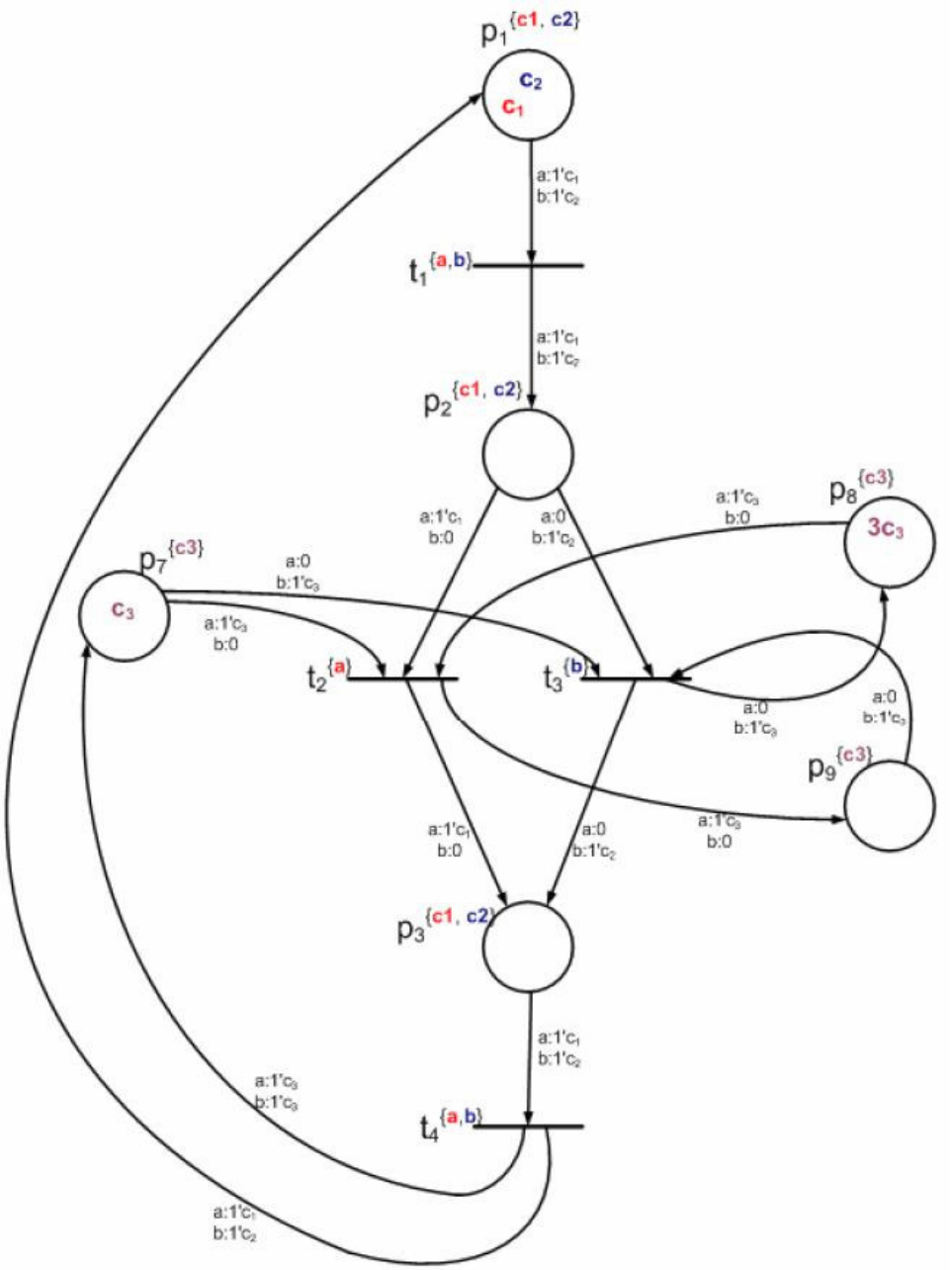
Şekil 2.1 deki(ÖRNEK–1) Renkli Petri Ağı örneği [2] kullanılaraktan geliştirilen yazılımın giriş dosyası formatından bahsedilecek ve benzeri yazılımlara göre üstünlükleri vurgulanacaktır. Örnekteki Petri Ağı 6 yer, 4 geçiş, 3 jeton rengi ve 2 geçiş renginden oluşmaktadır. Sistemin ilk işaretlemesine göre P1 yerinde 1 tane c1 ve 1 tane c2 jetonu bulunmaktadır. P7 ve P8 yerlerinde birer tane c3 jetonu bulunmaktadır.

Şekil 2.1 de verilen Renkli Petri Ağı örneği iki robotun çalıştığı bir üretim sistemi üzerinedir. Sistemde R1 ve R2 olmak üzere iki robot bulunmaktadır. R1 robotu A cinsinden paketleri işlerken R2 robotu B cinsinden paketleri işlemektedir. Robotlar ortak bir üretim bandı üzerinde çalışmaktadırlar. Bu nedenle iki robotun aynı anda ortak üretim bandına girmesi engellenmelidir. Ayrıca sonuç ürünün saklanabileceği tampon miktarı 3 olsun. Her bir tampona bir tane A ve bir tane B cinsi işlenmiş paket konsun. Sistemin Renkli Petri Ağı modelinin bileşenleri ve tanımları Çizelge 2.1 de verilmiştir.

Çizelge 2.1 : Renkli Petri Ağı Örneğinin Bileşenleri ve Tanımları

P1	Robot R1 ve/veya R2 çalışma alanı dışında beklemededir.
P2	Robot R1 ve/veya R2 çalışma alanına girmek için izin beklemektedir.
P3	Robot R1 ve/veya R2 çalışma alanında işlem yapmaktadır.

P7	İki robotu aynı anda ortak alanda olmalarını engellemek amaçlı
P8 / P9	Sıradaki tampondaki boş/dolu yeri temsil eder.
T1	Robot R1 (R2) çalışma alanına girmek için istek yapar.
T2	Robot R1 çalışma alanına girer.
T3	Robot R2 çalışma alanına girer.
T4	Robot R1 (R2) çalışma alanını terk eder.
c1	R1 Robotunu temsil eder.
c2	R2 Robotunu temsil eder.
c3	Uygunluk ve kapasite belirleme amaçlı kullanılır.
a	R1 Robotu için ateşlenme olaylarını temsil eder.
b	R2 Robotu için ateşlenme olaylarını temsil eder.



Şekil 2.1 : Erişebilirlik Ağacı Çıkarılacak Renkli Petri Ağı Örneği(ÖRNEK–1)

```
# case04_tr.petrinet: Serkan'ın Tezindeki Örnek
tr                        # Tüm çıktılar Türkçe dilinde verilir
ms 0                      # Sadece Sonuçlari Göster
ay                        # agaci_yaz
sm zz                     # zamansiz
eg                        # Erişilebilirlik Grafiğini Bul
jeton_rengi c1 c2 c3      # Modelde 3 tane jeton rengi var
#r c1 c2 c3
gecis_rengi a(0) b(1)     # Modelde 2 tane geçiş rengi var
#gr a(0) b(1)
#gecis_rengi_onceligi a(0) b(1)
#gro a(0) b(1)
yer P1(1c1,1c2) P2 P3 P7(1c3) P8(3c3) P9
#y P1(1c1,1c2) P2 P3 P7(1c3) P8(3c3) P9
gecis T1(0) T2(1) T3(2) T4(3)
#g T1(0) T2(1) T3(2) T4(3)
#gecis_onceligi T1(0) T2(1) T3(2) T4(3)
#go T1(0) T2(1) T3(2) T4(3)

gi T1 P1 a(1c1) b(1c2)
ci T1 P2 a(1c1) b(1c2)
gi T2 P2 a(1c1)
gi T2 P7 a(1c3)
gi T2 P8 a(1c3)
ci T2 P3 a(1c1)
ci T2 P9 a(1c3)
gi T3 P2 b(1c2)
gi T3 P7 b(1c3)
gi T3 P9 b(1c3)
ci T3 P3 b(1c2)
ci T3 P8 b(1c3)
gi T4 P3 a(1c1) b(1c2)
ci T4 P1 a(1c1) b(1c2)
ci T4 P7 a(1c3) b(1c3)

# Zaman değerleri
ios T1(2a,2b) T2(2a) T3(2b) T4(2a,2b)
as  T1(3a,3b) T2(3a) T3(3b) T4(3a,3b)
bs  T1(4a,4b) T2(4a) T3(4b) T4(4a,4b)
rs P1(2c1,2c2) P2(6c1,5c2) P3(3c1,1c2) P7(4c1,4c2,4c3) P8(3c3) P9
#
or T1(0.5a,0.5b) T2(0.5a) T3(0.5b) T4(0.5a,0.5b)
#bz 0.25
#
ai P1 P2(1c2) P3 P7 P8(2c3) P9(1c3)
ai P1 P2(1c1,1c2) P3 P7(1c3) P8(1c3) P9(2c3)
ai P1(1c1) P2(1c2) P3 P7(1c3) P8 P9(3c3)
```

Şekil 2.2 : Erişebilirlik Ağacı Çıkarma Giriş Dosyası Örneği

Şekil 2.2 de örnek olarak verilen Renkli Petri Ağına ait giriş dosyası verilmiştir. Yorum operatörü olarak "#" karakteri kullanılmıştır. Eğer bir satırın başında bu operatör varsa o satır program tarafından dikkate alınmaz. Yazılım, giriş dosyasının ilk satırlarında çalışma parametrelerinin ne olacağına hakkında bilgiler bekler.

Sonuçların ne kadar ayrıntılı olacağı ya da hangi modda çalışılacağı gibi. Yazılım için önemli anahtar kelimeler de verilmiştir.

Çizelge 2.2 : Geliştirilen Yazılımın Anahtar Kelimeleri

Yazılımın Anahtar Kelimeleri			
Türkçe		**İngilizce**	
turkce	tr	turkish	tr
ingilizce	en	english	en
mesaj_seviyesi	ms	Debug_level	dl
agaci_yaz	ay	dump_graph	dg
agaci_cikisa_yaz	acy	dump_stdout	ds
sistem_modu	sm	system_mode	sm
zamansiz	zz	no_time	nt
belirli_zamanli	bz	deterministic	d
rastlantisal_zamanli	rz	stochastic	s
yerleri_belirli_zamanli	ybz	placed_deterministic	pd
erisilebilirlik_grafigi	eg	reachability_graph	rg
jeton_rengi, renk	r	Token_color	c
gecis_rengi	gr	transition_color	tc
gecis_rengi_onceligi	gro	transition_color_priority	tcp
yer	y	place	p
gecis	g	transition	t

gecis_onceligi	go	transition_priorities	tp
giris	gi, ->	input	in, ->
cikis	ci, <-	Output	ou, <-
izinli_olma_suresi, yetkinlik_suresi	ios, ys	enabled_time	et
ateslenme_suresi	as	firing_time	ft
bekleme_suresi	bs	waiting_time	wt
renk_suresi	rs	color_time	ct
birim_zaman	bz	unit_time	ut
aranan_isaretleme	ai	marking_request	mr
oran	or	rate	r

"mesaj_seviyesi" parametresi sayesinde yazılımın ağın analizini yaparken neler yaptığını da çıkış dosyasına yazdırabilir(*ms 4*) ya da sadece sonuçları(*ms 0*) çıkış dosyasına yazmayı seçebiliriz.

Yazılımın dört farklı ana çalışma modu vardır. Çalışma modu "sistem_modu" parametresiyle seçilir. Bu modlar Zamansız, Belirli Zamanlı, Rastlantısal Zamanlı ve Yerleri Belirli Zamanlı' dır. Zamansız modu kendi içinde iki farklı moda ayrılır. Bunlar Erişilebilirlik Ağacını çıkarma ve Koşullu(*constraint*) Analizdir. Eğer çalışma modu zamansız ise ve eg parametresi de giriş dosyasında belirtilmişse yazılım ağın Erişilebilirlik Ağacını çıkarmaya çalışacaktır. Eğer mod zamansız ancak eg parametresi yazılmamışsa kod zamansız koşullu modda yani çelişki durumlarında öncelik atamasına göre çözümlemenin yapıldığı ve paralel işleme izin verildiği şeklinde ağın analizi yapılır. Diğer modlarda eg parametresinin yazılmış olmasının etkisi yoktur.

Çalışma parametrelerinden bir diğeriyse "agaci_yaz" parametresidir. Bu parametre sayesinde Erişebilirlik Grafiğinin görsel olarak "*Graphviz"* programında çizdirilebilmesi için gereken *gv* uzantılı giriş dosyasının üretilmesini sağlar.

Çalışma parametrelerinden sonra jeton renkleri bilgisi girilir. "jeton_rengi" parametresinden sonra her boşluk karakteri görülene kadar ki metin ifadeleri bir jeton rengine karşılık gelir. Verilen örneğe göre jeton renkleri c1 c2 ve c3 olabilmektedir.

Geçiş renklerinin neler olacağı bilgisi "gecis_rengi" parametresiyle girilir. Bu parametreden sonra her boşluk karakteri görülene kadar ki metin ifadeleri bir jeton rengine karşılık gelir. Verilen örneğe göre geçiş renkleri a ve b olabilmektedir. Geçiş renklerinin her birinin yanında parantez içerisinde belirtilen bir sayı ifadesi görülmektedir. Bu sayı ifadesi çelişki durumlarında hangi geçiş renginin daha öncelikli olacağının belirlenmesinde kullanılır. Değeri en düşük olan geçiş rengi en öncelikli geçiş rengidir. Örnekte a(0) ve b(1) olduğundan a geçiş rengi b geçiş renginden daha önceliklidir.

Geçiş renklerinin önceliği geçiş renkleri tanımlanırken verilebilineceği gibi "gecis_rengi_onceligi" parametresiyle de verilebilinir. Eğer bu değişken tanımlandıysa "gecis_rengi" daki değerler ezilir yani dikkate alınmaz. Bu parametre sayesinde sistemde geçiş renklerinin öncelikleri değiştirildikçe sistemin değişik davranışlar gösterip göstermediği incelenebilir.

Ağdaki yerlerin tanımlanması için *"yer"* parametresi kullanılır. Bu parametreden sonra her boşluk karakteri görülene kadar ki metin ifadeleri bir yer ismine karşılık gelir. Verilen örneğe göre yer isimleri P1, P2, P3, P7, P8 ve P9' dur.

Ağdaki geçişlerin tanımlanması için *"gecis"* parametresi kullanılır. Bu parametreden sonra her boşluk karakteri görülene kadar ki metin ifadeleri bir geçiş ismine karşılık gelir. Verilen örneğe göre geçiş isimleri T1, T2, T3 ve T4' dür. Geçiş isimlerinin her birinin yanında parantez içerisinde belirtilen sayı ifadesi görülmektedir. Bu sayı ifadesi çelişki durumlarında hangi geçişin daha öncelikli olacağının belirlenmesinde kullanılır. Değeri en düşük olan geçiş en öncelikli geçiştir. Örnekte geçiş isimlendirmesi T1(0), T2(1), T3(2) ve T4(3) olduğundan T1 geçişi T2, T3 ve T4 geçişlerinden, T2 geçişi T3 ve T4 geçişlerinden, T3 geçişi T4 geçişinden daha önceliklidir.

Geçişlerin önceliği geçiş isimlendirme tanımlanırken verilebilineceği gibi "gecis_onceligi" parametresiyle de verilebilinir. Eğer bu değişken tanımlandıysa *"gecis"* deki değerler ezilir yani dikkate alınmaz. Bu parametre sayesinde sistemde

geçiş öncelikleri değiştirildikçe sistemin değişik davranışlar gösterip göstermediği incelenebilir.

"giris" parametresi bir geçişin girişinde bulunan giriş yeri ile geçişin ilişkisini ifade eder. "gi T1 P1 a(1c1) b(1c2)" giriş ifadesinde ilk iki parametre T1 geçişiyle bu geçişin girişi olan P1 yeri arasındaki ilişkiyi ifade ettiğini gösterir. Sonraki parametrelerde ilgili geçiş renginde geçişin izinli olabilmesi için gereken jeton miktarı ve jeton tipini belirtir. Üçüncü parametre a(1c1) de a geçiş renginde izinli olabilmesi için P1 yerinde en azından 1 tane c1 jetonu olmalıdır. Eğer birden fazla tipte jetona ihtiyaç duyuluyorsa "," operatörü kullanılarak diğer jetonlarda miktarlarıyla birlikte ifade edilir. Örneğin ifade "gi T1 P1 a(1c1,4c2,7c3) b(1c2,5c3)" şeklinde olsaydı geçişin a geçiş renginde izinli olabilmesi için en az 1 tane c1, 4 tane c3 ve 7 tane c3 jetonun P1 yerinde aynı anda olması gerekirdi.

"cikis" parametresi bir geçişin çıkışında bulunan çıkış yeri ile geçişin ilişkisini ifade eder. *Giriş* ifadesiyle kullanımı aynıdır sadece geçişin giriş yeri değil çıkış yeridir.

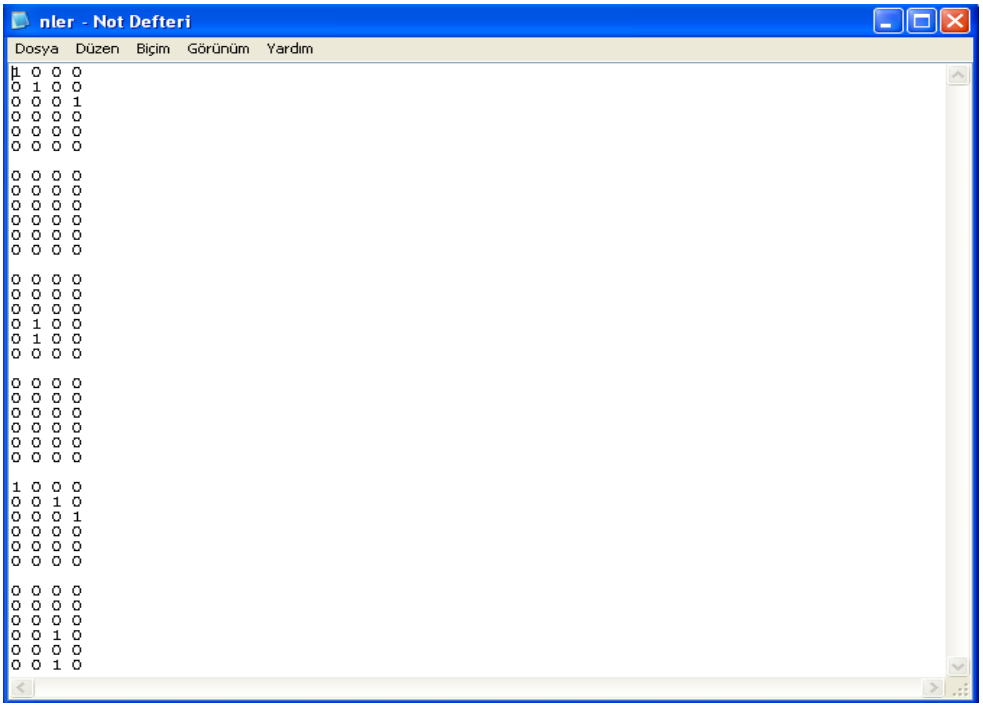
Geçişlere ait zaman ifadeleri *"izinli_olma_suresi"* parametresiyle verilir. Tüm parametreleri geçiş isimleridir. Geçiş isimlerinin her birinin yanında parantez içerisinde belirtilen sayı ifadesi görülmektedir. Parantez içerisindeki bu ifadeler o geçiş için tanımlanmış geçiş renklerinin her biri için izinli olma süresi değeri yazılır. Örnekteki "ios T1(2a,2b) T2(2a) T3(2b) T4(2a,2b)" zaman ifadesinde T1 geçişinin a geçiş rengi için izinli olma süresi 2 birim zaman, b geçiş rengi için izinli olma süresi de 2 birim zamandır. Geçişin giriş yerlerindeki koşulların tümü sağlandıktan sonra izinli olabilmesi için beklenmesi gereken süre değeri et parametresi sayesinde verilir. Benzer şekilde ateşlenme süresi ve yeni bir ateşleme için bekleme süresi değerleri de verilir. Ateşlenme süresi gecikmeleri için *"ateslenme_suresi"*, bekleme süreleri girişi için *"bekleme_suresi"* değerleri girilir. Ayrıca jetonların ilgili giriş yerinde elverişsiz durumdan elverişli duruma geçebilmeleri için gereken gecikme süreleri "renk_suresi" parametresiyle girilir. Sadece diğerlerinden farkı parametreleri yerler ve jetonlar renklerine göre yazılmasıdır.

Rastlantısal Zamanlı Petri Ağlarında kullanılan oran parametresi *"oran"* parametresiyle girilir. Rastlantısal süreç analiz edilirken kullanılacak ilk rastlantısal zaman ifadelerin elde edilmesinde kullanılacak değer bu parametreyle girilir. Her geçişin geçiş renkleri için farklı oran değeri verilebilir.

Birim zaman parametresi olan *"birim_zaman"*) iki fonksiyonu olan bu parametrenin asıl işlevi Rastlantısal Zamanlı Petri Ağlarını incelerken bulunan rastlantısal zamanların OBEB(En büyük ortak bölen) değerine ihtiyaç vardır. Bulunan tüm rastlantısal değerler bu birim zaman parametresinin katı olacak şekilde yuvarlanır. Çünkü hazırlanan yazılım girilen modelin zaman analizini yaparken kullanıcının girdiği zaman değerlerinin OBEB'ini bulur. Eğer yoksa tüm değerler tamsayı ise birim zaman 1 alınır. Eğer ondalık sayı ise 0.1, 0.01, 0.001 gibi tüm değerleri bölebilecek bir değer seçilir. Belirli Zamanlı Petri Ağ analizinde bu işlem bu şekilde yazılım tarafından yapılmasına rağmen Rastlantısal Zamanlı Petri Ağlarında rastgele değerlerin yuvarlanabilmesi için bir değer kullanıcı tarafından girilmesi daha doğrudur. Eğer kullanıcı bu değeri tanımlamazsa varsayılan değer olan 0.25 kullanılacaktır. Bu parametrenin ikinci işlevi ise Belirli Zamanlı Petri Ağları analiz edilirken kullanıcının belirtmediği e_t, f_t, w_t ve c_t zamanları için kullanılacak değer olarak seçilir. Böylece sisteme hızlı bir şekilde zaman ifadesi tanımlamada da kullanılabilinir. Zamansız Petri Ağı analizi yapıldığında zaman ifadeleri dikkate alınmaz.

Ağ analizi yapılırken ilk işaretlemeden gidilecek tüm işaretlemeler bulunur. Ancak kullanıcı belli bir işaretlemeye sistemin ulaşıp ulaşamadığını merak edebilir. Bunun için *"aranan_isaretleme"* parametresi kullanılabilir. Yer tanımlamakla(*yer* parametresi) aynı şekilde tanımlanır. Eğer tüm yerlerin bilgisi girilirse aranan işaretlemeyi bulmaya çalışır. Eğer tüm yerlerin bilgisi girilmeyip sadece incelenmek istenen yerlerin bilgileri girilirse bu işaretlemeleri kapsayan işaretlemelerin listesi çıkarılır. Bu işaretlemeye ulaşabilmek için başlangıç işaretlemesinden bu işaretlemeye kadar geçilen olası tüm yollar ve ateşlenme dizilerinin listesi de çıkış dosyasında gösterilir.

Benzeri yazılımlarda birçok giriş dosyası kullanılmaktadır. Örneğin Y. Serkan[1] tez çalışmasında sunduğu Renkli Petri Ağlarını inceleyen CPN-RPA v1.0 yazılımında 5 farklı giriş dosyasına ihtiyaç duymaktadır. Örneğin ağın giriş yerlerindeki koşulları tutan "nler.txt" dosyasının bir örneği Şekil 6.3 de verilmiştir. Görüldüğü gibi birçok matristen oluşan bu ifade şeklinin hazırlanması ya da hazırlandıktan sonra değişikliğe gidilmesi ve hatta yapılan bir hatanın görülmesi çok zordur. Benzeri şekilde Y. Ahmet[3] tez çalışmasında da birçok giriş dosyasına ihtiyaç duyulmakta ve matris yapısı kullanılmaktadır.



Şekil 2.3 : Benzer Bir Yazılımın Giriş Dosyası Örneği

Erişebilirlik Ağacı Çıkarılacak Renkli Petri Ağı Örneği geliştirilen yazılım ile analiz edildiğinde çıkış dosyası aşağıdaki gibidir.

Şekil 2.4 de görüldüğü gibi analiz sonucunda verilen Renkli Petri Ağı örneği için ilk işaretlemeyle birlikte erişilebilir 28 durum vardır. M7 ve M12 işaretlemelerinde çelişki durumlarıyla karşılaşıldığı sonuçlarda gösterilmektedir. M7 deki çelişki T2 ve T3 geçişlerine ortak giriş yeri olan P7 den kaynaklanmaktadır. P7 de bulunan 1 tane c3 jetonu T2 geçişini ve T3 geçişini ayrı ayrı izinli yapabiliyorken birlikte izinli yapamadığından çelişki durumu oluşmuştur. Aynı sebeplerden ötürü çelişki durumu M12 işaretlemesinde de görülmektedir.

Şekil 2.4 de de görüldüğü gibi her işaretlemenin altında bu işaretlemeden gidilebilinecek işaretlemeler ve bu işaretleme ulaşan işaretlemelerin listesi ateşleme dizisiyle birlikte verilmektedir. Verilen örnekte aranan işaretlemelerin bulunup bulunmadığı sonuçları Şekil 2.5 deki gibidir.

```
### Isaretleme M0 ###
# Yer  :  c1  c2  c3
#    P1:   1   1   0
#    P2:   0   0   0
#    P3:   0   0   0
#    P7:   0   0   1
#    P8:   0   0   3
#    P9:   0   0   0
# ILK Isaretleme M0
# M8 -> M0: T4,b
# M0 -> M1: T1,a
# M0 -> M2: T1,b

### Isaretleme M1 ###
# Yer  :  c1  c2  c3
#    P1:   0   1   0
#    P2:   1   0   0
#    P3:   0   0   0
#    P7:   0   0   1
#    P8:   0   0   3
#    P9:   0   0   0
# M0 -> M1: T1,a
# M10 -> M1: T4,b
# M1 -> M3: T1,b
# M1 -> M4: T2,a

### Isaretleme M2 ###
# Yer  :  c1  c2  c3
#    P1:   1   0   0
#    P2:   0   1   0
#    P3:   0   0   0
#    P7:   0   0   1
#    P8:   0   0   3
#    P9:   0   0   0
# M0 -> M2: T1,b
# M2 -> M3: T1,a
# ESKI Isaretleme M3

### Isaretleme M3 ###
# Yer  :  c1  c2  c3
#    P1:   0   0   0
#    P2:   1   1   0
#    P3:   0   0   0
#    P7:   0   0   1
#    P8:   0   0   3
#    P9:   0   0   0
# M1 -> M3: T1,b
# M2 -> M3: T1,a
# M3 -> M5: T2,a
### Isaretleme M4 ###
# Yer  :  c1  c2  c3
#    P1:   0   1   0
#    P2:   0   0   0
#    P3:   1   0   0
#    P7:   0   0   0
#    P8:   0   0   2
#    P9:   0   0   1
# M1 -> M4: T2,a
# M4 -> M5: T1,b
# M4 -> M27: T4,a
# ESKI Isaretleme M5
# ESKI Isaretleme M27

### Isaretleme M5 ###
# Yer  :  c1  c2  c3
#    P1:   0   0   0
#    P2:   0   1   0
#    P3:   1   0   0
#    P7:   0   0   0
#    P8:   0   0   2
#    P9:   0   0   1
# M3 -> M5: T2,a
# M4 -> M5: T1,b
# M5 -> M6: T4,a

### Isaretleme M6 ###
# Yer  :  c1  c2  c3
#    P1:   1   0   0
#    P2:   0   1   0
#    P3:   0   0   0
#    P7:   0   0   1
#    P8:   0   0   2
#    P9:   0   0   1
# M5 -> M6: T4,a
# M27 -> M6: T1,b
# M6 -> M7: T1,a
# M6 -> M8: T3,b

### Isaretleme M7 ###
# Yer  :  c1  c2  c3
#    P1:   0   0   0
#    P2:   1   1   0
#    P3:   0   0   0
#    P7:   0   0   1
#    P8:   0   0   2
#    P9:   0   0   1
# M6 -> M7: T1,a
# M25 -> M7: T1,b
# M7 -> M9: T2,a
# M7 -> M10: T3,b
# CELISKI P7: T2,a & T3,b
### Isaretleme M8 ###
# Yer  :  c1  c2  c3
#    P1:   1   0   0
#    P2:   0   0   0
#    P3:   0   1   0
#    P7:   0   0   0
#    P8:   0   0   3
#    P9:   0   0   0
# M6 -> M8: T3,b
# M8 -> M10: T1,a
# M8 -> M0: T4,b
# ESKI Isaretleme M10
# ESKI Isaretleme M0

### Isaretleme M9 ###
# Yer  :  c1  c2  c3
#    P1:   0   0   0
#    P2:   0   1   0
#    P3:   1   0   0
#    P7:   0   0   0
#    P8:   0   0   1
#    P9:   0   0   2
# M7 -> M9: T2,a
# M26 -> M9: T1,b
# M9 -> M11: T4,a

### Isaretleme M10 ###
# Yer  :  c1  c2  c3
#    P1:   0   0   0
#    P2:   1   0   0
#    P3:   0   1   0
#    P7:   0   0   0
#    P8:   0   0   3
#    P9:   0   0   0
# M7 -> M10: T3,b
# M8 -> M10: T1,a
# M10 -> M1: T4,b
# ESKI Isaretleme M1

### Isaretleme M11 ###
# Yer  :  c1  c2  c3
#    P1:   1   0   0
#    P2:   0   1   0
#    P3:   0   0   0
#    P7:   0   0   1
#    P8:   0   0   1
#    P9:   0   0   2
# M9 -> M11: T4,a
# M24 -> M11: T1,b
# M11 -> M12: T1,a
# M11 -> M13: T3,b

### Isaretleme M12 ###
# Yer  :  c1  c2  c3
#    P1:   0   0   0
#    P2:   1   1   0
#    P3:   0   0   0
#    P7:   0   0   1
#    P8:   0   0   1
#    P9:   0   0   2
# M11 -> M12: T1,a
# M20 -> M12: T1,b
# M12 -> M14: T2,a
# M12 -> M15: T3,b
# CELISKI P7: T2,a & T3,b
### Isaretleme M13 ###
# Yer  :  c1  c2  c3
#    P1:   1   0   0
#    P2:   0   0   0
#    P3:   0   1   0
#    P7:   0   0   0
#    P8:   0   0   2
#    P9:   0   0   1
# M11 -> M13: T3,b
# M13 -> M15: T1,a
# M13 -> M27: T4,b
# ESKI Isaretleme M15

### Isaretleme M14 ###
# Yer  :  c1  c2  c3
#    P1:   0   0   0
#    P2:   0   1   0
#    P3:   1   0   0
#    P7:   0   0   0
#    P8:   0   0   0
#    P9:   0   0   3
# M12 -> M14: T2,a
# M21 -> M14: T1,b
# M14 -> M16: T4,a
```

```
### Isaretleme M15 ###
# Yer  :  c1  c2  c3
#    P1:   0   0   0
#    P2:   1   0   0
#    P3:   0   1   0
#    P7:   0   0   0
#    P8:   0   0   2
#    P9:   0   0   1
# M12 -> M15: T3,b
# M13 -> M15: T1,a
# M15 -> M25: T4,b

### Isaretleme M16 ###
# Yer  :  c1  c2  c3
#    P1:   1   0   0
#    P2:   0   1   0
#    P3:   0   0   0
#    P7:   0   0   1
#    P8:   0   0   0
#    P9:   0   0   3
# M14 -> M16: T4,a
# M22 -> M16: T1,b
# M16 -> M17: T1,a
# M16 -> M18: T3,b

### Isaretleme M17 ###
# Yer  :  c1  c2  c3
#    P1:   0   0   0
#    P2:   1   1   0
#    P3:   0   0   0
#    P7:   0   0   1
#    P8:   0   0   0
#    P9:   0   0   3
# M16 -> M17: T1,a
# M23 -> M17: T1,b
# M17 -> M19: T3,b

### Isaretleme M18 ###
# Yer  :  c1  c2  c3
#    P1:   1   0   0
#    P2:   0   0   0
#    P3:   0   1   0
#    P7:   0   0   0
#    P8:   0   0   1
#    P9:   0   0   2
# M16 -> M18: T3,b
# M18 -> M19: T1,a
# M18 -> M24: T4,b
# ESKI Isaretleme M19

### Isaretleme M19 ###
# Yer  :  c1  c2  c3
#    P1:   0   0   0
#    P2:   1   0   0
#    P3:   0   1   0
#    P7:   0   0   0
#    P8:   0   0   1
#    P9:   0   0   2
# M17 -> M19: T3,b
# M18 -> M19: T1,a
# M19 -> M20: T4,b

### Isaretleme M20 ###
# Yer  :  c1  c2  c3
#    P1:   0   1   0
#    P2:   1   0   0
#    P3:   0   0   0
#    P7:   0   0   1
#    P8:   0   0   1
#    P9:   0   0   2
# M19 -> M20: T4,b
# M24 -> M20: T1,a
# M20 -> M12: T1,b
# M20 -> M21: T2,a
# ESKI Isaretleme M12

### Isaretleme M21 ###
# Yer  :  c1  c2  c3
#    P1:   0   1   0
#    P2:   0   0   0
#    P3:   1   0   0
#    P7:   0   0   0
#    P8:   0   0   0
#    P9:   0   0   3
# M20 -> M21: T2,a
# M21 -> M14: T1,b
# M21 -> M22: T4,a
# ESKI Isaretleme M14

### Isaretleme M22 ###
# Yer  :  c1  c2  c3
#    P1:   1   1   0
#    P2:   0   0   0
#    P3:   0   0   0
#    P7:   0   0   1
#    P8:   0   0   0
#    P9:   0   0   3
# M21 -> M22: T4,a
# M22 -> M23: T1,a
# M22 -> M16: T1,b
# ESKI Isaretleme M16

### Isaretleme M23 ###
# Yer  :  c1  c2  c3
#    P1:   0   1   0
#    P2:   1   0   0
#    P3:   0   0   0
#    P7:   0   0   1
#    P8:   0   0   0
#    P9:   0   0   3
# M22 -> M23: T1,a
# M23 -> M17: T1,b
# ESKI Isaretleme M17

### Isaretleme M24 ###
# Yer  :  c1  c2  c3
#    P1:   1   1   0
#    P2:   0   0   0
#    P3:   0   0   0
#    P7:   0   0   1
#    P8:   0   0   1
#    P9:   0   0   2
# M18 -> M24: T4,b
# M26 -> M24: T4,a
# M24 -> M20: T1,a
# M24 -> M11: T1,b
# ESKI Isaretleme M20
# ESKI Isaretleme M11

### Isaretleme M25 ###
# Yer  :  c1  c2  c3
#    P1:   0   1   0
#    P2:   1   0   0
#    P3:   0   0   0
#    P7:   0   0   1
#    P8:   0   0   2
#    P9:   0   0   1
# M15 -> M25: T4,b
# M27 -> M25: T1,a
# M25 -> M7: T1,b
# M25 -> M26: T2,a
# ESKI Isaretleme M7

### Isaretleme M26 ###
# Yer  :  c1  c2  c3
#    P1:   0   1   0
#    P2:   0   0   0
#    P3:   1   0   0
#    P7:   0   0   0
#    P8:   0   0   1
#    P9:   0   0   2
# M25 -> M26: T2,a
# M26 -> M9: T1,b
# M26 -> M24: T4,a
# ESKI Isaretleme M9
# ESKI Isaretleme M24

### Isaretleme M27 ###
# Yer  :  c1  c2  c3
#    P1:   1   1   0
#    P2:   0   0   0
#    P3:   0   0   0
#    P7:   0   0   1
#    P8:   0   0   2
#    P9:   0   0   1
# M13 -> M27: T4,b
# M4 -> M27: T4,a
# M27 -> M25: T1,a
# M27 -> M6: T1,b
# ESKI Isaretleme M25
# ESKI Isaretleme M6
```

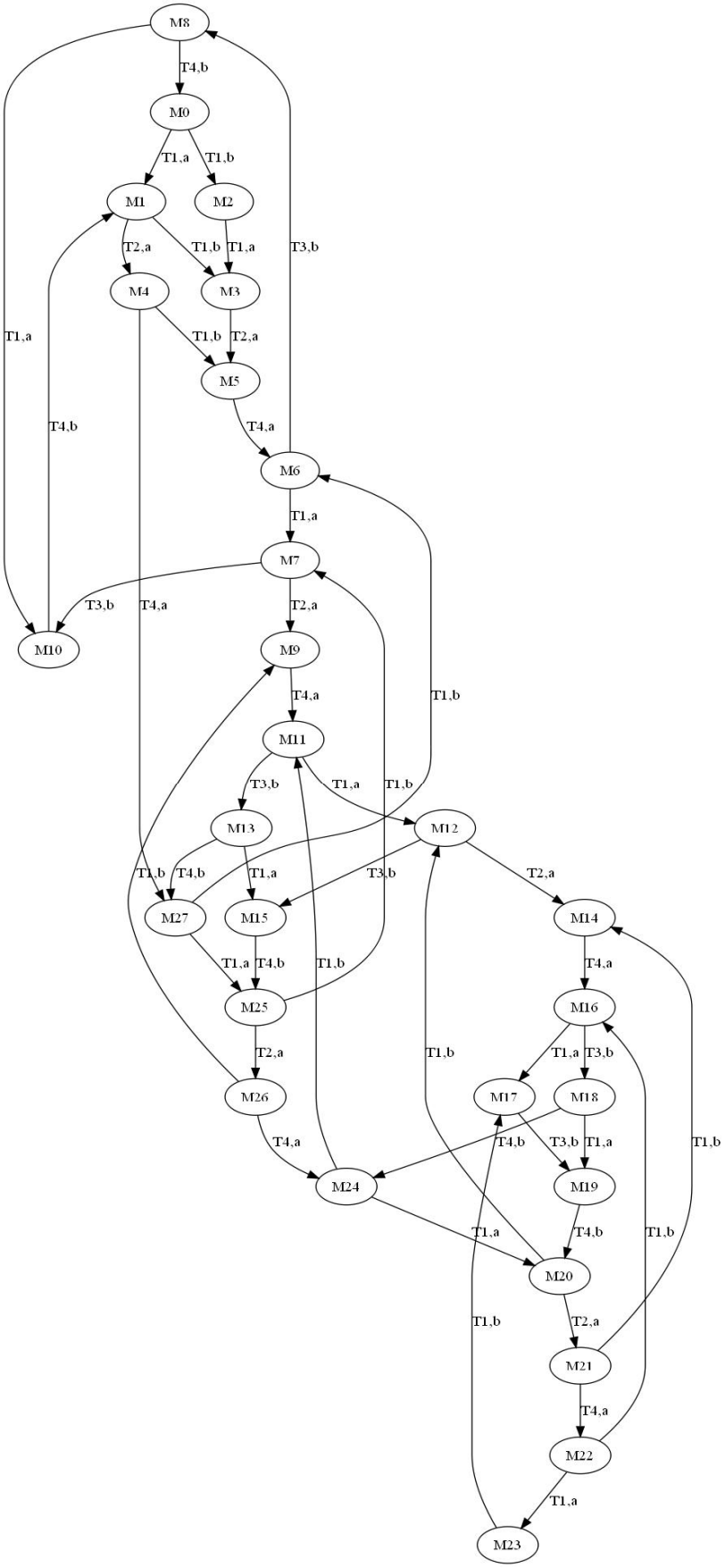
Şekil 2.4 : ÖRNEK-1'in Zamansız Analiz Sonucu Bulunan Erişilebilir Durumları

```
### Aranan Isaretleme 1 ###
# Yer   :  c1  c2  c3
#    P1:   0   0   0
#    P2:   0   1   0
#    P3:   0   0   0
#    P7:   0   0   0
#    P8:   0   0   2
#    P9:   0   0   1
# Istek 1 bulunamadi ###

### Aranan Isaretleme 2 ###
# Yer   :  c1  c2  c3
#    P1:   0   0   0
#    P2:   1   1   0
#    P3:   0   0   0
#    P7:   0   0   1
#    P8:   0   0   1
#    P9:   0   0   2
# ************************************
# Istek 2, M12 isaretlemesinde bulundu ###
# M12 <- M11 <- M9 <- M7 <- M6 <- M5 <- M3 <- M1 <- M0
# M11 -> M12: T1,a
# M9 -> M11: T4,a
# M7 -> M9: T2,a
# M6 -> M7: T1,a
# M5 -> M6: T4,a
# M3 -> M5: T2,a
# M1 -> M3: T1,b
# M0 -> M1: T1,a
### Aranan Isaretleme 3 ###
# Yer   :  c1  c2  c3
#    P1:   1   0   0
#    P2:   0   1   0
#    P3:   0   0   0
#    P7:   0   0   1
#    P8:   0   0   0
#    P9:   0   0   3
# ************************************
# Istek 3, M16 isaretlemesinde bulundu ###
# M16 <- M14 <- M12 <- M11 <- M9 <- M7 <- M6 <- M5 <- M3 <- M1 <- M0
# M14 -> M16: T4,a
# M12 -> M14: T2,a
# M11 -> M12: T1,a
# M9 -> M11: T4,a
# M7 -> M9: T2,a
# M6 -> M7: T1,a
# M5 -> M6: T4,a
# M3 -> M5: T2,a
# M1 -> M3: T1,b
# M0 -> M1: T1,a
```

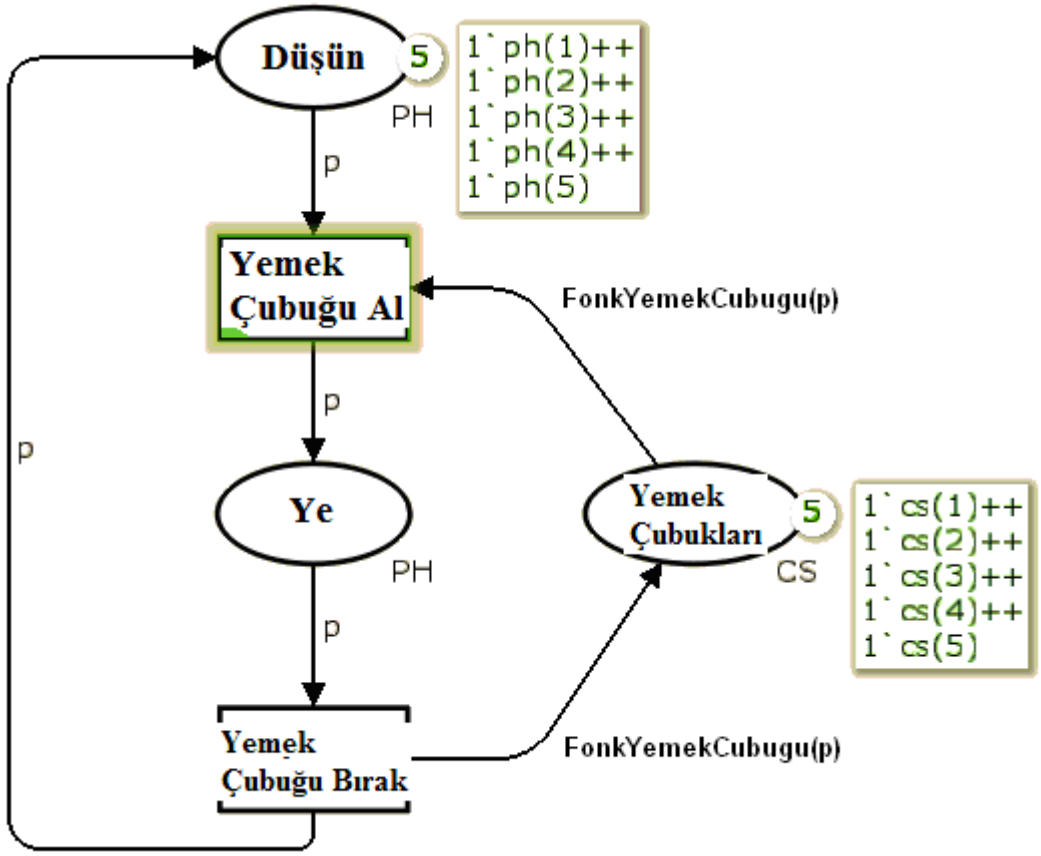
Şekil 2.5 : ÖRNEK-1'in Zamansız Analiz Aranan İşaretleme Sonuçları

Verilen ağa ait Erişilebilirlik Grafiği Şekil 2.6 de verilmiştir.



Şekil 2.6 : ÖRNEK-1'in Zamansız Analiz Erişilebilirlik Grafiği

İkinci örnekte(ÖRNEK–2) Renkli Petri Ağları üzerinedir. Kurt Jensen tarafından geliştirilen CPN-Tool yemek yiyen filozoflar örneğinin geliştirilen yazılım için uyarlanması üzerine bir inceleme yapılacaktır. Bu inceleme sonucunda daha yüksek seviyeden modelleme dilinin kullanılmasının getireceği yarar üzerine durulacaktır. Örneğin CPN-Tool da giriş dosyası Şekil 2.7 deki gibi görsel arayüz ile yapılmaktadır. Görsel arayüzün kullanıcıya getirdiği anlaşılabilir yönünden dolayı CPN-Tool geliştirilen yazılımdan daha üstündür.



Şekil 2.7 : CPN-Tool'un Yemek Yiyen Filozof Örneği (ÖRNEK–2)

Yemek yiyen filozof örneğinde 5 tane filozof vardır. Her filozof yemek yerken iki tane yemek çubuğu kullanmaktadırlar. Yemek yeme işleri bittiğinde yemek çubuklarını tekrar geri bırakmaktadırlar ve tekrar düşünmeye başlıyorlar. CPN-Tool değişkenler ve fonksiyonlar kullanarak ağı tanımlayabilmektedir. Bu özelliğiyle de daha esnek tasarım sağlamaktadır. Örnekte p adlı değişken kullanılaraktan filozof numarası tutulmaktadır. Ayrıca "*FonkYemekCubugu(p)*" fonksiyonuyla da filozofa göre seçilecek yemek çubukları seçilmektedir. Bu fonksiyon "*fun FonkYemekCubugu(p) = 1`cs(p) ++ 1`cs(if p=5 then 1 else p+1);*" şeklinde tanımlıdır. Yani eğer 1 nolu filozof yemek yemeğe gidecekse 1 ve 2 nolu yemek

çubuklarını kullanır. Eğer 2 nolu filozof yemek yiyecekse 2 ve 3 nolu yemek çubuklarını kullanır. Eğer 5 nolu filozof yemek yiyecekse 5 ve 1 nolu yemek çubuklarını kullanacaktır.

ÖRNEK–2'nin geliştiren yazılımla yazılmış giriş dosyasının Şekil 2.8 deki gibidir.

```
# case17_tr.petrinet: CPN-Tool' un Yemek Yiyen Filozof Ornegi
tr
ms 0
sm zz
eg
ay

jeton_rengi ph1 ph2 ph3 ph4 ph5 cs1 cs2 cs3 cs4 cs5

gecis_rengi a(0) b(1) c(2) d(3) e(4)

yer Dusun(1ph1,1ph2,1ph3,1ph4,1ph5) YE KllnYmkCbg(1cs1,1cs2,1cs3,1cs4,1cs5)

gecis YmkCbgAl(0) YmkCbgBirak(1)

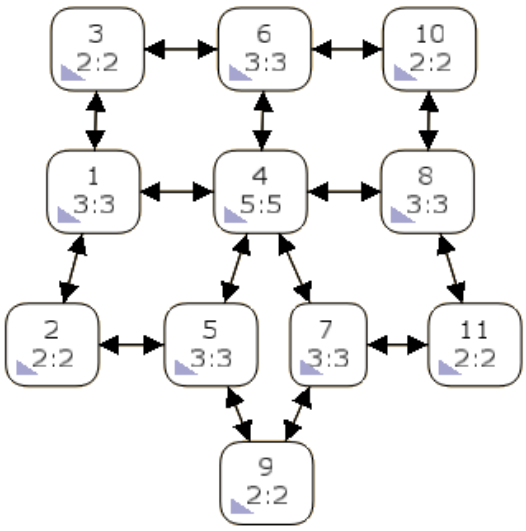
gi YmkCbgAl Dusun a(1ph1) b(1ph2) c(1ph3) d(1ph4) e(1ph5)
gi YmkCbgAl KllnYmkCbg a(1cs1,1cs2) b(1cs2,1cs3) c(1cs3,1cs4) d(1cs4,1cs5) e(1cs5,1cs1)
ci YmkCbgAl YE a(1ph1) b(1ph2) c(1ph3) d(1ph4) e(1ph5)

gi YmkCbgBirak YE a(1ph1) b(1ph2) c(1ph3) d(1ph4) e(1ph5)
ci YmkCbgBirak Dusun a(1ph1) b(1ph2) c(1ph3) d(1ph4) e(1ph5)
ci YmkCbgBirak KllnYmkCbg a(1cs1,1cs2) b(1cs2,1cs3) c(1cs3,1cs4) d(1cs4,1cs5) e(1cs5,1cs1)

ios YmkCbgAl(2a) YmkCbgBirak(2a)
as  YmkCbgAl(3a) YmkCbgBirak(3a)
bs  YmkCbgAl(4a) YmkCbgBirak(4a)
ct  Dusun YE KllnYmkCbg

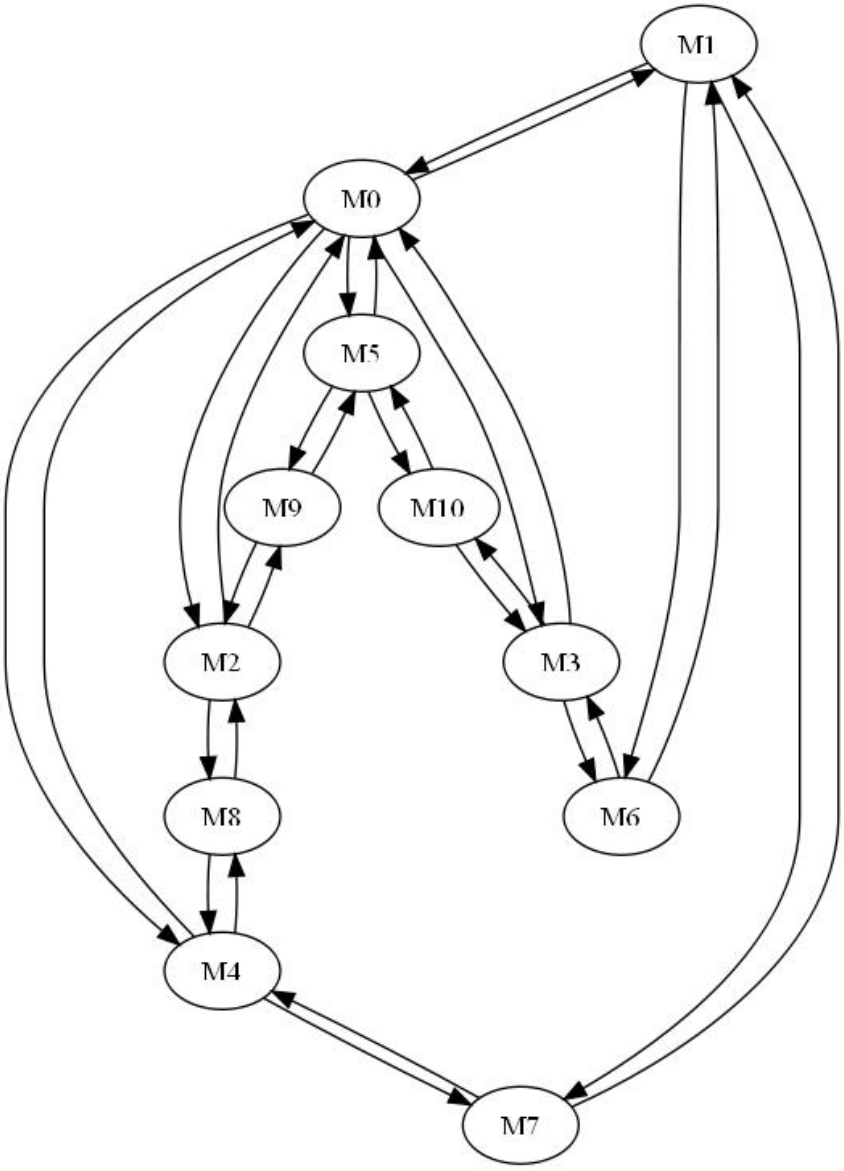
mr Dusun(1ph1,1ph2,1ph3,1ph4,1ph5) YE KllnYmkCbg(1cs1,1cs2,1cs3,1cs4,1cs5)
mr Dusun(1ph1,1ph2,1ph3,1ph4,1ph5) YE KllnYmkCbg(1cs1,1cs2,1cs3,1cs4,1cs5)
```

Şekil 2.8 : Yemek Yiyen Filozof Örneği Giriş Dosyası



Şekil 2.9 : Yemek Yiyen Filozof Örneği CPN-Tool Durum Uzayı

CPN-Tool da Durum Uzayı (*State Space*) analizi yapıldığında 11 farklı işaretle bulunur. Aynı analiz geliştirilen yazılım ile Zamansız Erişilebilirlik Grafiği üretildiğinde de 11 farklı işaretleme elde edilir. Şekil 2.9 da CPN-Tool'un elde ettiği Erişilebilirlik Grafiği gösterilirken Şekil 2.10 da geliştirilen yazılımın elde ettiği Erişilebilirlik Grafiği ve Şekil 2.11 de erişilebilir durumlar görülmektedir.



Şekil 2.10 : Yemek Yiyen Filozof Örneği Erişebilirlik Grafiği

```
### Isaretleme M0 ###
#      Yer  : ph1 ph2 ph3 ph4 ph5 cs1 cs2 cs3 cs4 cs5
#      Dusun:   1   1   1   1   1   0   0   0   0   0
#         YE:   0   0   0   0   0   0   0   0   0   0
# KllnYmkCbg:   0   0   0   0   0   1   1   1   1   1

### Isaretleme M1 ###
#      Yer  : ph1 ph2 ph3 ph4 ph5 cs1 cs2 cs3 cs4 cs5
#      Dusun:   0   1   1   1   1   0   0   0   0   0
#         YE:   1   0   0   0   0   0   0   0   0   0
# KllnYmkCbg:   0   0   0   0   0   0   0   1   1   1

### Isaretleme M2 ###
#      Yer  : ph1 ph2 ph3 ph4 ph5 cs1 cs2 cs3 cs4 cs5
#      Dusun:   1   0   1   1   1   0   0   0   0   0
#         YE:   0   1   0   0   0   0   0   0   0   0
# KllnYmkCbg:   0   0   0   0   0   1   0   0   1   1

### Isaretleme M3 ###
#      Yer  : ph1 ph2 ph3 ph4 ph5 cs1 cs2 cs3 cs4 cs5
#      Dusun:   1   1   0   1   1   0   0   0   0   0
#         YE:   0   0   1   0   0   0   0   0   0   0
# KllnYmkCbg:   0   0   0   0   0   1   1   0   0   1

### Isaretleme M4 ###
#      Yer  : ph1 ph2 ph3 ph4 ph5 cs1 cs2 cs3 cs4 cs5
#      Dusun:   1   1   1   0   1   0   0   0   0   0
#         YE:   0   0   0   1   0   0   0   0   0   0
# KllnYmkCbg:   0   0   0   0   0   1   1   1   0   0

### Isaretleme M5 ###
#      Yer  : ph1 ph2 ph3 ph4 ph5 cs1 cs2 cs3 cs4 cs5
#      Dusun:   1   1   1   1   0   0   0   0   0   0
#         YE:   0   0   0   0   1   0   0   0   0   0
# KllnYmkCbg:   0   0   0   0   0   0   1   1   1   0
### Isaretleme M6 ###
#      Yer  : ph1 ph2 ph3 ph4 ph5 cs1 cs2 cs3 cs4 cs5
#      Dusun:   0   1   0   1   1   0   0   0   0   0
#         YE:   1   0   1   0   0   0   0   0   0   0
# KllnYmkCbg:   0   0   0   0   0   0   0   0   0   1

### Isaretleme M6 ###
#      Yer  : ph1 ph2 ph3 ph4 ph5 cs1 cs2 cs3 cs4 cs5
#      Dusun:   0   1   0   1   1   0   0   0   0   0
#         YE:   1   0   1   0   0   0   0   0   0   0
# KllnYmkCbg:   0   0   0   0   0   0   0   0   0   1

### Isaretleme M7 ###
#      Yer  : ph1 ph2 ph3 ph4 ph5 cs1 cs2 cs3 cs4 cs5
#      Dusun:   0   1   1   0   1   0   0   0   0   0
#         YE:   1   0   0   1   0   0   0   0   0   0
# KllnYmkCbg:   0   0   0   0   0   0   0   1   0   0

### Isaretleme M8 ###
#      Yer  : ph1 ph2 ph3 ph4 ph5 cs1 cs2 cs3 cs4 cs5
#      Dusun:   1   0   1   0   1   0   0   0   0   0
#         YE:   0   1   0   1   0   0   0   0   0   0
# KllnYmkCbg:   0   0   0   0   0   1   0   0   0   0

### Isaretleme M9 ###
#      Yer  : ph1 ph2 ph3 ph4 ph5 cs1 cs2 cs3 cs4 cs5
#      Dusun:   1   0   1   1   0   0   0   0   0   0
#         YE:   0   1   0   0   1   0   0   0   0   0
# KllnYmkCbg:   0   0   0   0   0   0   0   0   1   0

### Isaretleme M10 ###
#      Yer  : ph1 ph2 ph3 ph4 ph5 cs1 cs2 cs3 cs4 cs5
#      Dusun:   1   1   0   1   0   0   0   0   0   0
#         YE:   0   0   1   0   1   0   0   0   0   0
# KllnYmkCbg:   0   0   0   0   0   0   1   0   0   0
```

Şekil 2.11 : Yemek Yiyen Filozof Örneği Erişebilir Durumlar

Şekil 2.1 de verilen Renkli Petri Ağı örneğini zamansız ama erişebilirlik ağacını çıkarmadan incelersek yani çelişki durumlarına öncelik ataması ve paralel işleme izin verilirse kısaca buna koşullu analiz denir. Ağa ait erişilebilir durumlar Şekil 2.12 deki gibi olacaktır. Ayrıca ağa ait erişilebilirlik grafiği Şekil 2.13 deki gibi olacaktır. Giriş dosyasında sadece "eg" operatörünü silinmesi ya da başına yorum karakteri koyup kaydetmek yeterlidir.

```
### Isaretleme M0 ###
# Yer   :  c1  c2  c3
#    P1:   1   1   0
#    P2:   0   0   0
#    P3:   0   0   0
#    P7:   0   0   1
#    P8:   0   0   3
#    P9:   0   0   0
# ILK Isaretleme M0
# M0 -> M1: T1,a // T1,b

### Isaretleme M1 ###
# Yer   :  c1  c2  c3
#    P1:   0   0   0
#    P2:   1   1   0
#    P3:   0   0   0
#    P7:   0   0   1
#    P8:   0   0   3
#    P9:   0   0   0
# M0 -> M1: T1,a // T1,b
# M1 -> M2: T2,a

### Isaretleme M2 ###
# Yer   :  c1  c2  c3
#    P1:   0   0   0
#    P2:   0   1   0
#    P3:   1   0   0
#    P7:   0   0   0
#    P8:   0   0   2
#    P9:   0   0   1
# M1 -> M2: T2,a
# M5 -> M2: T1,b // T2,a
# M2 -> M3: T4,a

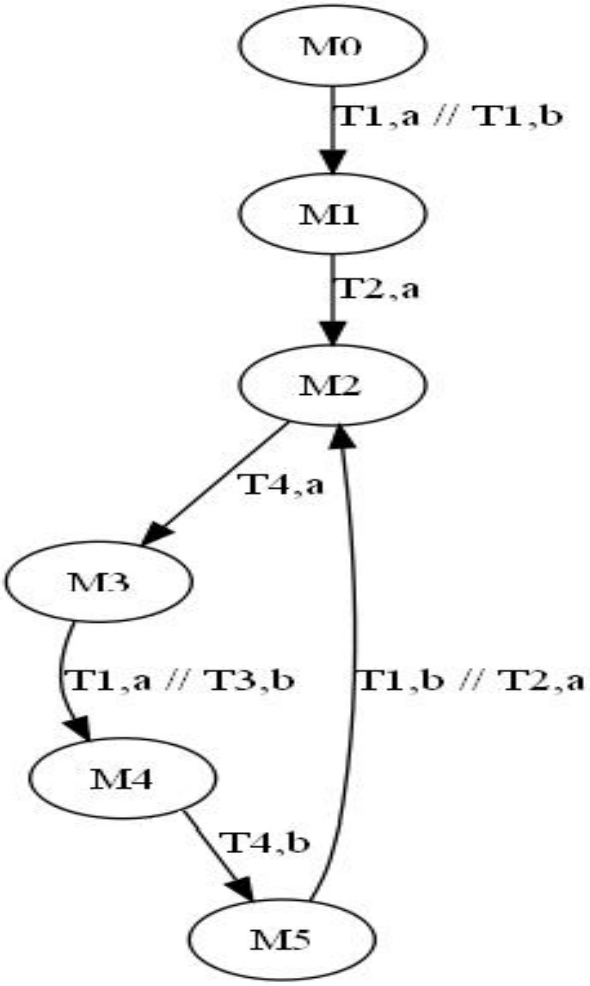
### Isaretleme M3 ###
# Yer   :  c1  c2  c3
#    P1:   1   0   0
#    P2:   0   1   0
#    P3:   0   0   0
#    P7:   0   0   1
#    P8:   0   0   2
#    P9:   0   0   1
# M2 -> M3: T4,a
# M3 -> M4: T1,a // T3,b

### Isaretleme M4 ###
# Yer   :  c1  c2  c3
#    P1:   0   0   0
#    P2:   1   0   0
#    P3:   0   1   0
#    P7:   0   0   0
#    P8:   0   0   3
#    P9:   0   0   0
# M3 -> M4: T1,a // T3,b
# M4 -> M5: T4,b

### Isaretleme M5 ###
# Yer   :  c1  c2  c3
#    P1:   0   1   0
#    P2:   1   0   0
#    P3:   0   0   0
#    P7:   0   0   1
#    P8:   0   0   3
#    P9:   0   0   0
# M4 -> M5: T4,b
# M5 -> M2: T1,b // T2,a
# ESKI Isaretleme M2
```

Şekil 2.12 : Örnek–1 Zamansız Koşullu Analiz Erişilebilir Durumlar

Şekil 2.13 den de görülebileceği gibi paralel işlemler görülmektedir. Başlangıç işaretlemesinde her iki robotta ortak alana girme isteğinde bulunuyorlar böylece T1 geçişi a ve b renkleri için paralel ateşleniyor. Ortak alana hangisin gireceğini yöneten mekanizma T2 geçişinin a rengi için ateşlenmesine izin vererek R1 robotunun ortak alana girmesine izin veriyor. Ortak alana giren R1 robotu işi bitince T4 geçişini a rengi için ateşleyerek ortak alanı terk ediyor ve R2 robotunun ortak alana girmesine izin veriliyor. Bu sırada R1 robotu tekrar ortak alana girmek istiyor ve T1 geçişi a rengi içinde tekrar ateşleniyor. Sonra R2 robotu da ortak alanı terk ediyor ve yeniden ortak alana girme isteği gönderirken R1 robotu ortak alana giriyor. Bu süreç verilen ilk işaretleme sayesinde bu şekilde devam etmektedir.



Şekil 2.13 : Örnek–1 Zamansız Koşullu Analiz Erişilebilirlik Grafiği

ÖRNEK–3 olarak ÖRNEK–1 de verilen Renkli Petri Ağ modeli kullanılacaktır. ÖRNEK–1 de verilen örneği Belirli Zamanlı Petri Ağlarını incelemek için giriş dosyasında ufak birkaç değişiklik yapılması yeterlidir. Bunun için "*sistem_modu*" parametresinin parametresi *"belirli_zamanli"* olarak değiştirilmesi yeterlidir. Analiz sonucunda elde edilen erişilebilir durumların sayısı ve zamanlamayı gösteren çıkış dosyası Şekil 2.14 dedir.

```
### Isaretleme M0 ### [0.000, 2.000) GERCEK
# Yer   :  c1  c2  c3
#    P1:   1   1   0
#    P2:   0   0   0
#    P3:   0   0   0
#    P7:   0   0   1
#    P8:   0   0   3
#    P9:   0   0   0
# ILK Isaretleme M0
# M0 -> M1: T1,a // T1,b
# Gecislerin DURUMLARI zaman=0.000 iken: T1,a,et,2.000 & T1,b,et,2.000

### Isaretleme M1 ### [2.000, 5.000) GERCEK
# Yer   :  c1  c2  c3
#    P1:   0   0   0
#    P2:   0   0   0
#    P3:   0   0   0
#    P7:   0   0   1
#    P8:   0   0   3
#    P9:   0   0   0
# M0 -> M1: T1,a // T1,b
# M1 -> M2: T1,a // T1,b // T2,a
# Gecislerin DURUMLARI zaman=2.000 iken: T1,a,ft,3.000 & T1,b,ft,3.000

### Isaretleme M2 ### [5.000, 7.000) GERCEK
# Yer   :  c1  c2  c3
#    P1:   0   0   0
#    P2:   1   1   0
#    P3:   0   0   0
#    P7:   0   0   1
#    P8:   0   0   3
#    P9:   0   0   0
# M1 -> M2: T1,a // T1,b // T2,a
# M2 -> M3: T1,a // T1,b // T2,a
# Gecislerin DURUMLARI zaman=5.000 iken: T1,a,wt,4.000 & T1,b,wt,4.000 & T2,a,et,2.000

### Isaretleme M3 ### [7.000, 10.000) GERCEK
# Yer   :  c1  c2  c3
#    P1:   0   0   0
#    P2:   0   1   0
#    P3:   0   0   0
#    P7:   0   0   0
#    P8:   0   0   2
#    P9:   0   0   0
# M2 -> M3: T1,a // T1,b // T2,a
# M3 -> M4: T2,a // T4,a
# Gecislerin DURUMLARI zaman=7.000 iken: T1,a,wt,2.000 & T1,b,wt,2.000 & T2,a,ft,3.000

### Isaretleme M4 ### [10.000, 12.000) GERCEK
# Yer   :  c1  c2  c3
#    P1:   0   0   0
#    P2:   0   1   0
#    P3:   1   0   0
#    P7:   0   0   0
#    P8:   0   0   2
#    P9:   0   0   1
# M3 -> M4: T2,a // T4,a
# M4 -> M5: T2,a // T4,a
# Gecislerin DURUMLARI zaman=10.000 iken: T2,a,wt,4.000 & T4,a,et,2.000

### Isaretleme M5 ### [12.000, 15.000) GERCEK
# Yer   :  c1  c2  c3
#    P1:   0   0   0
#    P2:   0   1   0
#    P3:   0   0   0
#    P7:   0   0   0
#    P8:   0   0   2
#    P9:   0   0   1
# M4 -> M5: T2,a // T4,a
# M5 -> M6: T4,a // T1,a // T3,b
# Gecislerin DURUMLARI zaman=12.000 iken: T2,a,wt,2.000 & T4,a,ft,3.000
```

```
### Isaretleme M6 ### [15.000, 17.000) GERCEK
# Yer  :  c1  c2  c3
#    P1:   1   0   0
#    P2:   0   1   0
#    P3:   0   0   0
#    P7:   0   0   1
#    P8:   0   0   2
#    P9:   0   0   1
# M5 -> M6: T4,a // T1,a // T3,b
# M13 -> M6: T4,a // T1,a // T3,b
# M6 -> M7: T1,a // T3,b // T4,a
# Gecislerin DURUMLARI zaman=15.000 iken: T4,a,wt,4.000 & T1,a,et,2.000 & T3,b,et,2.000

### Isaretleme M7 ### [17.000, 20.000) GERCEK
# Yer  :  c1  c2  c3
#    P1:   0   0   0
#    P2:   0   0   0
#    P3:   0   0   0
#    P7:   0   0   0
#    P8:   0   0   2
#    P9:   0   0   0
# M6 -> M7: T1,a // T3,b // T4,a
# M7 -> M8: T1,a // T3,b // T4,b
# Gecislerin DURUMLARI zaman=17.000 iken: T1,a,ft,3.000 & T3,b,ft,3.000 & T4,a,wt,2.000

### Isaretleme M8 ### [20.000, 22.000) GERCEK
# Yer  :  c1  c2  c3
#    P1:   0   0   0
#    P2:   1   0   0
#    P3:   0   1   0
#    P7:   0   0   0
#    P8:   0   0   3
#    P9:   0   0   0
# M7 -> M8: T1,a // T3,b // T4,b
# M8 -> M9: T1,a // T3,b // T4,b
# Gecislerin DURUMLARI zaman=20.000 iken: T1,a,wt,4.000 & T3,b,wt,4.000 & T4,b,et,2.000
### Isaretleme M9 ### [22.000, 25.000) GERCEK
# Yer  :  c1  c2  c3
#    P1:   0   0   0
#    P2:   1   0   0
#    P3:   0   0   0
#    P7:   0   0   0
#    P8:   0   0   3
#    P9:   0   0   0
# M8 -> M9: T1,a // T3,b // T4,b
# M9 -> M10: T4,b // T1,b // T2,a
# Gecislerin DURUMLARI zaman=22.000 iken: T1,a,wt,2.000 & T3,b,wt,2.000 & T4,b,ft,3.000

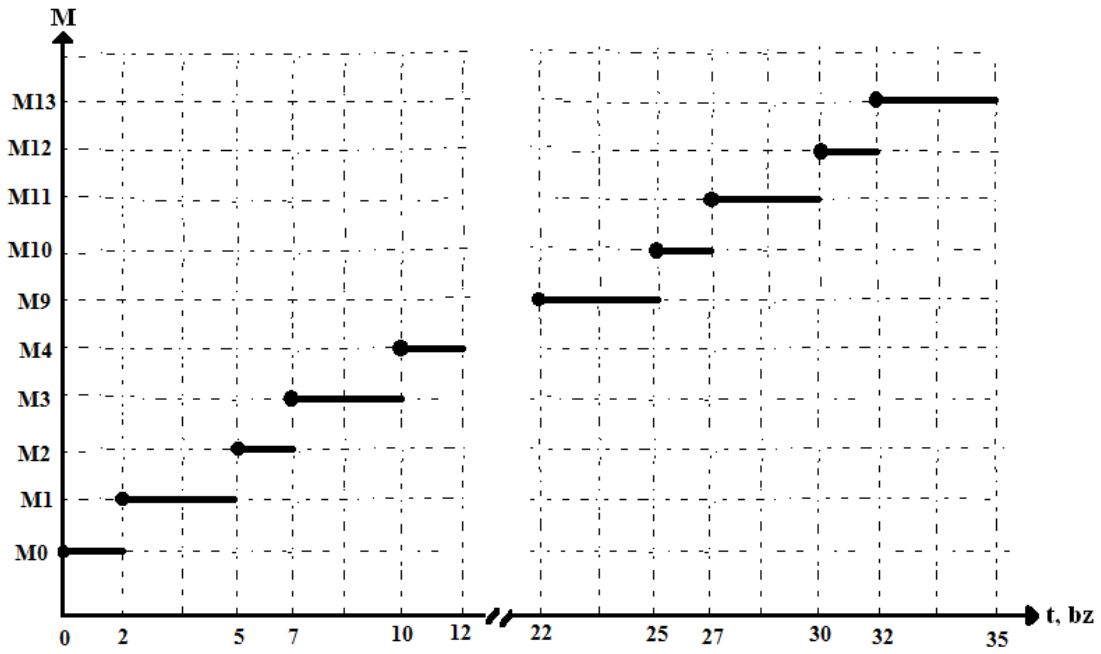
### Isaretleme M10 ### [25.000, 27.000) GERCEK
# Yer  :  c1  c2  c3
#    P1:   0   1   0
#    P2:   1   0   0
#    P3:   0   0   0
#    P7:   0   0   1
#    P8:   0   0   3
#    P9:   0   0   0
# M9 -> M10: T4,b // T1,b // T2,a
# M10 -> M11: T1,b // T2,a // T4,b
# Gecislerin DURUMLARI zaman=25.000 iken: T4,b,wt,4.000 & T1,b,et,2.000 & T2,a,et,2.000
### Isaretleme M11 ### [27.000, 30.000) GERCEK
# Yer  :  c1  c2  c3
#    P1:   0   0   0
#    P2:   0   0   0
#    P3:   0   0   0
#    P7:   0   0   0
#    P8:   0   0   2
#    P9:   0   0   0
# M10 -> M11: T1,b // T2,a // T4,b
# M11 -> M12: T1,b // T2,a // T4,a
# Gecislerin DURUMLARI zaman=27.000 iken: T1,b,ft,3.000 & T2,a,ft,3.000 & T4,b,wt,2.000
```

```
### Isaretleme M12 ### [30.000, 32.000) GERCEK
# Yer   :  c1  c2  c3
#     P1:   0   0   0
#     P2:   0   1   0
#     P3:   1   0   0
#     P7:   0   0   0
#     P8:   0   0   2
#     P9:   0   0   1
# M11 -> M12: T1,b // T2,a // T4,a
# M12 -> M13: T1,b // T2,a // T4,a
# Gecislerin DURUMLARI zaman=30.000 iken: T1,b,wt,4.000 & T2,a,wt,4.000 & T4,a,et,2.000

### Isaretleme M13 ### [32.000, 35.000) GERCEK
# Yer   :  c1  c2  c3
#     P1:   0   0   0
#     P2:   0   1   0
#     P3:   0   0   0
#     P7:   0   0   0
#     P8:   0   0   2
#     P9:   0   0   1
# M12 -> M13: T1,b // T2,a // T4,a
# M13 -> M6: T4,a // T1,a // T3,b
# Gecislerin DURUMLARI zaman=32.000 iken: T1,b,wt,2.000 & T2,a,wt,2.000 & T4,a,ft,3.000
# ESKI Isaretleme M6
```

Şekil 2.14 : Örnek–1'in Belirli Zamanlı Analizi Erişilebilir Durumları

ÖRNEK-1'in işaretlemelerin zamanla değişim grafiği Şekil 2.15'deki gibidir.



Şekil 2.15 : Örnek–1'in Belirli Zamanlı Analizi Erişilebilirlik Diyagramı

ÖRNEK–4 olarak ÖRNEK–1 de verilen Renkli Petri Ağ modeli kullanılacaktır. ÖRNEK–1 de verilen örnek üzerinden Rastlantısal Zamanlı Petri Ağlarını incelemek için giriş dosyasında ufak değişiklik yapılması yeterlidir. Bunun için "*sistem_modu*" parametresinin parametresi "*rastlantisal_zamanli*" olarak değiştirilmesi yeterlidir. Analiz sonucunda elde edilen erişilebilir durumlar her analiz sonucunda değişecektir. Bunun sebebi rastlantısal seçilen zaman bilgilerinin değişimine göre sistemin

davranışının değişmesidir. Bu nedenle Rastlantısal Petri Ağlarında performans analizi ortalama değerler üzerinden gösterilecektir. Program 4 kere çalıştırıldığında 1.çalıştırmada 120 işaretleme, 2.çalıştırmada 141 işaretleme, 3.çalıştırmada 66 işaretleme, 4.çalıştırmada 195 işaretleme bulunmuştur. Her analiz için bulunan ortalama zaman değerleri Çizelge 2.3 de verilmiştir.

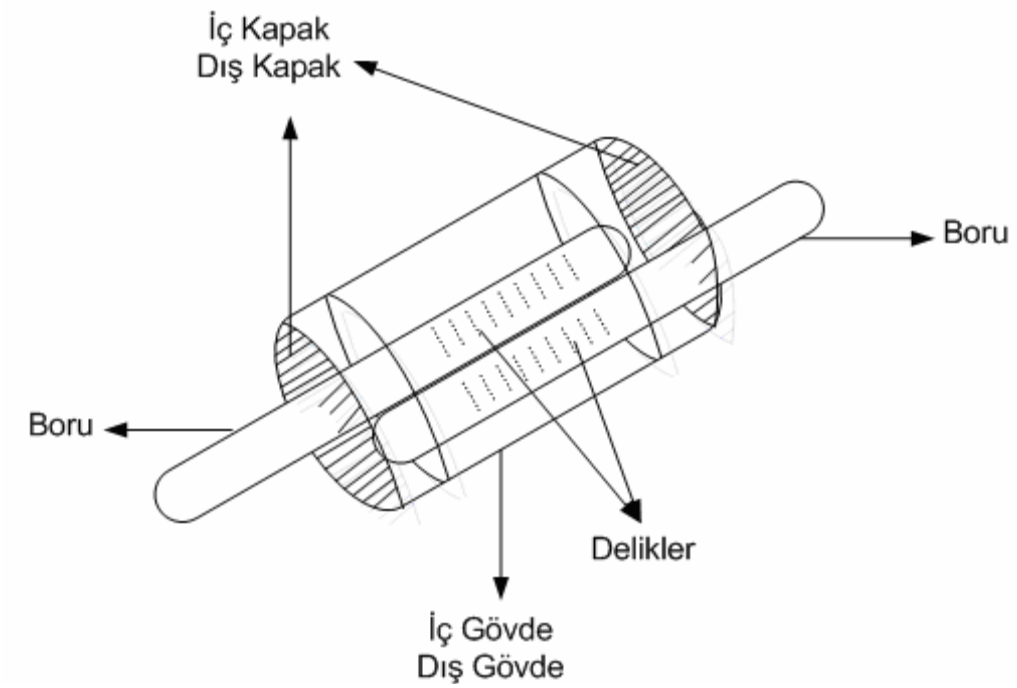
Çizelge 2.3 : ÖRNEK-1'in Rastlantısal Değişkenli Analiz Sonucu Ortalama Zamanlar

	1. Çalışma	**2.çalışma**	**3.çalışma**	**4.çalışma**
T1,a et süresi	1.750	3.519	2.125	2.047
T1,b et süresi	1.722	2.365	3.000	1.111
T1,a ft süresi	2.327	1.617	0.893	2.574
T1,b ft süresi	1.375	0.875	4.000	1.469
T1,a wt süresi	2.667	2.267	2.150	3.150
T1,b wt süresi	1.156	2.769	1.400	2.406
T2,a et süresi	1.563	2.367	1.964	2.275
T2,a ft süresi	1.306	1.438	1.000	2.158
T2,a wt süresi	0.932	2.538	0.875	2.158
T3,b et süresi	1.875	2.023	2.821	1.906
T3,b ft süresi	2.000	3.075	2.300	1.800
T3,b wt süresi	1.775	0.981	2.375	2.882
T4,a et süresi	3.159	2.025	1.643	1.912
T4,b et süresi	3.861	1.827	3.188	2.200
T4,a ft süresi	1.545	1.500	0.393	1.575

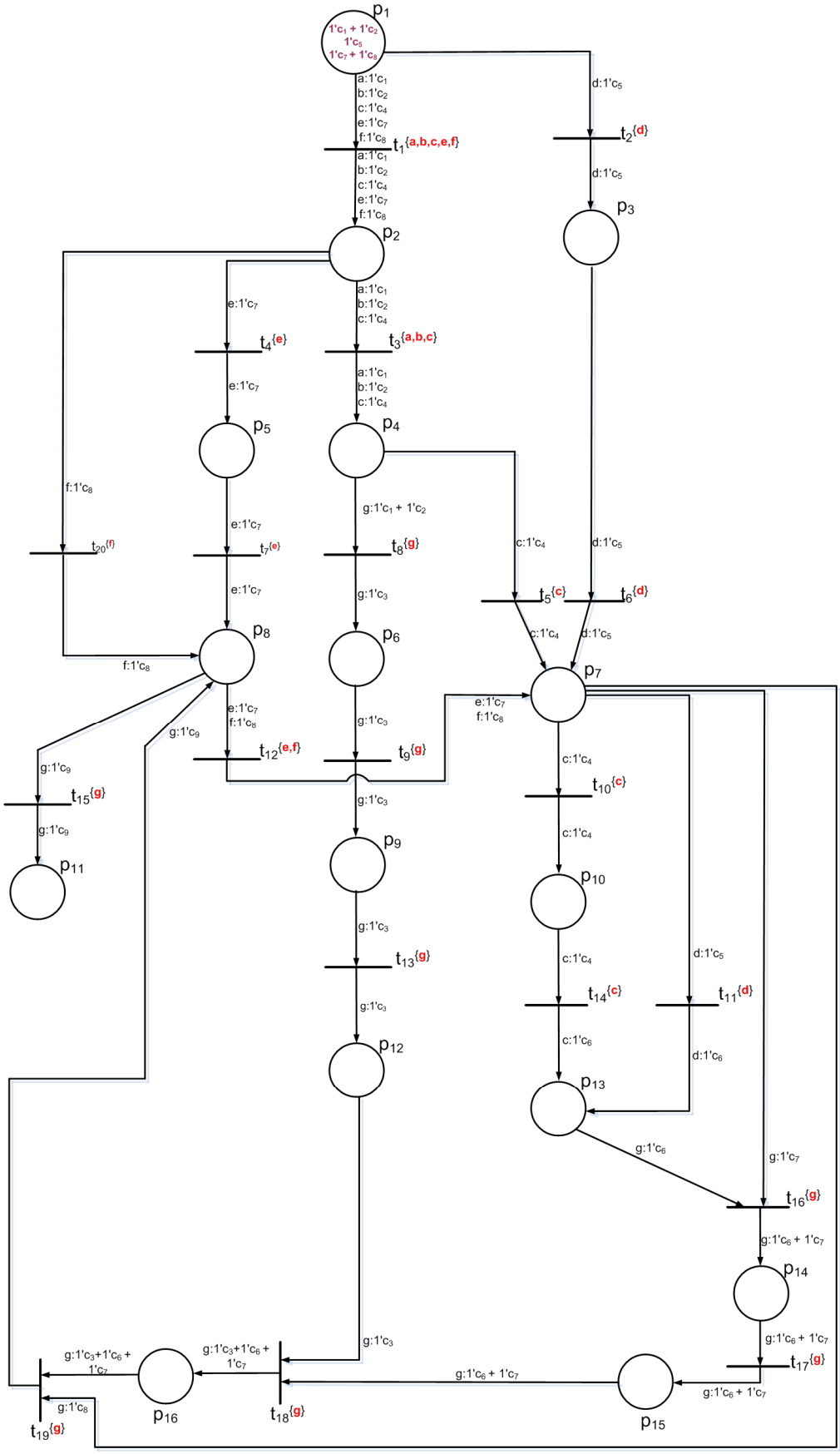
T4,b ft süresi	2.194	2.568	0.938	2.200
T4,a wt süresi	2.438	2.107	2.679	1.855
T4,b wt süresi	1.625	1.712	3.583	2.191

İşaretlemelerin sayısından da görüldüğü gibi üretilen sistemin zamanın değişimine bağlılığı fazladır. Üretim bandında işlerin beklenen değerlerden başka kayması sistemin çalışmasına önemli etkide bulunmaktadır. Markovian analizi de yapıldığında sistemin kararlı durumlarına ne kadar olasılıkla gelebildiği görüleceğinden sistem hakkında yapılacak yorumlar daha doğru olacaktır.

ÖRNEK–5 olarak Susturucu Egzoz üretim sisteminin analizi yapılacaktır.[2] Bu örneğin seçilmesinin sebebi Y. Serkan tarafından gerçeklenen CPN-RPA v1 yazılımı tarafından da analiz edilip zaman performans değerleri verilmiş olmasıdır. Geliştirilen yazılım ile CPN-RPA yazılımı arasında analiz süresi karşılaştırması yapılıp geliştirilen tolun 2,5 kat daha hızlı çalıştığını gösterilecektir. Üretilecek malzeme Şekil 2.16 deki gibidir.



Şekil 2.16 : Egzoz Bileşenleri



Şekil 2.17 : Susturucu Egzoz Üretim Aşamaları RPA Modeli

Çizelge 2.4 : Susturucu Egzoz Üretim Modelinin Bileşenleri ve Tanımları

P1	Ham malzemeler mevcut ve beklemede
P2	Şerit kesildi
P3	Kesim yapıldı
P4	Boy kesildi
P5	Sık delik kesildi
P6	Puntolama bitti
P7	Delik delindi
P8	Kapak sıvandı
P9	Marka basma ve köşe kesim işlemi bitti
P10	Uç açma kapama bitti
P11	Ürün monte edildi, son
P12	Gövde şekil verme bitti
P13	İç boru hazır
P14	İç eleman monte edildi
P15	İç eleman kaynak yapıldı
P16	
T1	Şerit kesim işlemi
T2	Kesim işlemi (iç boru kesim işlemi)
T3	Boy kesim işlemi
T4	Sık delik delme işlemi

T5	Delik delme işlemi
T6	Delik delme işlemi
T7	İç Kapak sıvama
T8	Puntolama işlemi
T9	Marka basma ve köşe kesim işlemi
T10	Uç açma kapama işlemi
T11	Büküm ve uç şekillendirme işlemi
T12	Delik delme işlemi
T13	Gövde şekil verme işlemi
T14	İç eleman monte edildi
T15	Kapak sıvaması işlemi
T16	İç eleman montaj işlemi
T17	İç eleman kaynak
T18	İç eleman yerleştirme - ABS malzeme doldurma ve iç eleman ve gövde kaynak İşlemi
T19	Kapak sıvaması işlemi
T20	Dış Kapak sıvama
c1	Dış gövde
c2	İç gövde
c3	Hazır olan dış veya iç gövde
c4	Sac iç boru

c5	Boru (plastik) iç boru
c6	Hazır olan iç boru (boru veya sac)
c7	İç kapak
c8	Dış kapak
c9	Hazır olan birleştirilmiş ürün
a	Dış gövde için
b	İç gövde için
c	Sac iç boru için
d	Boru (plastik) iç boru için
e	İç kapı için
f	Dış kapı için
g	Hazır olan parçalar için

Susturucu örneğine ait giriş dosyası Şekil 2.18 deki gibidir. CPN-RPA programının giriş dosyalarına uygun şekilde ağda değişikliğe gidilmiştir. Ağın RPA modeli de CPN-RPA yazılımının giriş dosyalarına göre düzenlenmiştir. Amaç her iki araç arasındaki performansı ölçmektir.

CPN-RPA v1.0 yazılımı ağı analiz ettiğinde 1062 tane durum bulmaktadır. Aynı ağı geliştirilen yazılım analiz ettiğinde de 1062 tane durum bulmaktadır. Aranan işaretlemeyi CPN-RPA 1.0 yazılımı 1586. işaretlemede bulurken geliştirilen yazılım 681. işaretlemede bulmaktadır. Bu farklılıkta her iki ağı analiz yaklaşımlarımızın farklılığını gösterebilir. En dikkate değer fark ise ağın analizinin tamamlanma süreleridir. CPN-RPA yazılımı ağı 7,96 saniyede analiz edebilirken bu süre geliştirilen yazılım için 3,3 saniye olmaktadır. Bu da geliştirilen yazılımın 2,5 kat daha hızlı çalışabildiğinin göstergesidir.

```
# case23_tr.petrinet: Serkan'ın Tezindeki Örnek. Egzoz Uretim Asamalari
tr
ms 0
ay
sm zz
eg

jeton_rengi c1 c2 c3 c4 c5 c6 c7 c8 c9
gecis_rengi a(0) b(1) c(2) d(3) e(4) f(5) g(6)
yer P1(1c1,1c2,1c5,1c7,1c8) P2 P3 P4 P5 P6 P7 P8 P9 P10
yer P11 P12 P13 P14 P15 P16
gecis T1(0) T2(1) T3(2) T4(3) T5(4) T6(5) T7(6) T8(7) T9(8) T10(9) T11(10)
gecis T12(11) T13(12) T14(13) T15(14) T16(15) T17(16) T18(17) T19(18) T20(19)

gi T1 P1 a(1c1) b(1c2) c(1c4) e(1c7) f(1c8)
ci T1 P2 a(1c1) b(1c2) c(1c4) e(1c7) f(1c8)
gi T2 P1 d(1c5)
ci T2 P3 d(1c5)
gi T3 P2 a(1c1) b(1c2) c(1c4)
ci T3 P4 a(1c1) b(1c2) c(1c4)
gi T4 P2 e(1c7)
ci T4 P5 e(1c7)
gi T5 P4 c(1c4)
ci T5 P7 c(1c4)
gi T6 P3 d(1c5)
ci T6 P7 d(1c5)
gi T7 P5 e(1c7)
ci T7 P8 e(1c7)
gi T8 P4 g(1c1,1c2)
ci T8 P6 g(1c3)
gi T9 P6 g(1c3)
ci T9 P9 g(1c3)
gi T10 P7 c(1c4)
ci T10 P10 c(1c4)
gi T11 P7 d(1c5)
ci T11 P13 d(1c6)
gi T12 P8 e(1c7) f(1c8)
ci T12 P7 e(1c7) f(1c8)
gi T13 P9 g(1c3)
ci T13 P12 g(1c3)
gi T14 P10 c(1c4)
ci T14 P13 c(1c6)
gi T15 P8 g(1c9)
ci T15 P11 g(1c9)
gi T16 P13 g(1c6)
gi T16 P7 g(1c7)
ci T16 P14 g(1c6,1c7)
gi T17 P14 g(1c6,1c7)
ci T17 P15 g(1c6,1c7)
gi T18 P15 g(1c6,1c7)
gi T18 P12 g(1c3)
ci T18 P16 g(1c3,1c6,1c7)
gi T19 P16 g(1c3,1c6,1c7)
gi T19 P7 g(1c8)
ci T19 P8 g(1c9)
gi T20 P2 f(1c8)
ci T20 P8 f(1c8)

#or T1 T2 T3 T4 T5 T6 T7 T8 T9 T10 T11 T12 T13 T14 T15 T16 T17 T18 T19 T20
#bz 0.25
ai P1(1c2) P2(1c7) P3 P4(1c1) P5 P6 P7 P8(1c8) P9 P10 P11 P12 P13(1c6) P14 P15 P16
```

Şekil 2.18 : Susturucu Egzoz Örneği Giriş Dosyası

3. SONUÇ VE ÖNERİLER

Bu tez çalışması kapsamında görsel gösterimi ve matematiksel temelleri olan Petri Ağlarıyla modelleme incelenmiştir. İlk geliştirilen Klasik Petri Ağlarından başlanarak Zamanlı Petri Ağları'nın belirli ve rastlantısal zamanlı değişimleri incelenmiştir. Ayrıca Renkli Petri Ağları da incelendikten sonra Zamanlı ve Renkli Petri Ağların birleşimi olan Zamanlı Renkli Petri Ağları incelenmiştir. Zamanlı Renkli Petri Ağlarını analiz edebilen algoritma verildikten sonra bu algoritmayı kullanan "Çok Fonksiyonel Zamanlı Renkli Petri Ağları Yazılımı" yazılımla ilgili önekler verilmiştir. Benzeri yazılımlara göre 2,5 kat daha hızlı çalışabildiği gösterilmiştir.

Petri ağları ile ilgili birçok yöntem, araç ve algoritma geliştirilmiştir. Ancak bu yöntemler henüz standartlaşması tamamlanamamıştır. Bu yüzden Petri ağlarının elektronik sistemlerin tasarımında kullanımına yönelik çalışmalar yapılabilir. Petri Ağları üzerine geliştiren birçok matematiksel model olmasına rağmen tümünü analiz edebilen bilgisayar destekli yazılım çok fazla bulunmamaktadır. Geliştirilen yazılım sayesinde bu alandaki eksikliği gidermek için literatüre katkı sağlanmıştır.

Bu çalışma doğrultusunda ilerideki çalışmalarda geliştirilen yazılım da gelişmiş özellikli ve esnek bir yapıya kavuşturulabilinir. Yazılımın giriş dosyası formatı daha esnek bir yapıya kavuşturulabilir. Mevcut yapıda giriş dosyası hazırlanırken uyulması gereken sıralama vardır. Örneğin kullanıcı jetonları tanımlamadan jetonların kullanıldığı yer tanımı veya geçişlerin giriş ve çıkışlarını tanımlama gibi işlemleri daha önceden yapamaz. Bunun sebebi giriş dosyasından okunan değerler C dilindeki yapılarda tutulmaktadır. Yapılar kendi içinde başka yapılara referans ettiğinden tanımlanmamış bir yapının elemanı olan bir yapıya referans edilmeye çalışmasından ötürü programda hata oluşmaktadır. Ayrıca kullanıcının giriş dosyasını hazırlarken yaptığı hatalarda bilgilendirme yapabilecek şekilde değişikliğe gidilebilir. Yazılım bu hataları sezemediğinden hiç çalışmayabiliyor ya da hatalı çalışabilmektedir. Dosya kullanılmadan önce analiz edilerekten kullanıcı bu konuda hata mesajı verilebilir.

Petri Ağları görsel bir modelleme dilidir. Giriş dosyasını birçok yazılım "Nesneye Dayalı Programlama" kullanarak yapmaktadır. Yazılımın giriş dosyasını görsel bir arayüzden alması sağlanabilecek şekilde geliştirme yapılabilir.

Mevcut yazılımda geçişler sadece bir kez ateşlenebilirdir. Tez içerisinde de bahsedilen çoklu ateşlenme özelliğini desteklememektedir. Geçişlere katılacak yeni bir parametre sayesinde ne tip davranacağı kullanıcının kontrolüne bırakılabilir. Böylece geliştirilecek yazılım daha fazla ağ modellini destekleyebilecek özelliğe kavuşacaktır.

Zamanlı Petri Ağları'nda atomik ateşlenmeli geçişlerin olduğu Genelleştirilmiş Zamanlı Petri Ağları'nı da modelleyebilecek şekilde yazılımda değişikliğe gidilebilir.

Geçişlerin giriş koşulları VE işlemine tabi tutulan yapı seçilmiştir. Ancak VEYA işlemi de seçilebilecek şekilde yapısal değişikliğe gidilebilir. Hatta kullanıcının seçimine bırakılabilir. Bu özelliğinde eklenmesiyle daha değişik modellerde incelenebilir.

Yazılımda yapılacak eklemelerle ve görsel arayüz sayesinde gerçek zamanlı benzetim yapılarak birim zaman yerine gerçek zamanda analiz yapılabilir. Böylece tüm sistemin analizi yapılmadan adım adım sistemin davranışı incelenebilir.

Rastlantısal Petri Ağları için Markovian Sürecine dayalı ağın analizi yapılaraktan sistemin kararlı işaretlemelerinin olasılık dağılımı hesaplama gibi performans ölçütleri elde edilebilir.

Mevcut yazılımda sonsuz ateşlenme ve ağdaki jeton birikmesi gibi yazılımı olumsuz etkileyecek durumlar için önlemler alınmamıştır. Bir geçiş sonsuza kadar ateşlenebilir ve her ateşlemesi yeni bir işaretlemeye sebep olacağından yazılım bu geçiş için sonsuz ateşlenmenin söz konusu olduğunu anlayamayacağından devamlı çalışacak ve bellek tüketimine sebep olacaktır. İşaretlemenin değişimi takip edilebilineceği gibi geçişlere maksimum ateşlenebilme sayısı verilerekten kontrol sağlanabilir. Ayrıca ağdaki bir geçiş ağda devamlı ateşlenmesinden dolayı jeton birikmesine sebep olabileceğinden ağın sınırlı kalıp kalmadığına da bakılabilir.

Verilen öneriler sayesinde yazılım daha kararlı daha esnek ve daha çok ağ modelini modelleyebilir bir yapıya kavuşturulabilir.

KAYNAKLAR

[1] **Petri C.A., 1962.** *Kommunikation mit automaten*, PhD Thesis, University of Bonn, West Germany, (1962).

[2] **Yorulmaz, Serkan, 2009.** *Renkli Petri Ağlarında Erişilebilirlik Ağacı ve P-Değişmezleri Analizi,* Yüksek Lisans Tezi, İstanbul/Türkiye.

[3] **Yılmaz, Ahmet, 2009.** *Zamanlanmış Petri Ağlarında Erişilebilirlik Analizi,* Yüksek Lisans Tezi, İstanbul/Türkiye.

[4] **Ramchandani C., 1974.** *Analysis of asynchronous concurrent systems by Petri nets,* PhD Thesis, M.I.T., Cambridge.

[5] **Coolahan J. and Roussopoulos J., 1983**. *Timing requirements for time driven systems using augmented Petri nets*, IEEE Transaction on Software Engineering, SE-9(5).

[6] **Zhu J. and Denton R., 1988**. *Timed Petri nets their application to communication protocol specification,* Proceedings of the 21 st IEEE Military Communication Conference, San Diego, 195-198

[7] **Yurttakal, Oktay, 2009.** *Zamanlanmış Petri Ağlarında Ateşlenebilirlik Problemlerinin İncelenmesi.* Yüksek Lisans Tezi, İstanbul/Türkiye.

[8] **Merlin P. And Farber D., 1976.** *Recoverability of communicaiton protocols – Implication of a theoretical study*, IEEE Transaction on Communications, 1036-1043

[9] **Natkin S., 1980.** Les reseaux de Petri stochastiques et leur application a I'evaluation des systems informatiques, These de Docteur Ingegneur, Paris, France

[10] **Molloy M., 1981.** *On the integration of delay and throuhgtput measures in distributed processing models*, PhD thesis, Los Angles

[11] **Marsan A. M., Balbo G., Conte G., 1984.** *A Class of Generalized Stochastic Petri Nets for the Performance Evaluation of Multiprocessor Systems*, ACM Transactions on Computer Systems, Vol.2, No.2, p 93-122.

[12] **Bowden F., 2001.** *The modelling and Analysis of Command and Control Decision Processes using Extended Time Petri Nets,* The University of Adelaide, Departmant of Applied Mathematics, PhD thesis.

[13] **Ergan Z.H., 2004**. *Zamanlandırılmış Petri Ağlarında Sistem Çıkmazı Analizi* Yüksek Lisans Tezi, Anadolu Üniversitesi, Fen Bilimleri Enstitüsü.

[14] **Jensen, K., 1993.** *An Introduction to the Theoretical Aspects of Coloured Petri Nets*, Lecture Notes In Computer Science; Vol. 803, Springer-Verlag London, UK.

[15] **Çayır, Sinan, 2004.** *Petri Ağlarında Değişmez Analizi.* Yüksek Lisans Tezi, İstanbul/Türkiye.

[16] **Uzam, Murat, 1998.** *Petri-Net-Based Supervisory Control of Discrete Event Systems and Their Ladder Logic Diagram Implementations,* Doktora Tezi, Telford Research Institute University of Salford, Salford, UK

[17] **Wang J., 1998**. *TIMED PETRI NETS Theory and Application*, Kluwer Academic Publlishers, Boston/Dordrecht/London.

[18] **Wikipedia**, http://en.wikipedia.org/wiki/Exponential_distribution

[19] **Wang J.** *Petri Nets for Dynamic Event-Driven System Modeling,* Handbook of Dynamic System Modeling, Ed: Paul Fishwick, CRC Press, 2007.

[20] **Marsan, M. Ajmone, Conte, G., Donatelli, S., Franceschinis, G., 1994.** *MODELLING WITH GENERALISED STOCHASTIC PETRI NETS,* Università degli Studi di Torino.

[21] **Zurawski R. and Zhou M.C., 1994.** *Nets and Industrial Applications: A Tutorial,* IEEE Transactions On Industrial Electronics, Vol. 41. No. 6. pages 567-583.

EKLER

EK A.1: Hazırlanan yazılım kaynak kodları, tez kapsamında verilen örnekleri ve tezin bilgisayar ortamındaki halini içeren CD verilmektedir.

ÖZGEÇMİŞ

Ad Soyadı: Hakkı KURUMAHMUT

Doğum Yeri ve Tarihi: Trabzon / OF, 1984

Adres: Haznedar Mahallesi Menderes Caddesi Çıkmazı No: 6 / 5 Güngören / İSTANBUL

Lisans Üniversitesi: Yıldız Teknik Üniversitesi, kurumahmut@gmail.com