İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

PETRİ AĞLARINDA MİNİMAL VE BAZ SAVAK-KAPAN ANALİZİ

YÜKSEK LİSANS TEZİ

Müh. Taner VAROL

Anabilim Dalı: ELEKTRONİK VE HABERLEŞME MÜHENDİSLİĞİ

Programı: ELEKTRONİK MÜHENDİSLİĞİ

EYLÜL 2007

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

PETRİ AĞLARINDA MİNİMAL VE BAZ SAVAK-KAPAN ANALİZİ

YÜKSEK LİSANS TEZİ

Müh. Taner VAROL
(504041241)

Tezin Enstitüye Verildiği Tarih : 13 Temmuz 2007
Tezin Savunulduğu Tarih : 17 Eylül 2007

Tez Danışmanı : Doç.Dr. Mürvet KIRCI

Diğer Jüri Üyeleri Prof.Dr. Ece Olcay GÜNEŞ

Prof.Dr. Müjde GÜZELKAYA

EYLÜL 2007

ÖNSÖZ

Bu çalışmanın gerçekleştirilmesi sırasında çalışmalarınımı yönlendiren ve yardımlarını hiçbir zaman esirgemeyen tez danışmanım sayın Doç. Dr. Mürvet ÜÇER'e, Devreler ve Sistemler Anabilim Dalı öğretim üyeleri ve asistanları ile çalışmalarımın her safhasında bana destek olan 'aileme' saygı ve minnet borçlu olduğumu belirterek teşekkürlerimi sunuyorum.

Temmuz, 2007 **Taner VAROL**

İÇİNDEKİLER

KISALTMALAR **v**
TABLO LİSTESİ **vi**
ŞEKİL LİSTESİ **vii**
ÖZET **ix**
SUMMARY **x**

1. GİRİŞ **1**

2. PETRİ AĞLARININ TEMELLERİ **3**
2.1 Petri Ağları Tanımları 3
2.2 Petri Ağı Yapısı ve Notasyonları 5
2.3 Petri Ağlarında Giriş ve Çıkış Fonksiyonları 7
2.4 Petri Ağları İşaretleri ve Durum Uzayı 7
2.5 Petri Ağları Dinamikleri 10
2.5.1 Erişilebilir durumlar 15
2.5.2 Erişilebilirlik ağacı 16
2.6 Analiz Yöntemleri 17
2.6.1 Erişilebilirlik ağacı yöntemi 17
2.6.2 Durum denklemleri ve durum-geçiş matrisi 21

3. PETRİ AĞLARININ ÖZELLİKLERİ **24**
3.1 Petri Ağlarının Davranışsal Özellikleri 24
3.1.1 Güvenlik (safeness) özelliği 24
3.1.2 Sınırlılık (boundedness) özelliği 24
3.1.3 Canlılık (liveness) özelliği 25
3.1.4 Tutarlılık (conservation) özelliği 26
3.1.5 Kapsanabilirlik (coverability) özelliği 29
3.1.6 Devamlılık (persistence) özelliği 29

4. PETRİ AĞLARI İLE MODELLEME **31**
4.1 Petri Ağları Yapıları 31
4.1.1 Paralellik (concurrency) 31
4.1.2 Senkronizasyon (synchronization) 32
4.1.3 Çakışma (conflict) 33
4.1.4 Sıralama (sequentiality) 33
4.1.5 Kaynak sınırlama (limited resources) 34
4.1.6 Bozucu (inhibitor) dallar 35
4.2 Petri Ağlarının Sınıflandırılması 37

5. PETRİ AĞLARINDA SAVAK(SIPHON) VE KAPAN(TRAP) YAPILARI **40**
5.1 Petri Ağlarında Savak(Siphon) ve Kapan(Trap) Yapıları 40
5.1.1 Petri ağlarında baz savak ve kapan yapıları 45
5.1.2 Petri ağlarında minimal savak ve kapan yapıları 45
5.1.3 Petri ağlarında yer minimal savak ve kapan yapıları 45
5.2 Savak ve Kapanlara Dayanan Özellikler 47

6. YER MİNİMAL İLE MİNİMAL SAVAK VE KAPAN PROGRAMLARI VE ÖZELLİKLERİ **52**
6.1 Programların Çalışma Ortamı 52
6.1.1 Programların dosya giriş ve çıkış formatları 52
6.2 Durum-Geçiş Matrisine Dayalı Programlama Yöntemi 54
6.3 Lojik Tabanlı İkili Tamsayı Lineer Programlama Yöntemi 59
6.3.1 Savakların lojik ifadelerle tanımlanması 59
6.3.2 Savak yapısının ikili tamsayı lineer programlama ile modellenmesi 61
6.3.3 Lojik tabanlı yöntemin VHDL benzetimi 65
6.4 Programların Analiz Sonuçları 69

7. SONUÇLAR VE ÖNERİLER **71**

KAYNAKLAR **72**

EKLER **73**

ÖZGEÇMİŞ **81**

KISALTMALAR

PA	: Petri Ağı
DM	: Durum Makinesi
İG	: İşaretli Graf
SSA	: Serbest Seçim Ağı
GSSA	: Genişletilmiş Serbest Seçim Ağı
ASA	: Asimetrik Serbest Seçim Ağı
EAY	: Erişilebilirlik Ağacı Yöntemi
KD	: Kapan Devresi
SD	: Savak Devresi
KİD	: Kapan İçeren Devre
SİD	: Savak İçeren Devre

TABLO LİSTESİ

Sayfa No

Tablo 3.1. Tutarlı PA örneğindeki geçişlerin giriş ve çıkış yeri sayısı.....29
Tablo 5.1. PA1 ve PA2'ye ait savak ve kapanlar. 45
Tablo 5.2. Şekil 5.4'teki yerlere ait yapılar.....46
Tablo 6.1. Örnek PA'na ait giriş excel dosyası 53
Tablo 6.2. Programların analiz sonuçları 70

ŞEKİL LİSTESİ

Sayfa No

Şekil 2.1 : Petri Ağı grafı bileşenleri 3
Şekil 2.2 : Asansörün Petri Ağı modeli 4
Şekil 2.3 : Asansör PA'nın P2 durumu 5
Şekil 2.4 : Örnek bir PN grafı 6
Şekil 2.5 : İşaretli Petri Ağı grafına örnek $x_1 = [1,0,0]$ 8
Şekil 2.6 : İşaretli Petri Ağı grafına örnek $x_2 = [2,1,0]$ 9
Şekil 2.7 : PA'nın işaretlemelerindeki değişim 11
Şekil 2.8 : x_0 işaretlemeli t_1, t_3 ve t_4 geçişlerinin izinli olduğu PA 12
Şekil 2.9 : t_4'ün ateşlenmesiyle oluşan $x_1 = [1,0,1,3,0]$ işaretlemesi 13
Şekil 2.10 : t_1'in ateşlenmesiyle oluşan $x_2 = [0,1,2,5,0]$ işaretlemesi 14
Şekil 2.11 : t_3'ün ateşlenmesiyle oluşan $x_3 = [0,1,2,3,1]$ işaretlemesi 15
Şekil 2.12 : Örnek bir PA grafı için erişilebilir durumlar 16
Şekil 2.13 : Örnek PA grafı 16
Şekil 2.14 : (Şekil 2.13)'deki PA'nın erişilebilirlik grafı 17
Şekil 2.15 : Erişilebilirlik ağacı örneği 17
Şekil 2.16 : t_1 ateşlenmesi ile oluşan erişilebilirlik ağacının 1. adımı 18
Şekil 2.17 : t_3 ateşlenmesi ile oluşan erişilebilirlik ağacının 2. adımı 19
Şekil 2.18 : t_1 ateşlenmesi ile oluşan erişilebilirlik ağacının 3. adımı 19
Şekil 2.19 : t_3 ateşlenmesi ile oluşan erişilebilirlik ağacının 4. adımı 20
Şekil 2.20 : t_2 ateşlenmesi ile oluşan erişilebilirlik ağacının son adımı 20
Şekil 2.21 : Erişilebilirlik grafı 21
Şekil 3.1 : Canlılık özelliğine ilişkin örnek PA grafı 26
Şekil 3.2 : Tutarsız Petri Ağı 28
Şekil 3.3 : Tutarlı PA grafı örneği 28
Şekil 3.4 : Devamlılık özelliği olmayan örnek bir PA grafı 30
Şekil 3.5 : Devamlılık özelliği olan örnek bir PA grafı 30
Şekil 4.1 : Paralellik örneği 32
Şekil 4.2 : Senkronizasyon örneği 32
Şekil 4.3 : Çakışma örneği 33
Şekil 4.4 : Sıralama örneği 33
Şekil 4.5 : Kaynak sınırlama örneği 34
Şekil 4.6 : Bozucu dal örneği 35
Şekil 4.7 : Çoklu bozucu dal örneği 36
Şekil 4.8 : Geçişlere öncelik atama 36
Şekil 4.9 : PA sınıfları 39
Şekil 5.1 : PA yapıları (a) savak ve (b) kapan 41
Şekil 5.2 : Savak-kapan yapılarına ait örnek PA1 44
Şekil 5.3 : Savak-kapan yapılarına ait örnek PA2 44
Şekil 5.4 : Baz ve minimal savak(kapan) örneği[3]. 46

Şekil 5.5 : Canlı asimetrik seçim ağı (ASA).....47
Şekil 5.6 : Canlı asimetrik seçim ağı (ASA).....48
Şekil 5.7 : Özellik 3'e ilişkilendirilen örnek (a) KDA (N, X_0) ve.....49
(b) alt ağı (N_x, X_{0x}) 49
Şekil 5.8 : Özellik 4'e ilişkilendirilen örnek 5.6'ya ait PA'nın alt ağı (N_x, X_{dx}) ...50
Şekil 6.1 : Giriş dosyası için örnek PA.....52
Şekil 6.1 : Minimal savak (siphon) çıkış dosyası.....53
Şekil 6.2 : Yer minimal savak (siphon) çıkış dosyası 54
Şekil 6.3 : Durum-geçiş matrisi algoritması 55
Şekil 6.4 : p_1 yerine göre üretilen yer minimal savaklar 57
Şekil 6.5 : p_2, p_3, p_4 *ve* p_5 yerlerine göre üretilen yer minimal savaklar.....58
Şekil 6.6 : Lojiğe dayalı ikili tamsayı lineer programlama algoritması 64
Şekil 6.7 : VHDL ile programlanan PA örneği.....66

PETRİ AĞLARINDA MİNİMAL VE BAZ SAVAK-KAPAN ANALİZİ

ÖZET

Ayrık olaylı ve eş zamanlı sistemlerin davranışlarını incelemek ve analiz etmek için bu sistemlerin modellenmesine ihtiyaç vardır. Bu modellemeleri gerçekleştirmek için farklı yöntem ve modelleme araçları mevcuttur. Bu çalışmada, bu modelleme yöntemlerinden en iyisi olan Petri Ağları(Petri Nets) incelenmiş ve Petri Ağı yapısal özelliklerinden olan Minimal savak (kapan) ile Baz savak (kapan) yapılarını bulan iki algoritma geliştirilmiş ve bu algoritmaları uygulayan bilgisayar programları gerçekleştirilmiştir.

Petri Ağları sistemlerin grafiksel olarak modellenmesini sağlamakta, sunduğu matematiksel ve yapısal analiz teknikleriyle bir sistemin davranışlarının incelenmesini hayli kolaylaştırmaktadır. Petri Ağları bir graf olarak ifade edilmekte ve bu graf modellenecek sistemin yapısına karşılık düşürülmektedir. Sistemin dinamik davranışını jeton denilen aktif neslelerle ifade edilmektedir. Böylelikle modellenen sistemin hem statik yapısı hem de dinamik yapısı tek bir graf ile gösterilebilmektedir. Bunun yanı sıra matematiksel olarak ifade edilmesi bakımından Petri Ağları güçlü analiz teknikleri sunmaktadır.

Bu çalışmanın esasını temsil eden, Petri Ağlarının yapısal özelliklerinden olan minimal savak(kapan) ile baz savak(kapan) yapıları tanımlanmış, Petri Ağ özelliklerinden olan canlılık, erişilebilirlik ve tersinirlilik gibi özelliklerinde kullanım alanları incelenmiştir. Bu yapıları bulan biri durum-geçiş matrisine diğerise lojik tabanına dayanan iki farklı program geliştirilmiştir. Programların akış diyagramları ve algoritmaları gösterilmiş, farklı yapı ve boyuta sahip Petri Ağları için bu yapılar elde edilmiş ve zaman analizleri gösterilmiştir.

MINIMAL AND BASIS SIPHON-TRAP ANALYSIS IN THE PETRİ NETS

SUMMARY

It is a necessity to model discrete and concurrent systems in order to analyze and inspect the behavior of that system. For this purpose, there are numerous methods and modeling tools. Petri Nets, one of the best defined modeling tool were researched during this study. For computing Minimal siphon(trap) and Basis siphon(trap), structural properties of Petri Nets to be presented two algoritm and developed computer programs implementing algoritms.

Petri Nets combines both graphical and mathematical aspects of a modeling tool in one. It provides a graphical representation of the system as well as mathematical analysis and inspection opportunities for hat system. Graphical aspect of petri Nets makes it easy to comprehend the system. Petri Nets actually consists of a graph and a token traveling through that graph. The graph structure is used to model the static aspects of the system whereas the token is used to model the dynamic behavior of the same system. Besides that the mathematical properties exposed by the Petri Nets are used to analyze the modeled system in detail.

The objective of this study is the structural properties of Petri Nets, minimal siphon(trap) and basis siphon(trap) were examined on the basis of the definitions and possible uses of these structure for liveness, reachability and reversibility. Two different computer programs, one of them based on state transition matrix and the other based on logic was developed. The program's workflows and algorithms were presented. In conslusion, performance analysis of the programs for different structured and sized Petri Nets were performed and tabulated.

1. GİRİŞ

Petri Ağları gerçek hayattaki sistemlerin modellenmesinde kullanılan hem grafiksel hem de matematiksel bir araçtır. Petri ağları matematiksel yönüyle sistemlerin daha detaylı analiz edilmesini, grafiksel yönüyle ise sistemlerin davranışlarının daha kolay incelenmesini sağlamaktadır.

Petri ağları ilk defa 1962 yılında Carl Adam Petri tarafından ortaya atılmıştır. C.A. Petri'nin bu tarihten itibaren Petri Ağları ile ilgili birçok makalesi yayımlanmış ve teoriden pratiğe geçilerek gerçek hayatta birçok sistemin modellenmesinde kullanılmaya başlanmıştır. 1970 ile 1975 tarihleri arasında MIT'li bir grup (Computation Structure Group) Petri ağları ile ilgili geniş araştırmalar yapmış ve çok sayıda rapor ile tez sunmuşlardır. Temmuz 1975'te Petri Ağlarına dayalı metodlar adı altında MIT'de konferans düzenlenmiştir. Ekim 1979'da Almanya'nın Hamburg kentinde çoğunluğu Avrupa ülkelerinden 135 araştırmacının katıldığı Petri Ağları süreçleri ve sistemleri ile ilgili iki haftalık ileri seviye kurslar düzenlenmiştir. Bu etkinliklerin ikincisi yine Almanya'nın Bad Honnef kentinde düzenlenmiş ve C.A. Petri tarafından eşzamanlılıkla ilgili yeni aksiyomları içeren bildiriler yayımlanmıştır. Bu tarihten itibaren Avrupanın çeşitli şehirlerinde her yıl Petri Ağlarının uygulama ve teorileri ile ilgili etkinlikler düzenlenmektedir[3].

Geniş uygulama alanları bulunan Petri ağları; eşzamanlı, asenkron, dağıtık, paralel, nedensel olmayan ve/veya stokastik sistemlerin modellenmesinde kullanılır.

Petri Ağları örnek uygulamaları:

- performans değerlendirme
- haberleşme protokolleri
- dağıtık-yazılım sistemleri
- dağıtık-veritabanı sistemleri
- eşzamanlı ve paralel programlar
- endüstriyel kontrol sistemleri

- ayrık-olay sistemleri
- çok işlemcili hafıza sistemleri
- veri akışı-düzenleme sistemleri
- hata-toleranslı sistemler

olarak verebiliriz.

Petri ağları ile bir sisteme ait aşağıdaki davranış şekilleri modellenebilir:

Senkronizasyon(synchronization): Bir olay gerçekleştirilmeden önce bu olaya bağlı diğer olayların gerçekleşmesinin beklenmesi.

Sıralama (sequentiality): Olayların gerçekleşme sıralarının belirlenmesi.

Paralellik(concurrency): Olayların aynı anda gerçekleşmesinin sağlanması

Çakışma (conflict): Aynı koşula bağlı iki ya da daha fazla olaydan aynı anda yalnızca birinin gerçekleşmesine izin verilmesi.

Petri ağları sunduğu güçlü analiz teknikleriyle bir sistemin daha kolay incelenebilmesini sağlamaktadır. Bu analiz teknikleri kullanılarak bir sistemin davranışları detaylı olarak incelenebilir ve ortaya çıkan problemler rahat bir şekilde giderilebilir.

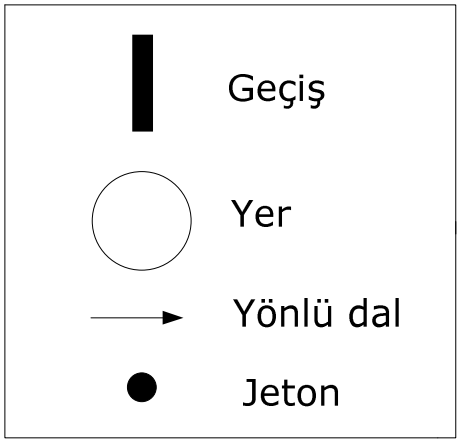
Bu çalışmanın amacı Petri Ağları ile modellenmiş sistemlerin, Petri Ağlarının yapısal özelliklerinden olan savak (siphon) ve kapan (trap) yapılarında, sistemlerin canlılığı gibi özelliklerinin analizinde kullanılan minimal ve yer minimal yapıları belirlemektir. Bu çalışmanın 2. bölümünde Petri Ağları ile ilgili temel bilgiler verilecek ve analiz yöntemlerinden bahsedilecektir, 3. bölümde Petri Ağlarının özelliklerinden bahsedilecek, 4. bölümde petri Ağlarının sistem modellenmesinde nasıl kullanılabileceği anlatılacak, 5. bölümde tez çalışmasını teşkil eden Petri Ağlarının yapısal özelliklerinden bahsedilecek ve 6. bölümde geliştirilen 2 farklı program anlatılacak ve sonuçları verilecektir.

2. PETRİ AĞLARININ TEMELLERİ

2.1 Petri Ağları Tanımları

Petri Ağı, ayrık olay sistemleri için geliştirilmiş, olayları belli kurallar dâhilinde kullanan bir araçtır. Petri ağları iki adımda açıklanabilir; ilk önce Petri ağ yapısı(Petri ağları grafı) tanımlanır, daha sonra bu grafa başlangıç durumu, işaretli durumlar, geçiş fonksiyonları ekleyerek bir Petri ağları modeli, ilgili dinamikleri ile beraber verilir[1].

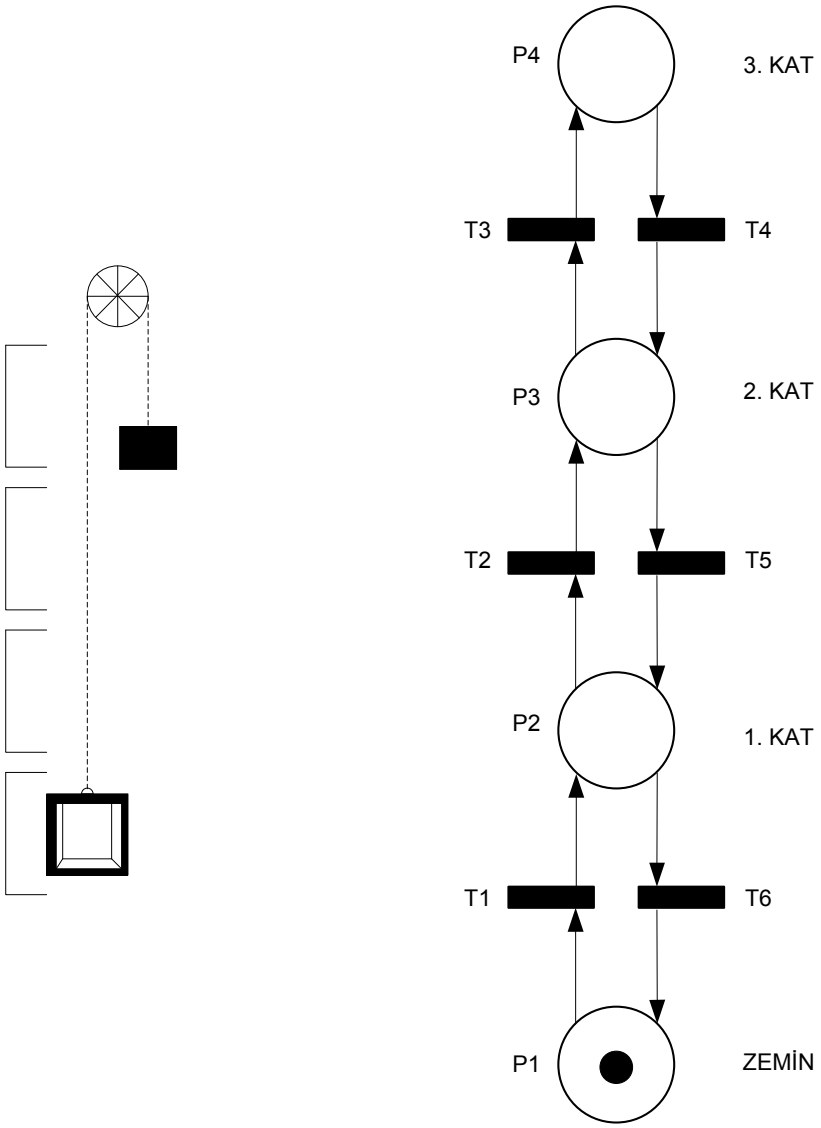
Petri ağı yapısı, yerler ve geçişler olmak üzere iki tip düğüme ve bunları birbirine bağlayan yönlü dallara sahiptir. Dallar, aynı tip düğümleri birbirine bağlamazlar ya yer düğümünü geçiş düğümüne ya da geçiş düğümünü yer düğümüne bağlarlar.



Şekil 2.1: Petri Ağı grafı bileşenleri

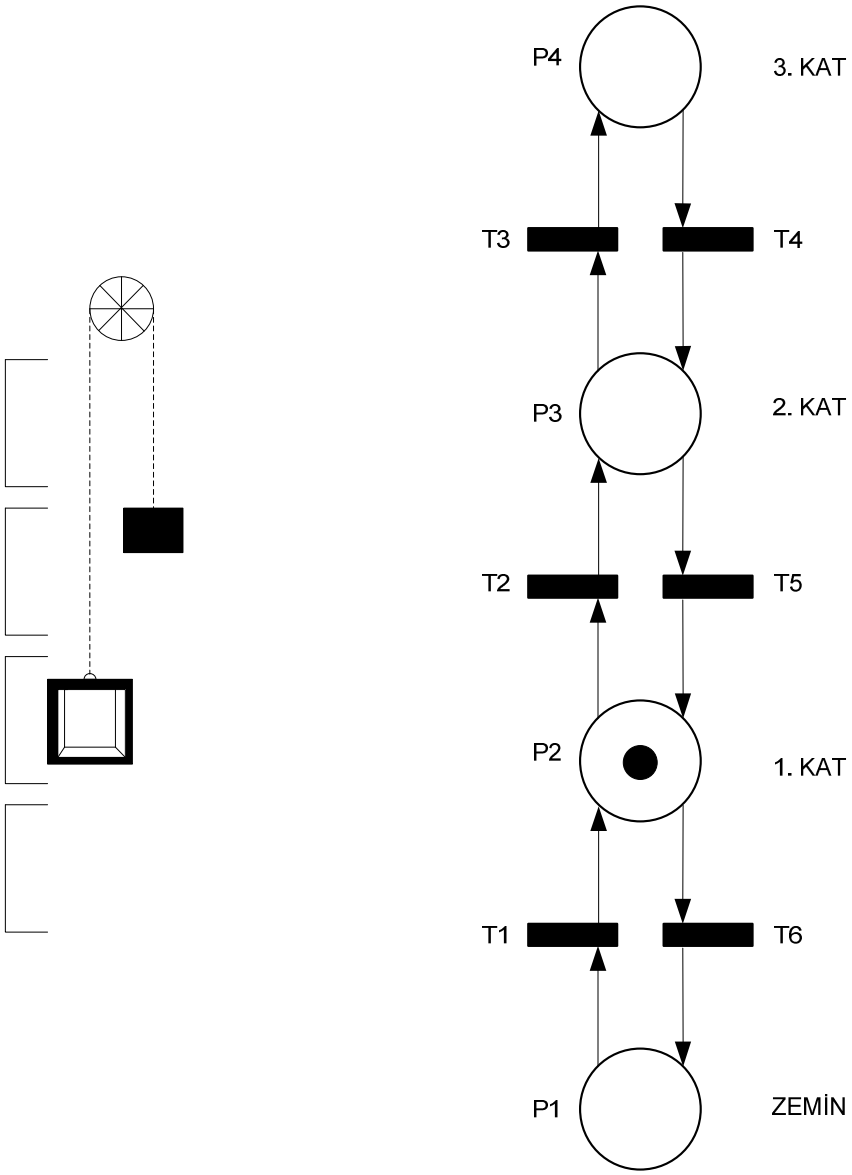
Petri ağındaki geçişler ayrık olay sistemlerindeki olayları temsil ederler. Bir geçişin olabilmesi için incelenen sisteme ilişkin, bazı şartların sağlanması gerekir. Bu şartlara ilişkin bilgiler tanımlı yerlerde bulunur. Bu gibi bazı yerler, bir geçişe 'giriş' olarak düşünülür ve bu geçişin sağlanabilmesi için gereken şartlar ile birlikte ilişkilendirilir. Aynı şekilde bazı yerler de bir geçişin 'çıkışı' olarak düşünülür. Bunlarda geçiş sonucu oluşan şartlarla ilişkilendirilir[1].

Petri Ağına basit bir örnek olarak asansörün Petri Ağı modelini verebiliriz[14]. Şekil 2.2'de bu model görülmektedir.



Şekil 2.2: Asansörün Petri Ağı modeli

Burada jeton asansörü temsil etmektedir. Bu modelde T1'den T3'e kadar olan geçişler asansörün yukarı doğru olan hareketini temsil eden olayları belirtir. T4'ten T6'ya kadar olan geçişler ise asansörün aşağıya doğru olan hareketini belirtir. P1'den P4'e kadar olan yerler ise asansörün durumunu belirten koşullara karşılık gelir. Petri ağından görüldüğü gibi ilk durumda sadece yukarı doğru hareket sağlanabilir. Bu yukarı doğru harekete karşılık gelen T1 olayının gerçekleşmesi ile P1 yerindeki jeton P2 yerine aktarılmış olur. P2 yerinde jetonun varlığı T2 ve T6 geçişlerine ait olayları mümkün kılmaktadır. T2 geçişine ait olayın gerçekleşmesi ile yukarı doğru hareket, T6 geçişine ait olayın gerçekleşmesi ile aşağı doğru hareket sağlanabilir. Bu yeni durum şekil 2.3'te görülmektedir.



Şekil 2.3: Asansör PA'nın P2 durumu

2.2 Petri Ağı Yapısı ve Notasyonları

Bir Petri Ağı dört temel bileşenden oluşur;

P: yerler kümesi

T: geçişler kümesi

F: yer ve geçiş kümeleri arasındaki ilişkiyi gösteren bağıntı

W: yerleri ve geçişleri birbirine bağlayan dalların ağırlıklarını tanımlar.

Bir Petri Ağı tanımında yerler, geçişler, yerlerle geçişler arasındaki bağıntı ve de dal ağırlıkları bilgileri bulunur.

Bir Petri ağı yapısını **N=(P,T,F,W)** şeklinde dört elemanlı bir küme olarak belirtebiliriz.

$P = \{p_1, p_2, ..., p_n\}$ sonlu yerler kümesi ($n \geq 0$)

$T = \{t_1, t_2, ..., t_m\}$ sonlu geçişler kümesi ($m \geq 0$)

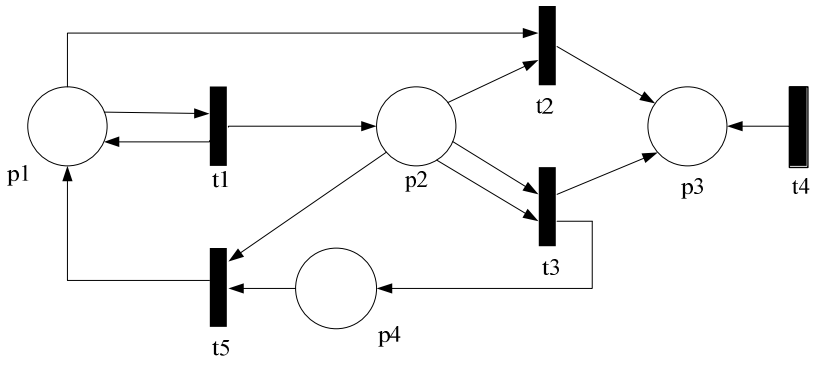
$F : F \subseteq (P \times T) \cup (T \times P)$ olmak üzere grafta yerlerden geçişlere ve geçişlerden yerlere çizilebilen dallar kümesi.

$W : F \rightarrow \{1, 2, 3, ...\}$ dalların ağırlık fonksiyonunu tanımlar.

Petri ağı yapısını oluşturan yerler ve geçişler kümesi ayrıktır, $P \cap T = \varnothing$

Yerler ile geçişlerin kümelerinin bileşkesi boş kümeden farklıdır. $P \cup T \neq \varnothing$

P kümesinin eleman sayısı n ve $\mathbf{p_i}$, i = 1, 2, ..., n, ile T kümesinin eleman sayısı m ve $\mathbf{t_j}$, j = 1, 2, ..., m ile ifade edilecektir.



Şekil 2.4: Örnek bir PN grafı

Şekil 2.4 'te gösterilen PA grafı için aşağıdakiler yazılabilir.

$P = \{p_1, p_2, p_3, p_4\}$

$T = \{t_1, t_2, t_3, t_4, t_5\}$

$$F = \left\{ \begin{matrix} (p_1,t_1),(p_1,t_2),(p_2,t_2),(p_2,t_3),(p_2,t_5),(p_4,t_5) \\ (t_1,p_1),(t_1,p_2),(t_2,p_3),(t_3,p_3),(t_3,p_4),(t_4,p_3),(t_5,p_1), \end{matrix} \right\}$$

$w(p_1,t_1)=1$ $w(p_1,t_2)=1$ $w(p_2,t_2)=1$ $w(p_2,t_3)=2$
$w(p_2,t_5)=1$ $w(p_4,t_5)=1$ $w(t_1,p_1)=1$ $w(t_1,p_2)=1$
$w(t_2,p_3)=1$ $w(t_3,p_3)=1$ $w(t_3,p_4)=1$ $w(t_4,p_3)=1$
$w(t_5,p_1)=1$

2.3 Petri Ağlarında Giriş ve Çıkış Fonksiyonları

Giriş ve çıkış fonksiyonları bir Petri ağında bulunan geçişleri ve yerleri ilişkilendirir. Giriş fonksiyonu **I**, çıkış fonksiyonu da **O** ile ifade edilir. **I** giriş fonksiyonu bir $\mathbf{t_j}$ geçişinden ilişkili olduğu yerlere $\mathbf{I(t_j)}$'ye bir eşlemedir. Bunlara o geçişin giriş yerleri olarak bakılabilir. Benzer şekilde **O** çıkış fonksiyonu $\mathbf{t_j}$ geçişinden çıkan tüm $\mathbf{O(t_j)}$ çıkış yerlerine bir eşlemedir. Bu eşlemelerde yer alan değerler geçiş ile yer arasındaki dalın ağırlığı kadar değer alır.

Bu durumda;

$$I(t_j)=\{p_i \in P:(p_i,t_j)\in F\} \quad \textbf{(2.1)}$$

$$O(t_j)=\{p_i \in P:(t_j,p_i)\in F\} \quad \textbf{(2.2)}$$

Aynı şekilde bir geçişten p_i yerine yapılan girişleri $I(p_i)$, çıkışlarıda $O(p_i)$ olarak gösterebiliriz.

Örnek 2.1: Şekil 2.4' te verilen örnek Petri Ağı için giriş ve çıkış yerlerini yazarsak,

$I(t_1)=\{p_1\}$ $O(t_1)=\{p_1,p_2\}$
$I(t_2)=\{p_1,p_2\}$ $O(t_2)=\{p_3\}$
$I(t_3)=\{p_2\}$ $O(t_3)=\{p_3,p_4\}$ bağıntılarını elde ederiz..
$I(t_4)=\{\varnothing\}$ $O(t_4)=\{p_3\}$
$I(t_5)=\{p_2,p_4\}$ $O(t_5)=\{p_1\}$

2.4 Petri Ağları İşaretleri ve Durum Uzayı

Petri ağı grafındaki geçişlerin ayrık olaylı sistemdeki olayları temsil etmesi için Petri ağındaki yerleri bu olayların meydana gelmesindeki şartlar olarak tanımlayabiliriz. Bu çerçevede şartların sağlanıp sağlanmadığını kontrol etmek için bir yapıya ihtiyaç vardır. Bu yapı yerlere jetonlar (token) atanarak sağlanabilir. Yerlere konulan jetonlar o yerin gerektirdiği şartlar olarak tanımlanır. Petri ağı grafında yerlere atanan

jetonlar **işaretleme (marking)** olarak tanımlanır[1,2]. Bu özellikleride göz önüne almak için Petri ağı tanımında ek büyüklüklere gerek olduğundan tanım aşağıdaki gibi düzenlenmiştir.

Tanım 2.1: Bir Petri ağları grafında (P,T,F,w) işaretli Petri ağını N ile gösterirsek

$N=(P,T,F,w,E,L,X_0,X_m)$ olur. Burada;

E: geçişleri etkilemek için olay kümesi

L: $T \to E$ geçiş etiketleme fonksiyonu

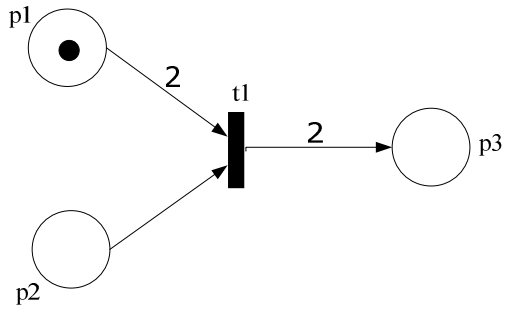
$X_0 \in N^n$ ağın ilk durum kümesi (her yerdeki jetonların ilk sayısı)

$$X : P \to N = \{0,1,2,...\} \quad \textbf{(2.3)}$$

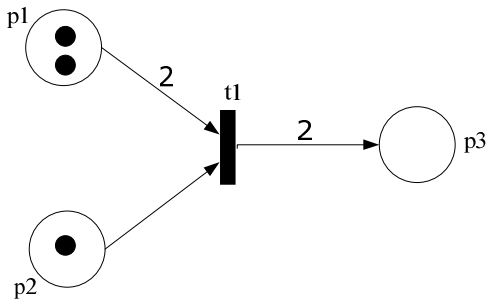
$$X = [x(p_1), x(p_2), x(p_3), ..., x(p_n),] \quad \textbf{(2.4)}$$

$X_m \subseteq N^n$ ağın işaretlenmiş durum kümesidir.

Petri ağındaki yer sayısı n olmak üzere işaretleme x satır vektörü olarak tanımlanır. Bu vektörün $i.$elemanı p_i yerindeki jeton sayısını gösterir. Petri ağı grafında bir jeton ilgili yerin üzerinde siyah bir nokta ile gösterilir.



Şekil 2.5: İşaretli Petri Ağı grafına örnek $x_1 = [1,0,0]$



Şekil 2.6: İşaretli Petri Ağı grafına örnek $x_2 = [2,1,0]$

Jetonların Petri ağı içindeki dağılımı petri ağının o anki durumunu tanımlar. Yerlere atanan jeton sayısı sıradan pozitif tamsayılardır. N yere sahip Petri ağının durum uzayı pozitif tam sayı elemanlı n boyutlu vektörler ile ifade edilebilir $X = N^n$.

Buraya kadar ki tanımlamalar Petri ağlarının durum geçiş mekanizmasını tanımlamaya yeterli değildir. Ayrık olaylı sistemi Petri ağları ile modelleyebilmek için bir notasyona daha ihtiyaç vardır. Petri ağları yapısındaki durum geçiş mekanizması için 'ateşlenebilir geçiş (enabled transition)' kavramıda kullanılmalıdır[1,2].

Tanım 2.2: Ateşlenebilir geçiş

Petri ağında bir geçişin $t_j \in T$ ateşlenebilmesi için aşağıdaki koşul sağlanmalıdır.

$Her \quad p_i \in I(t_j) \quad için \quad x(p_i) \geq w(p_i, t_j)$ olmalıdır.

İfadeden de anlaşılacağı üzere t_j geşinin ateşlenebilmesi için p_i'eki jeton sayısı en az p_i'yi t_j'ye bağlayan dal ağırlığı kadar olmalıdır.(t_j'ye girişi olan bütün p_i'ler için).

Özetlemek gerekirse, PA grafında yerler, bir geçişin ateşlenebilmesi için gerekli koşulları temsil eder. Eğer bir geçişe ilişkin bütün koşullar sağlanıyorsa bu geçiş ateşlenebilir hale gelir. Söz konusu koşulların sağlanmış olup olmadığı ise jetonlarla belirtilir[1].

2.5 Petri Ağları Dinamikleri

PA'da durum geçiş mekanizması jetonların hareket ettirilmesiyle sağlanır. Böylelikle PA'nın durumu değiştirilmiş olur ve PA bulunduğu durumdan başka bir duruma geçer. Bu da ancak izinli geçişlerin ateşlenmesiyle (firing) mümkün olabilir. PA'nın durum geçiş fonksiyonu, izinli geçişlerin ateşlenmesiyle PA'nın durumunda meydana gelen değişimler olarak tanımlanır[1].

Tanım 2.3: Durum geçiş fonksiyonu

Bir (P,T,F,w) PA'na ilişkin $f : N^n \times T \to N^n$ durum geçiş fonksiyonu aşağıdaki koşulu sağlayan $t_j \in T$ geçişi ile tanımlanmıştır.

$$\forall p_i \in I(t_j), x(p_i) \geq w(p_i,t_j) \quad \textbf{(2.5)}$$

Eğer $f(x,t_j)$ tanımlıysa, $x' = f(x,t_j)$ şu şekilde olur. $i = 1,2,...,n$

$$x'(p_i) = x(p_i) - w(p_i,t_j) + w(t_j,p_i) \quad \textbf{(2.6)}$$

Koşul (2.5), durum geçiş fonksiyonunun sadece izinli geçişler için tanımlanmış olmasını garanti eder. Denklem (2.6)' dan görüleceği gibi bir sonraki durumla ilgili geçişin giriş ve çıkış yerlerine ve bu yerlerden geçişe bağlanan dalların ağırlığına bağladır.

Denklem (2.6) ya göre, eğer bir p_i yeri t_j'nin giriş yeri ise bu yerdeki jeton sayısı p_i'yi t_j'ye bağlayan dalın ağırlık değeri kadar azalır; eğer p_i yeri t_j'nin çıkış yeri ise bu yerdeki jeton sayısı t_j'yi p_i'ye bağlayan dalın ağırlık değeri kadar artar[1,2].

(2.6)'dan görüleceği gibi aşağıdaki eşitsizlikler mümkündür. Bu durumda PA'da jeton sayısı herhangi bir geçişin ateşlenmesinin ardından korunmayabilir.

$$\sum_{p_i \in P} w(t_j,p_i) > \sum_{p_i \in P} w(p_i,t_j) \quad \textbf{(2.7)}$$

$$\sum_{p_i \in P} w(t_j,p_i) < \sum_{p_i \in P} w(p_i,t_j) \quad \textbf{(2.8)}$$

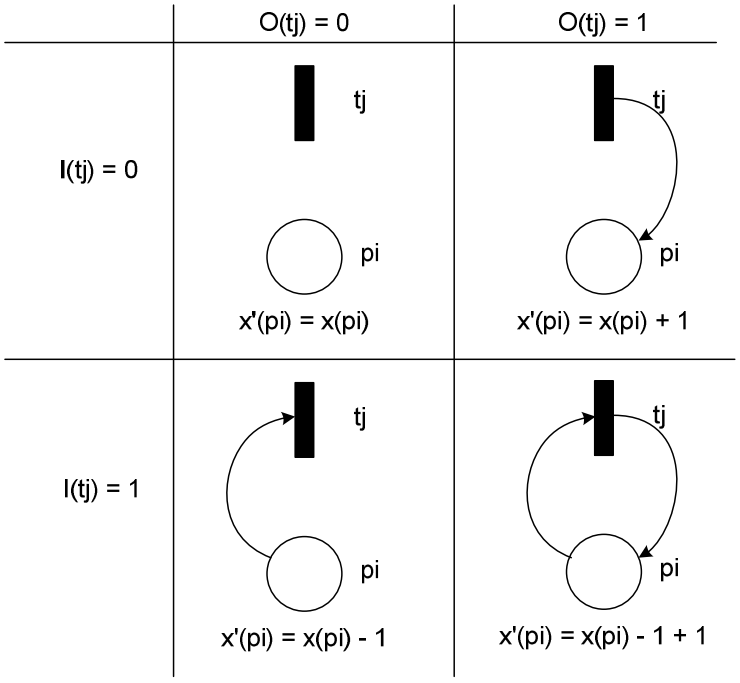
(2.7) eşitsizliğinin geçerli olması durumunda $x' = f(x,t_j)$ x'ten daha fazla sayıda jeton içerecektir. (2.8) eşitsizliğinin geçerli olması durumunda ise $x' = f(x,t_j)$ x'ten daha az sayıda jeton içerecektir.

Bazı durumlarda birkaç geçişten sonra x=[0,…..,0] durumuna varılması mümkün olmakla birlikte bir başka olasılık olarakta aşırı miktardaki geçişin ardından belli yerlerdeki jeton sayısı çok fazla artabilir.

Özetlemek gerekirse bir geçişin ateşlenmesi ile ilgili olarak şunları söyleyebiliriz.

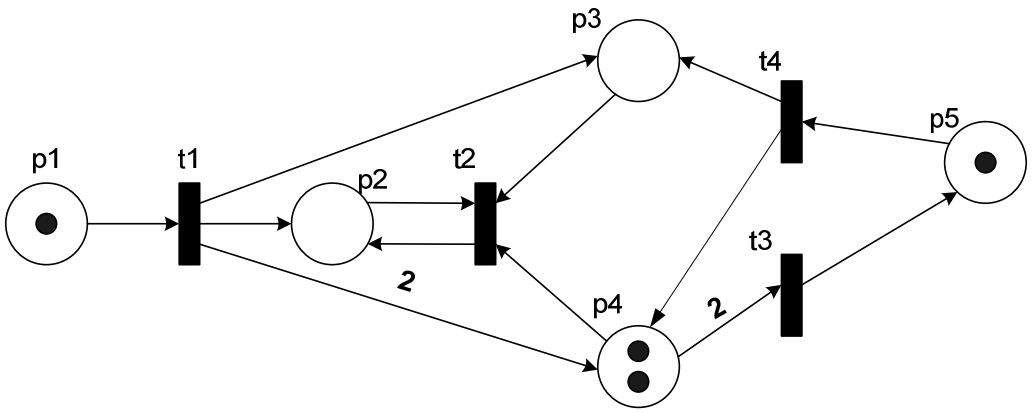
- Eğer bir geçişe ilişkin bütün giriş yerleri en az bu yerleri ilgili geçişe bağlayan dalların ağırlık değeri kadar jeton içeriyorsa söz konusu geçiş izinli demektir.
- Sadece izinli geçişler ateşlenebilir. İzinli geçişin ateşlenmesiyle PA'nın bir durumdan başka bir duruma geçmesi sağlanabilir.
- İzinli bir geçiş ateşlendiğinde, geçişe bağlı giriş yerlerinin dallarındaki ağırlık değeri kadar giriş yerlerinden jeton eksiltilir.
- İzinli bir geçiş ateşlendiğinde, geçişe bağlı çıkış yerlerinin dallarındaki ağırlık değeri kadar çıkış yerlerine jeton eklenir.

İzinli bir geçiş her zaman ateşlenmeyebilir. Çünkü PA'da geçişler olaylarla ilişkilendirilmiştir. Bu olaylara bağlı koşulların sağlanmış olsa bile geçişlerin ateşlenebilmesi için olaylara ilişkin bazı dış koşullarında sağlanmış olması gerekir.



Şekil 2.7: PA'nın işaretlemelerindeki değişim

Örnek 2.2: Bir petri ağının durumunun geçişler sonucu değişmesini bir örnekle gösterelim. İlk durumu $x_o = [1,0,0,2,1]$ olan bir petri ağını inceleyelim. Şekil 2,8'den görüldüğü gibi t_1, t_3 ve t_4 geçişleri mümkündür.



Şekil 2.8: x_0 işaretlemeli t_1, t_3 ve t_4 geçişlerinin izinli olduğu PA

Şekil2.8'den (2.5) koşullarının gerçeklendiğini görüyoruz.

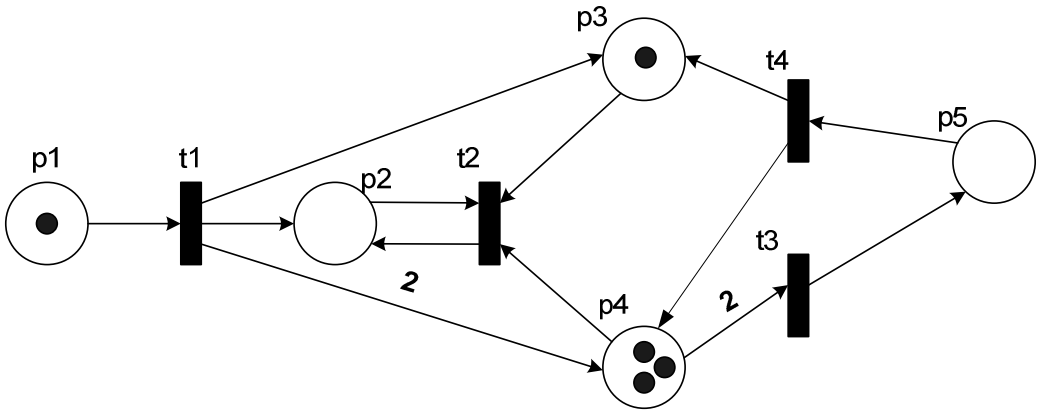
$$x(p_1) = 1 = w(p_1, t_1)$$
$$x(p_4) = 2 = w(p_4, t_3)$$
$$x(p_5) = 1 = w(p_5, t_4)$$

Eğer bir PA'da birden fazla izinli geçiş varsa bu geçişler istenen sırada ateşlenebilir.

t_4 geçişi ateşlenirse ve denklem (2.6) uygulanırsa x_1 durumuna aşağıdaki gibi ulaşılır.

$$x_1(p_3) = x_0(p_3) + w(t_4, p_3) = 0 + 1 = 1$$
$$x_1(p_4) = x_0(p_4) + w(t_4, p_4) = 2 + 1 = 3$$
$$x_1(p_5) = x_0(p_5) - w(p_5, t_4) = 1 - 1 = 0$$

p_1 ve p_2 yerlerindeki jeton sayısı değişmeyeceği için $x_1 = [1,0,1,3,0]$ işaretlemesi elde edilir. Bu vektöre ilişkin PA grafı Şekil 2.9'da gösterilmiştir.



Şekil 2.9: t_4 'ün ateşlenmesiyle oluşan $x_1 = [1,0,1,3,0]$ işaretlemesi

Şekil 2.9'dan görülebileceği gibi sadece t_1 ve t_3 geçişleri mümkündür.

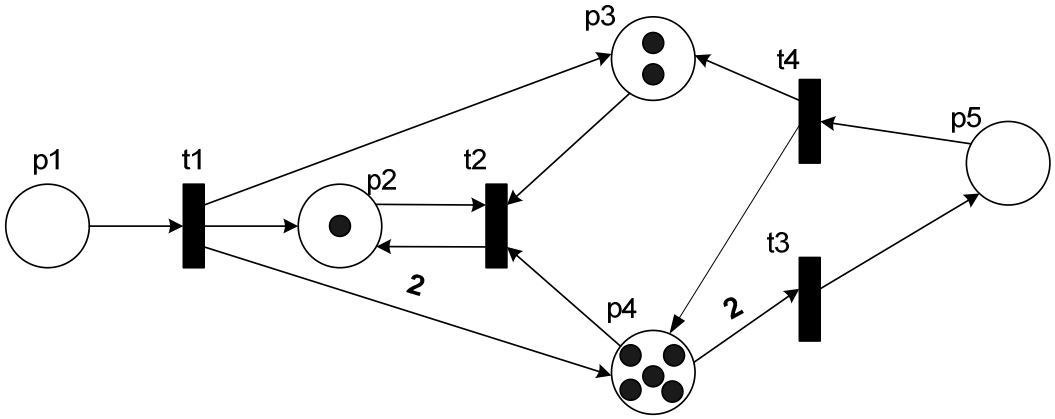
Koşul (2.5) göre:

$x(p_1) = 1 = w(p_1, t_1)$
$x(p_4) = 3 > w(p_4, t_3) = 2$

Eğer t_1 geçişi ateşlenirse ve denklem (2.6) uygulanırsa x_2 durumu aşağıdaki gibi olur;

$x_2(p_1) = x_1(p_1) - w(p_1, t_1) = 1 - 1 = 0$
$x_2(p_2) = x_1(p_2) + w(p_1, t_1) = 0 + 1 = 1$
$x_2(p_3) = x_1(p_3) + w(t_1, p_3) = 1 + 1 = 2$
$x_2(p_4) = x_1(p_4) + w(t_1, p_4) = 3 + 2 = 5$

p_5 yerindeki jeton sayısı değişmeyeceği için $x_2 = [0,1,2,5,0]$ işaretlemesi elde edilir. Bu vektöre ilişkin PA grafı Şekil 2.10'da gösterilmiştir.



Şekil 2.10: t_1'in ateşlenmesiyle oluşan $x_2 = [0,1,2,5,0]$ işaretlemesi

Şekil 2.10'dan görelebileceği gibi ancak t_2 ve t_3 geçişleri izinlidir. (2.5) koşulundan geçiş izinlerini incelersek;

$$x(p_2) = 1 = w(p_2, t_2)$$
$$x(p_3) = 2 > w(p_3, t_2) = 1 \qquad \Rightarrow t_2 \quad geçişi \quad için,$$
$$x(p_4) = 5 > w(p_4, t_2) = 1$$

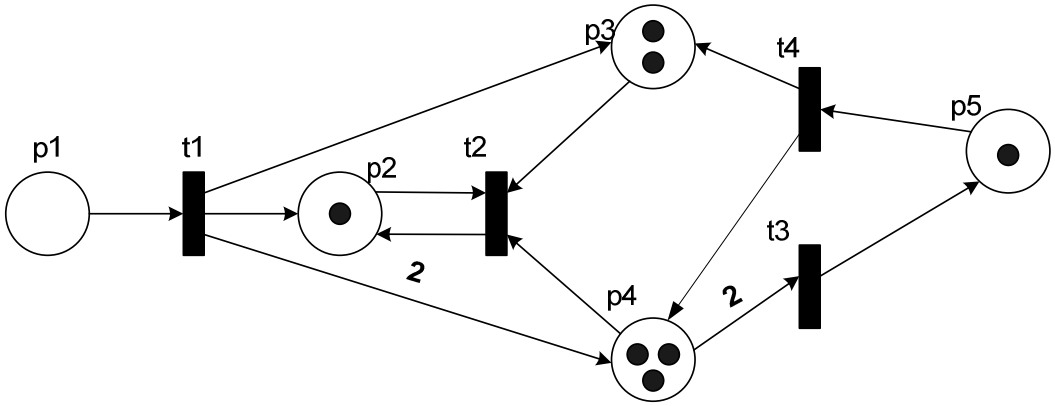
$$x(p_4) = 5 > w(p_4, t_3) = 2 \qquad \Rightarrow t_3 \quad geçişi \quad için.$$

Şartlarının sağlandığını görürüz.

Eğer t_3 geçişi ateşlenirse ve denklem (2.6)'ya göre x_3 durumu aşağıdaki gibi elde edilir.

$$x_3(p_4) = x_2(p_4) - w(p_4, t_3) = 5 - 2 = 3$$
$$x_3(p_5) = x_2(p_5) + w(t_3, p_5) = 0 + 1 = 1$$

p_1, p_2 ve p_3 yerlerindeki jeton sayısı değişmeyeceği için $x_3 = [0,1,2,3,1]$ işaretlemesi elde edilir. Bu vektöre ilişkin PA grafı Şekil 2.11'de gösterilmiştir.



Şekil 2.11: t_3'ün ateşlenmesiyle oluşan $x_3 = [0,1,2,3,1]$ işaretlemesi

Bu örnekte izinli geçişlerin ateşlenmesiyle birlikte devam eden işaretlemeleri aynı şekilde elde edebiliriz. Örnekte jeton sayısı korunmamaktadır. x_0 durumu 4, x_1 durumu 5, x_2 durumu 8 ve x_3 durumu 7 adet jeton içermektedir.

2.5.1 Erişilebilir durumlar

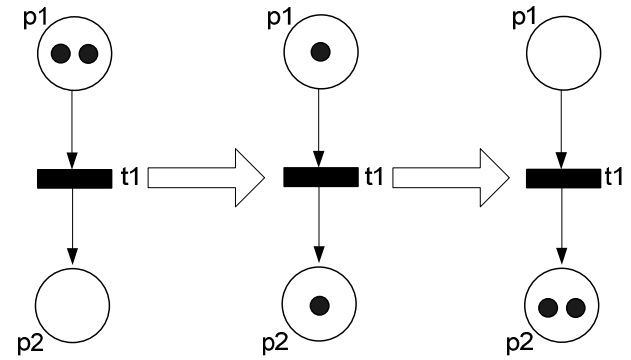
Bir PA'da N^n'deki bütün durumlara erişilmesi mümkün olmayabilir. Bunun için bütün erişilebilir(reachable) durumlar kümesinin oluşturulması gerekir.

Bir $N = (P,T,F,w)$ PA ve x_0 ilk işaretlemesi verilmiş olsun. Bu durumda PA'nın $R(N,x_0)$ ulaşılabilirlik kümesi aşağıdaki bağıntıyı sağlayan işaretlemeler kümesidir.

$$R[(P,T,F,w,x)] = \left\{ y \in N^n : \exists s \in T^* (f(x,s) = y) \right\} \quad \textbf{(2.9)}$$

(2.9)' da verilmiş tanım şu şekilde yorumlanabilir: Herhangi bir PA grafına ilişkin R erişilebilir durumlar kümesi, söz konusu PA grafında mümkün olan bütün geçiş ateşlemeleri tek tek yapıldıktan sonra elde edilen durumlardan oluşur[1].

Örnek 2.3: Şekil 2.12'de verilen PA grafını inceleyelim.



$x_0 = [2,0]$ $x_1 = [1,1]$ $x_2 = [0,2]$

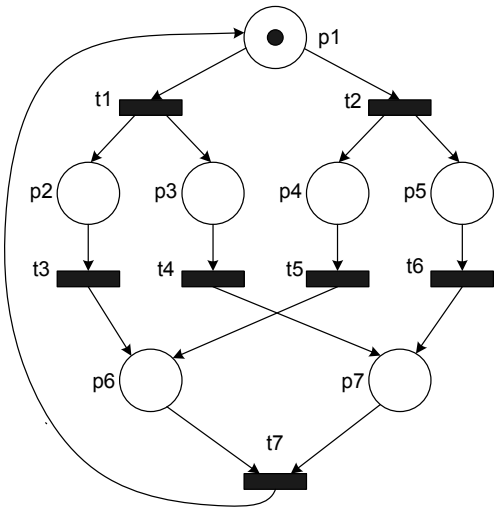
Şekil 2.12: Örnek bir PA grafı için erişilebilir durumlar

Görüldüğü gibi $x_0 = [2,0]$ ilk durumlu PA grafında sadece $x_1 = [1,1]$ ve $x_2 = [0,2]$ durumlarına erişmek mümkün olmaktadır. O halde Şekil 2.12'deki PA grafına ilişkin R erişilebilir durumlar kümesi $R = \{[2,0],[1,1],[0,2]\}$ olarak gösterilebilir.

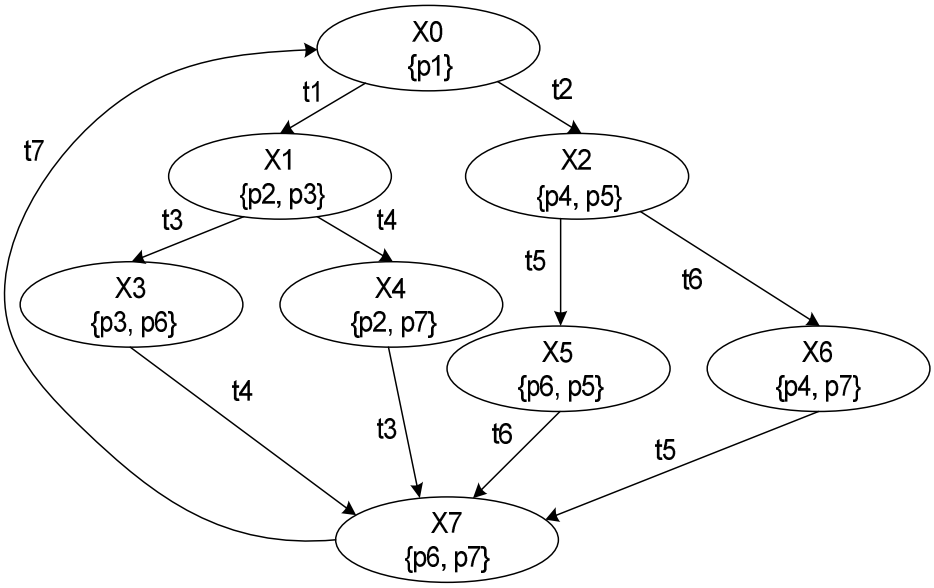
2.5.2 Erişilebilirlik grafı

Bir $N = (P,T,F,w)$ PA ve x_0 ilk işaretlemesi verilmiş olsun $R(N,x_0)$ ile elde edilmiş erişilebilirlik kümesinin elemanlarını grafiksel olarak ifade etmek için kullanılan grafa erişilebilirlik grafı denir[4].

Örnek 2.4: Aşağıda Şekil 2.13'de PA grafı görülmektedir. Erişilebilirlik kümesine ilişkin erişilebilirlik grafı ise Şekil 2.14'de gösterilmiştir.



Şekil 2.13: Örnek PA grafı



Şekil 2.14: (Şekil 2.13)'deki PA'nın erişilebilirlik grafı

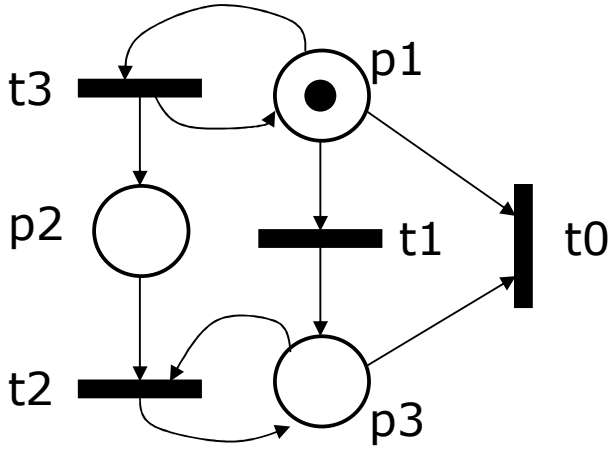
2.6 Analiz Yöntemleri

Petri ağı analizlerinde kullanılan iki temel analiz yöntemi geliştirilmiştir. Petri ağları ile iligili ilk yöntem yapısal analize dayanan erişilebilirlik ağacı yöntemidir, ikinci yöntem ise lineer cebirsel denklemlere dayanan durum denklemleri yöntemidir.

2.6.1 Erişilebilirlik ağacı yöntemi

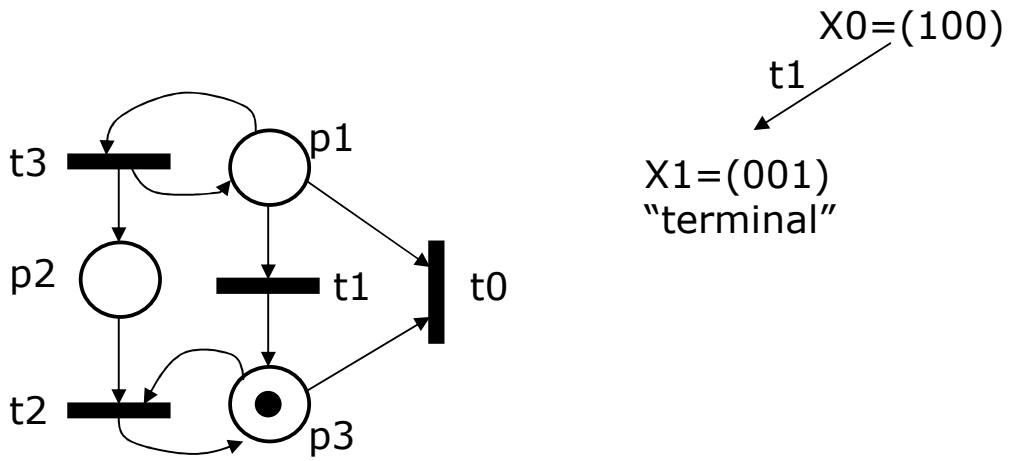
Erişilebilirlik ağacı bir Petri ağındaki erişilebilen işaretlemeler kümesini ifade eder. Bir örnek ile erişilebilirlik ağacı yöntemini açıklarsak;

Örnek 2.5:



Şekil 2.15: Erişilebilirlik ağacı örneği

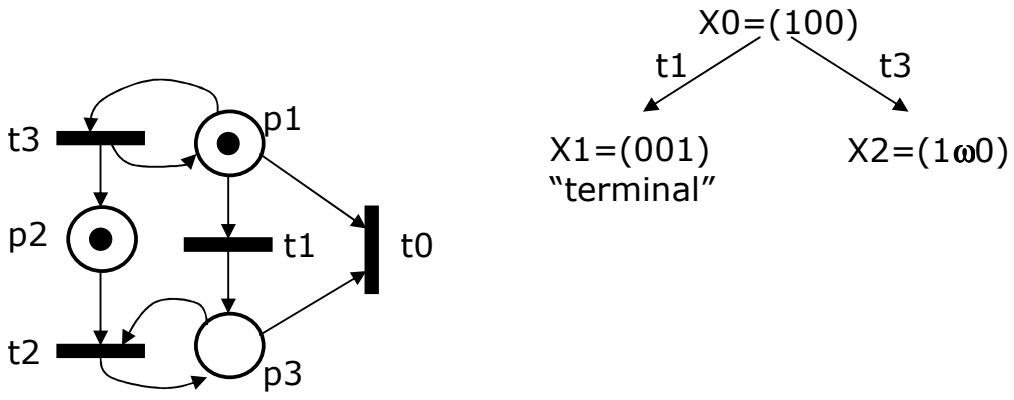
Şekil 2.15'te verilen PA'da ilk işaretleme altında t_1 ve t_3 geçişlerinin izinli olduğunu görüyoruz. Bu geçişlerin ateşlenmesi ile dallara ateşlenen geçişlerin bilgisi yazılarak yeni işaretlemeler elde edilir. Şekil 2.16'da t_1'in ateşlenmesi ile oluşan ağacın ilk bölümü gözükmektedir.



Şekil 2.16: t_1 ateşlenmesi ile oluşan erişilebilirlik ağacının 1. adımı

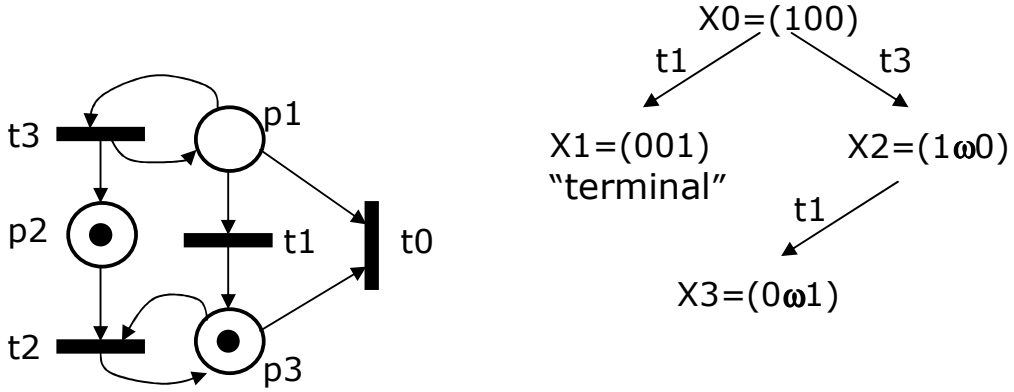
Burada dikkat edilirse t_1 geçişinin ateşlenmesi ile erişilebilir durumlar bittiği için X1 ile ulaşılan işaretlemeye 'terminal' işaretlemesi denir. t_3'ün ateşlenmesi durumunda (110) işaretlemesi elde edilir. Fakat verilen PA örneğine bakılacak olursa p_1 ile t_3 arasında bir çevrim olduğu için t_3 geçişi sonsuz kez ateşlenebilir. Bu durumda erişilebilirlik ağacı sonsuz elemanlı olur. Ağacı kullanışlı hale getirmek amacıyla boyutu sonlu hale getirilir.

Şekil 2.15'te görüldüğü gibi t_3 sonsuz kere ateşlenebilir, her ateşleme sonucu p_1'deki jetonların sayısı 1 artacaktır. Bu şekilde döngüye girip sonsuza varan yerlerdeki jeton sayısı 'ω' ile gösterilir. 'ω' sonsuz anlamına gelip gerektiği kadar büyük yapılabilecek bir sayıyı ifade etmektedir. Şekil 2.17'de t_3'ün ateşlenmesi ile elde edilen ağacın 2. dalı görülmektedir.



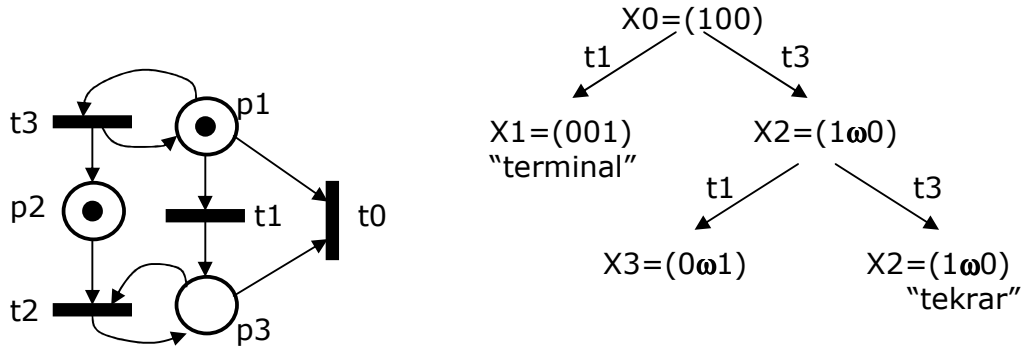
Şekil 2.17: t_3 ateşlenmesi ile oluşan erişilebilirlik ağacının 2. adımı

X2(1ω0) işaretlemesinden t_1 geçişinin izinli olma koşullarının sağlandığını görüyoruz. t_1 geçişinin ateşlenmesi ile şekil 2.18'de görülen X3=(0ω1) işaretlemesi elde edilir.



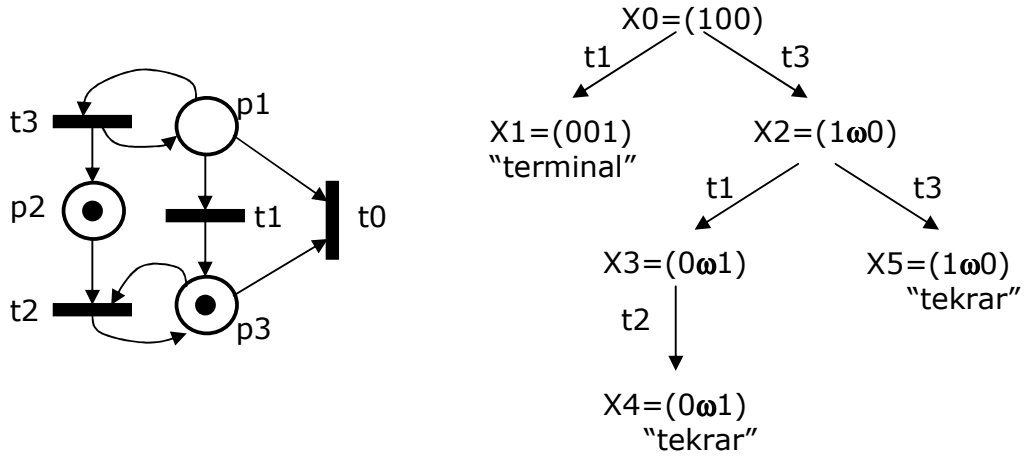
Şekil 2.18: t_1 ateşlenmesi ile oluşan erişilebilirlik ağacının 3. adımı

Erişilebilirlik ağacı yönteminde ilgili geçişlerin ateşlenmesi ile daha önce elde edilen işaretlemeler elde edilebilir. Bu şekilde elde edilen düğümlere tekrarlanan düğümler denir ve ilkinden sonra oluşan düğümler ilkinde temsil edilir. Şekil 2.19'da t_3'ün ateşlenmesi ile aynı düğüm elde edildiği için 'tekrar' düğümü olarak belirtilmiştir.



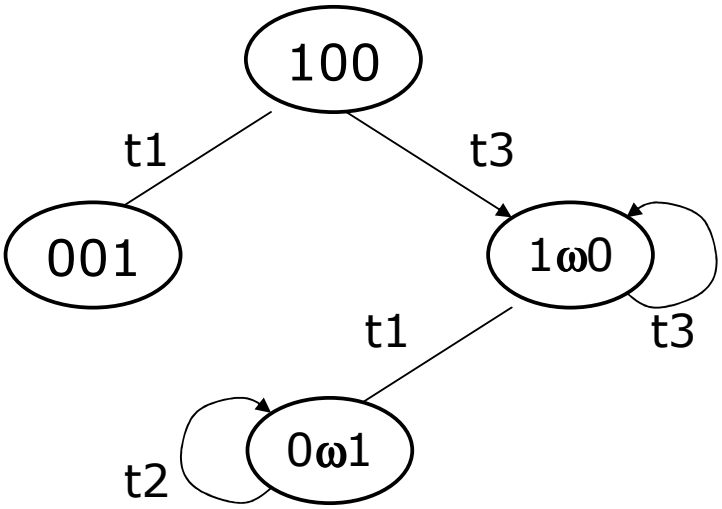
Şekil 2.19: t_3 ateşlenmesi ile oluşan erişilebilirlik ağacının 4. adımı

X3=(0ω1) işaretlemesinde t_2 geçişinin izinli olduğunu görüyoruz ve t_2 geçişinin ateşlenmesi ile X3 işaretlemesinde değişiklik olmayacağı için 'tekrar' düğümü elde ederiz. Bütün geçişlerin ateşlenmesi ile vermiş olduğumuz PA örneğinin olası bütün erişilebilecek durumlarına ulaşmış oluyoruz. Şekil 2.20'de olası erişilebilecek bütün durumlar görülmektedir.



Şekil 2.20: t_2 ateşlenmesi ile oluşan erişilebilirlik ağacının son adımı

Elde edilen erişilebilirlik ağacından hareketle erişilebilirlik grafını oluşturabiliriz. Erişilebilirlik grafında düğümler olası işaretlemeleri, dallarda ateşleme bilgilerini verir. Şekil 2.21'de elde edilen erişilebilirlik ağcının grafı görülmektedir.



Şekil 2.21: Erişilebilirlik grafı

2.6.2 Durum denklemleri ve durum-geçiş matrisi

Buraya kadar bir PA grafına ilişkin durumları grafiksel olarak nasıl elde edileceğinden bahsedildi. Ancak grafiksel yöntemin kullanılması karmaşık yapıdaki PA'larının incelenmesinde pratik bir yol değildir. Bu nedenle bir sonraki durum bilgisinin elde edilmesi için matematiksel yollara başvurulur. (2.6)'da verilmiş olan denklem incelenecek olursa, bir geçiş ateşlendiğinde bu geçişe ilişkin yerin bir sonraki durum değeri matematiksel olarak elde edilebilir. Buradan yola çıkarak PA'na ilişkin bir sonraki durum vektörü matematiksel olarak hesaplanabilir.

$x = [x(p_1), x(p_2), ..., x(p_n),]$ ilk durumuna sahip ve t_j geçişi ateşlenmiş olan bir PA'na ait $x' = [x'(p_1), x'(p_2), ..., x'(p_n),]$ bir sonraki durum vektörü şu şekilde elde edilebilinir[1]:

j. geçişinin olduğu sıradaki durumu ifade etmek için m bouytlu bir ateşleme (firing) vektörü aşağıdaki gibi tanımlanır[1]:

$$u = [0, ..., 0, 1, 0, ..., 0] \quad j \in \{1, 2, ..., m\} \qquad \textbf{(2.10)}$$

Burada u vektöründe j.'ci geçişin değeri 1 yapılarak ateşleneceğini belirtilir.

Petri ağında (mxn) boyutlu Durum-Geçiş matrisi (incidence matrix) A şu şekilde tanımlanır[1];

$$a_{ji} = w(t_j, p_i) - w(p_i, t_j) \qquad \textbf{(2.11)}$$

Bu ifadenin, denklem (2.6)'da gösterilen ağırlık farkına eşit olduğu görülebilir. Buradan hareketle A Durum-Geçiş matrisini kullanarak PA'na ilişkin bir sonraki durum vektörü şu şekilde yazılabilir;

$$x' = x + u.A \qquad \textbf{(2.12)}$$

(2.12) denkleminde;

x' : Bir sonraki durum vektörünü,

x : Başlangıç durum vektörünü,

u: Ateşleme vektörünü,

A : Durum-Geçiş matrisini

ifade etmektedir.

Denklem (2.12), u girişi sonucunda durum geçiş işlemini tanımlar. Daha önce tanımlanmış olan $f(x,t_j)$ geçiş fonksiyonun da

$$f(x,t_j) = x + u.A \qquad \textbf{(2.13)}$$

Eşit olduğu görülür. Fonksiyonda t_j ifadesi sıfırdan farklı olan u'daki j değerini gösterir. Durum denklemi geçiş işlemleri sonucunda değişen Petri ağının durumunu tanımlamaya yarar.

Örnek 2.6: Örnek 2.2'de, Şekil 2.8'de durumlarını grafiksel olarak bulduğumuz $x_o = [1,0,0,2,1]$ durumlu PA'nı inceleyelim

(2.11)'deki tanım ile Durum-Geçiş matrisi yazılacak olursa,

$$A = \begin{bmatrix} -1 & 1 & 1 & 2 & 0 \\ 0 & 0 & -1 & -1 & 0 \\ 0 & 0 & 0 & -2 & 1 \\ 0 & 0 & 1 & 1 & -1 \end{bmatrix}$$

t_4 geçişinin ateşlendiği zaman sahip olacağı x_1 durumunu incelersek,

$u = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}$ olur. Buradan bir sonraki durum vektörü;

$$x' = x + u.A$$

$$x_1 = [1 \quad 0 \quad 0 \quad 2 \quad 1] + [0 \quad 0 \quad 0 \quad 1].\begin{bmatrix} -1 & 1 & 1 & 2 & 0 \\ 0 & 0 & -1 & -1 & 0 \\ 0 & 0 & 0 & -2 & 1 \\ 0 & 0 & 1 & 1 & -1 \end{bmatrix}$$

$$x_1 = [1 \quad 0 \quad 0 \quad 2 \quad 1] + [0 \quad 0 \quad 1 \quad 1 \quad -1] = [1 \quad 0 \quad 1 \quad 3 \quad 0]$$

olarak bulunur. Bulunan bu sonucun şekil 2.8'de verilen örnekte grafiksel olarak bulunan sonuçla aynı olduğu görülür. Aynı örnek için PA grafının t_1 geçişi ateşlendiği zaman sahip olacağı x_2 durumunu benzer şekilde hesaplayabiliriz.

$$x_2 = [1 \quad 0 \quad 1 \quad 3 \quad 0] + [1 \quad 0 \quad 0 \quad 0].\begin{bmatrix} -1 & 1 & 1 & 2 & 0 \\ 0 & 0 & -1 & -1 & 0 \\ 0 & 0 & 0 & -2 & 1 \\ 0 & 0 & 1 & 1 & -1 \end{bmatrix}$$

$$x_2 = [1 \quad 0 \quad 1 \quad 3 \quad 0] + [-1 \quad 1 \quad 1 \quad 2 \quad 0] = [0 \quad 1 \quad 2 \quad 5 \quad 0]$$

Görüldüğü gibi Petri ağında izlenen yol geçiş sırasına göre $\{t^1, t^2, ...\}$ (t^k k. Oluşan geçiş)durumlarının sıralanmasıdır. $\{x_0, x_1, ...\}$

x_0 durumu verildiğinde sırayla tüm durumlar:

$$x_{k+1} = f(x_k, t_k) = x_k + u_k.A \qquad \textbf{(2.14)}$$

Denklemi ile üretilebilir.(u_k , k. Geçişin ateşlendiğini belirtir.)

3. PETRİ AĞLARININ ÖZELLİKLERİ

PA ile çok çeşitli sistemlerin (özellikle interaktif olayların meydana gelebileceği sistemler) modellenebilir. PA'larının eş zamanlı olayların modellenmesinde güçlü bir analiz metotudur[2].

PA'larının sistemlerin modellenmesinde kullanılmasının nedenlerinden biri sağladığı güçlü analiz imkânlarıdır. PA grafı analiz edilerek sistemlerin doğru bir şekilde modellenmiş olup olmadığını kontrol edilebilinir. Bu amaçla modellenen sistemlerin davranışları derinlemesine incelenebilir ve sistemlerde ortaya çıkan problemlerin nedenleri etkin bir şekilde araştırılabilinir. PA'larının analizleri için birçok teknik geliştirilmiştir. Fakat analizi gereken birçok problem hala açıktır. Geliştirilen analiz tekniklerinin kullanışlılığı değerlendirip ne tip problemlerin çözüme kavuşturulması gerektiği göz önüne almalıyız. Bu bölümde, bu zamana kadar PA'larının literatürde bahsedelmiş özelliklerini inceleceğiz.

3.1 Petri Ağlarının Davranışsal Özellikleri

3.1.1 Güvenlik (safeness) özelliği

Donanım cihazlarını modelleyen PA'nın güvenlik özelliği çok önemlidir. PA'da bir yerdeki jeton sayısı 1'i kesinlikle geçmiyorsa o yer güvenlidir. Bu durumda PA'daki yerlerde jeton sayısı 0 veya 1'dir. Bu şekilde yerler mantıksal bir durumu ifade edebilir(jeton varsa doğru yoksa yanlış şeklinde)[1].

Tanım 3.1: PA'da ki bir yer $p_i \in P \quad N = (P, T, I, O)$ eğer tüm yeni durumlar için

$$x' \in R(N, x), \quad x'(p_i) \leq 1 \qquad \textbf{(3.1)}$$

ise ilk durum x ile güvenlidir.

Bir PA'da tüm yerler güvenli ise Petri Ağı güvenlidir[2].

3.1.2 Sınırlılık (boundedness) özelliği

Klasik sistem teorisinde bir durum değişkeninin çok büyümesi o sistemdeki karasızlığı işaret eder. Aynı şekilde PA'nın bir yerinde biriken jetonlar sistemi karasızlığa götürür. Sınırlılık (boundedness) özelliği, güvenlik özelliğinin

genelleştirilmiş bir halidir. PA grafında bir yerdeki jeton sayısı, tüm yeni durumlar için belirli bir k pozitif tam sayısını (k>1) aşmıyorsa bu yer k-sınırlıdır. Görüldüğü gibi sınırlılık özelliği bir yerdeki jeton sayısının belirli bir değeri aşmaması koşuluna bağlıdır. Eğer PA'daki bütün yerler için bu koşul sağlanıyorsa söz konusu PA k- sınırlıdır[1].

Tanım 3.2: N Petri ağında x_0 başlangıç durumundaki $p_i \in P$ yeri eğer ulaşılabilir tüm durumlar için

$$x \in R(N) \quad x(p_i) \leq k \tag{3.2}$$

ise k-sınırlıdır denir.

Eğer PA'ki tüm yerler sınırlı ise o ağa 'sınırlıdır' denir. Eğer bir yer '1-sınırlı' ise o yer aynı zamanda güvenlidir. PA ile modellenen bir ayrık olay sisteminde ağın sınırlı olup olmadığı kontrol edilir ve bir sınırlılığa karar verilir. Eğer sınırlılık sağlanmıyorsa model değiştirlmelidir. Çünkü kararsızlığa neden olabilir[1].

3.1.3 Canlılık (liveness) özelliği

Bir PA'da erişilen her bir yeni durumda eğer ateşlenebilir olan en az bir geçiş varsa PA canlıdır(livenes). Diğer taraftan, eğer hiçbir geçişin ateşlenemeyeceği bir duruma gelinmişse kilitlenme (deadlock) gerçekleşir. Bu durumda canlı bir PA'da, ilk durumdan itibaren seçilen ateşleme sırası ne olursa olsun kilitlenme hiçbir zaman gerçekleşmeyecektir.

Tanım 3.3: x_0 başlangıç durumlu bir N Petri ağında, x_0 durumundan ulaşılan yeni durumlarda her zaman ateşlenebilir bir geçiş bulunursa o ağa canlıdır denir[1].

Canlılık özelliği ideal sistemlerde aranan bir özelliktir. Ancak büyük sistemler için bu özelliğin var olup olmadığı araştırılması bir hayli zordur. Bu amaçla canlılık özelliği kısımlara bölünmüş ve değişik seviyeler için tanımlanmıştır.

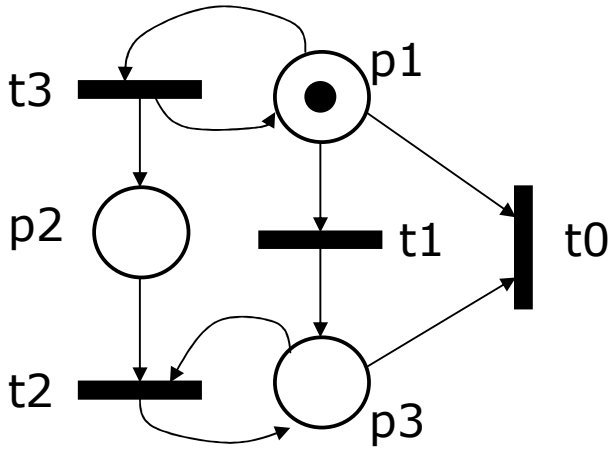
Bu seviyeler şöyle özetlenebilir.

- **LO-canlı:** PA'daki bir geçiş hiçbir zaman ateşlenemiyorsa bu geçiş ölü olarak tanımlanır ve LO-canlıdır.
- **L1-canlı:** PA'daki bir geçiş ilk durumdan itibaren en az bir kere ateşlenebiliyorsa bu geçiş L1-canlıdır.

- **L2-canlı:** PA'daki bir geçiş ilk durumdan itibaren bir k pozitif tamsayısı (k>1) kadar ateşlenebiliyorsa bu geçiş L2-canlıdır.
- **L3-canlı:** PA'daki bir geçiş herhangi bir ateşleme sırası için sonsuz sayıda ateşlenebiliyorsa bu geçiş L3-canlıdır.
- **L4-canlı:** PA'daki bir geçiş erişilen tüm yeni durumlar için L1-canlı ise bu geçiş L4-canlıdır.

Dikkat edilecek olursa L4-canlılık özelliği en üst seviyeli canlılık özelliğidir ve L1, L2 ve L3 seviyeli canlılık özelliklerini kapsamaktadır. Şekil 2.15'te erişilebilirlik ağacını elde ettiğimiz PA'nı canlılık acısından incelenmesi Örnek 3.1'de verilmiştir.

Örnek 3.1: L1, L2 ve L3-canlılık seviyelerini içeren PA grafı



Şekil 3.1: Canlılık özelliğine ilişkin örnek PA grafı

Şekil 3.1'deki PA grafında t_0 geçişi hiçbir zaman ateşlenmeyeceğinden ölü bir geçiştir, yani LO-canlılık seviyesindedir. t_1 geçişi sadece bir kere ateşlenebilir ve bu ateşlemeden sonra bir daha ateşlenmez. Bu durumda t_1 geçişi L1-canlıdır. t_2 geçişi bir kereden fazla sayıda ateşlenebilir; ancak sonsuz sayıda ateşlenemeyecektir. Bu nedenle t_2 geçişi L2-canlıdır. Son olarak dikkatli incelenecek olursa t_3 geçişi sonsuz sayıda ateşlenebilir, bundan dolayı L3-canlıdır.

3.1.4 Tutarlılık (conservation) özelliği

Bir PA'da ilk durumdaki toplam jeton sayısı eğer erişilen her yeni durum için sabit kalıyorsa söz konusu PA koruyucudur (conservative). Bu durum tutarlılık (conservative) özelliği olarak adlandırılır. Eğer jetonların sistemdeki bir kaynağı

temsil ettiği düşünülürse bu dikkat edilmesi gereken bir özelliktir; çünkü jeton sayısının korunması kaynakların kullanıldıktan sonra sisteme geri iade edildiği ve boşa harcanmadığını gösterir[2].

x_0 başlangıç durumlu bir N Petri ağında tüm durumlar için $x \in R(N)$ jeton sayısı sabitse o ağ koruyucudur[1].

Tutarlılık özelliğinin genelleştirilmesi için yerlere dayanan ağırlık vektörü tanımlanması gerekir.

Tanım 3.5: PA grafındaki yerlere $p_1, p_2, ..., p_n$ karşılık gelen ($\gamma_i \geq 0$) koşulunu sağlayan vektöre 'yer ağırlık vektörü' denir ve şu şekilde gösterilir.

$$\gamma = [\gamma_1, \gamma_2, ..., \gamma_n] \quad \textbf{(3.3)}$$

Tutarlılık özelliği bu vektörde belirtilmiş ağırlık değerlerine göre bağıl olarak tanımlanır. Bu ağırlık değerleri PA'daki dalların ağırlık değerleriyle karıştırılmamalıdır.

Tanım 3.6: x_0 başlangıç durumlu bir N Petri ağında, erişilen tüm yeni durumlar için aşağıdaki koşul sağlandığı taktirde, $\gamma = [\gamma_1, \gamma_2, ..., \gamma_n]$ vektöründeki belirtilmiş ağırlık değerlerine göre koruyucudur[1].

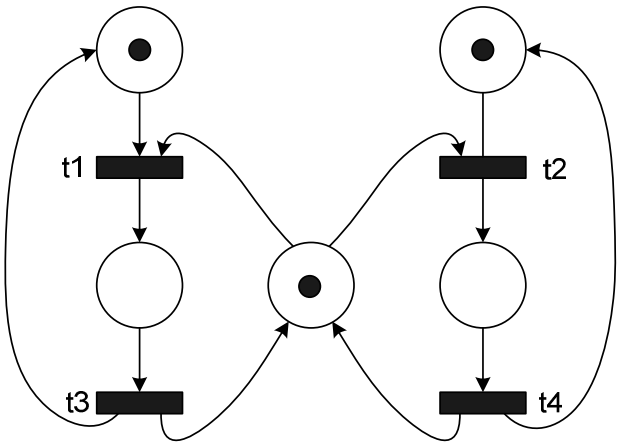
$$\sum_{i=1}^{n} \gamma_i x(p_i) = sabit \quad \textbf{(3.4)}$$

Bunun dışında bir de kesin (strict) koruyuculuk özelliği mevcuttur. Bir PA grafının kesin koruyucu olması için bu PA grafındaki her geçişin koruyucu olması gerekir. Bunun için PA grafındaki her bir geçiş için bu geçişe ilişkin toplam giriş sayısı ve toplam çıkış sayısı birbirine eşit olmalıdır. Bu koşul aşağıda belirtilmiştir.

$$\sum_{p_i \in P} w(t_j, p_i) = \sum_{p_k \in P} w(p_k, t_j), \quad p_i \in O(t_j) \quad ve \quad p_k \in I(t_j) \quad \textbf{(3.5)}$$

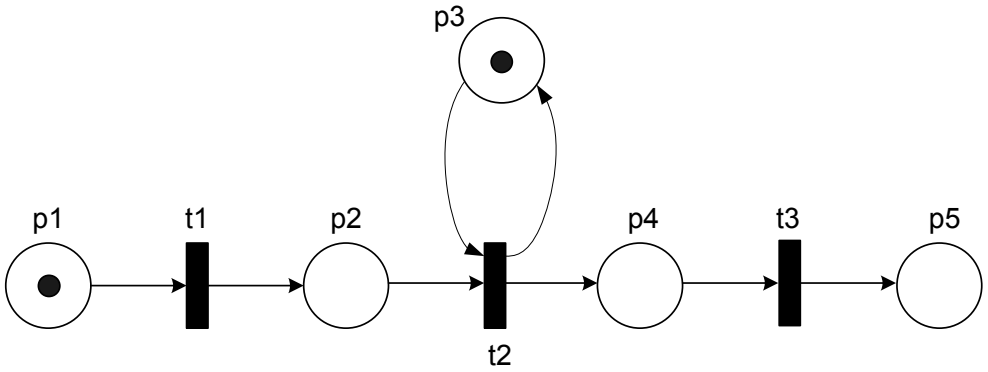
Aslında kesin koruyuculuk özelliği koruyuculuk özelliğinin bir alt kümesi niteliğindedir. Yani eğer bir PA grafı $\gamma = [1,1,...,1]$ ağırlık vektörüne göre koruyucuysa kesin koruyucu olarak anılır. Bu koşul aşağıda belirtilmiştir.

$$\sum_{i=1}^{n} x(p_i) = sabit \quad \textbf{(3.6)}$$



Şekil 3.2: Tutarsız Petri Ağı

Şekil 3.2'de PA koruyucu değildir. Çünkü t_1 ve t_2 geçişi jeton sayısını 1 azaltmaktadır. t_3 ve t_4 geçişleri jeton sayısını 1 arttırmaktadır.



Şekil 3.3: Tutarlı PA grafı örneği

Şekil 3.3'de tutarlı olan bir PA grafı verilmiştir. Dikkat edilecek olursa bu PA grafındaki her bir geçiş için o geçişe ait toplam giriş ve çıkış yerleri sayısı birbirine eşittir. Bu durum tablo 3.1'de belirtilmiştir.

Tablo 3.1. Tutarlı PA örneğindeki geçişlerin giriş ve çıkış yeri sayısı

Geçiş Adı	Giriş Yeri Sayısı	Çıkış Yeri Sayısı
t_1	$1\ \{p_1\}$	$1\ \{p_2\}$
t_2	$2\ \{p_2, p_3\}$	$2\ \{p_3, p_4\}$
t_3	$1\ \{p_4\}$	$1\ \{p_5\}$

3.1.5 Kapsanabilirlik (coverability) özelliği

Bu özellik daha önce incelediğimiz erişilebilir durumlar kavramının genelleştirilmiş bir halidir. Kapsanabilirlik (coverability) özelliği bir PA grafındaki herhangi bir geçişin ileriki durumlarda ateşlenip ateşlenmeyeceği ile ilgilenir.

Tanım 3.7: x_0 başlangıç durumlu bir N Petri ağında, $x' \in R(N, x_0)$ tanımlı olsun;

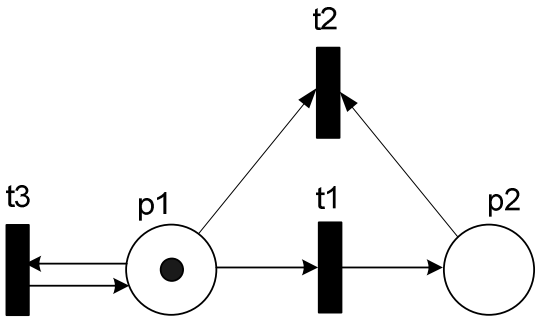
$$x'' \geq x' \quad x'' \in R(N, x_0) \tag{3.7}$$

Koşulunu sağlayan durum mevcutsa, x' durumu x'' durumu tarafından kapsanıyor denir.

Daha önceden de bahsedilidiği gibi bir geçişin ateşlenebilmesi için o geçişi ilişkin giriş yerlerindeki en azından birer jeton bulunmalıdır. Şimdi herhangi bir $y = [y(p_1), y(p_2), ..., y(p_n)]$ durumunun bir t_j geçişinin aktif hale gelmesi için yeter sayıda jeton içerdiğini ve PA grafının ilk başta x_0 durumunda olduğunu varsayalım. Eğer PA grafı x_0 durumundan, her $i = 1, 2..., n$ için $x(p_i) \geq y(p_i)$ olacak şekilde bir x durumuna erişilebiliyorsa x durumu y durumunu kapsıyor ya da y durumu kapsanabilirdir(coverable) denir[1].

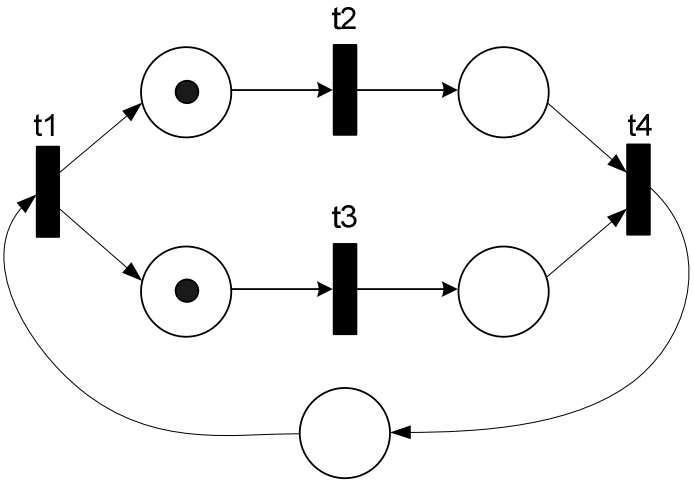
3.1.6 Devamlılık (persistence) özelliği

Bir PA'da eğer herhangi iki geçişten birinin ateşlenmesi diğer geçişin ateşlenmesini engellemiyor ise bu PA devamlılık (persistence) özelliğine sahiptir. Devamlılık özelliğine sahip bir PA'da eğer herhangi bir geçiş izinli hale gelmişse bu geçiş ateşleninceye kadar izinli halde kalır. Başka bir deyişle herhangi bir geçişin ateşlenmesi izinli hale gelmiş başka bir geçişin izinsiz hale gelmesine neden olmaz[1].



Şekil 3.4: Devamlılık özelliği olmayan örnek bir PA grafı

Şekil 3.4'te devamlılık özelliği olmayan bir PA grafı verilmiştir. Bu PA grafında ilk durumda t_1 ve t_3 geçişi izinli durumdadır. Bu durumda t_1 geçişinin ateşlenmesi izinli hale gelmiş olan t_3 geçişinin izinsiz hale gelmesine neden olur ve devamlılık özelliğine ters düşer[1].



Şekil 3.5: Devamlılık özelliği olan örnek bir PA grafı

Şekil 3.5'de şekil ilk durumda t_2 ve t_3 geçişleri izinli durumdadır. t_2 veya t_3 geçişlerinin herhangi birinin ateşlenmesi diğer geçişin izinli olma durumunda bir değişikliğe neden olmadığı için Şekil 3.5'teki örnek devamlılık özelliğine sahiptir.

4. PETRİ AĞLARI İLE MODELLEME

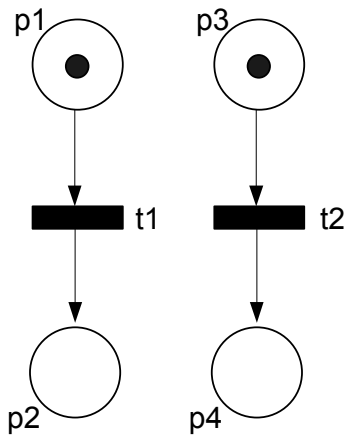
PA'larının kullanımındaki asıl amaç gerçek hayatta karşımıza çıkan sistemlerin modellenmesidir. Bu sistemlerin modellenmesinde kullanılan PA'lar, koşullar ve bu koşullara bağlı olarak gerçekleşen olaylardan oluşur. PA grafındaki yerler modellenen sistemin belirli kısımlarına ilişkin koşulları (örneğin bir makinenin boşta olması, çalışıyor olması ya da görevini yerine getiremiyor olması gibi) belirtirken PA grafındaki geçişler ise koşullara bağlı olarak bir olayın gerçekleşmesini (örneğin görevini yerine getirmeyen bir makinenin tamir edilmesi gibi) temsil eder. Bir olayın gerçekleşmesi yani PA grafındaki ilgili geçişin ateşlenebilmesi için bu olaya ilişkin tüm koşulların sağlanmış olması gerekir. Başka bir deyişle bu olayla ilgili geçişe ait tüm giriş yerlerinde yeterince jeton bulunması gerekir. Bir olayın gerçekleşmesi PA grafının işaretleme vektörü yani o anki durumunu değiştirip başka bir duruma geçişine neden olur. PA grafındaki bir yerde bulunan jeton sayısı bu yere ilişkin koşulları temsil edebileceği gibi sistemin yapısına bağlı olarak ilgili olayın gerçekleşmesi için gerekli kaynak (resource) sayısınıda belirtebilir[5].

4.1 Petri Ağları Yapıları

Sistem modellenmesinde kullanılan PA yapıları aşağıdaki alt başlıklarda verilmiştir.

4.1.1 Paralellik (concurrency)

Bağımsız olarak gerçekleşebilen iki ya da daha fazla olayın modellenmesinde kullanılan PA yapısıdır. Bu PA yapısında olaylar birbirinden bağımsız olarak aynı anda gerçekleşebilir[5].

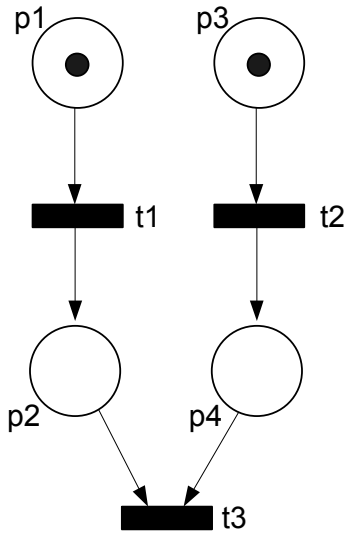


Şekil 4.1: Paralellik örneği

Şekil 4.1'de verilen PA grafında t_1 ve t_2 geçişleri aynı anda ateşlenmektedir[5].

4.1.2 Senkronizasyon (synchronization)

Bir geçişin gerçekleşebilmesi için iki ya da daha fazla koşulun sağlanmış olmasını gerektiren PA yapısıdır. Bir PA yapısında herhangi bir geçiş ancak tüm koşullar sağlandığı zaman ateşlenebilir. Aksi takdirde eğer koşulların en az biri bile sağlanmıyorsa ilgili geçiş ateşlenmez[5].



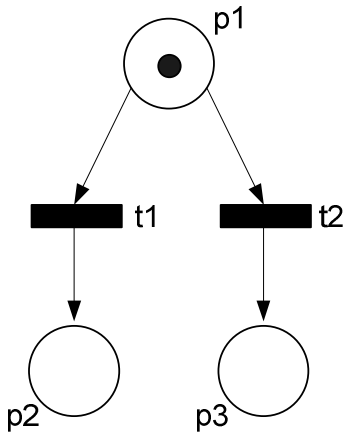
Şekil 4.2: Senkronizasyon örneği

Şekil 4.2'de verilen PA grafında t_3 geçişinin ateşlenebilmesi için hem t_1 hem de t_2 geçişinin ateşlenmiş olması gerekmektedir[5].

4.1.3 Çakışma (conflict)

Bir sistemde aynı kaynağı paylaşan iki ya da daha fazla olay olabilir. Bu durumda kaynağa aynı anda yalnızca bir olay erişmeli ve diğer olayların söz konusu kaynağa olan erişimleri engellenmelidir[5].

İki ya da daha fazla olay tarafından paylaşılmış olan kaynağa aynı anda yalnızca bir olayın erişmesinin sağlandığı PA yapısı çakışma (conflict) olarak adlandırılır[5].

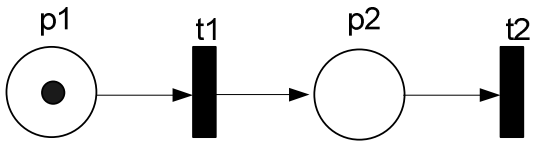


Şekil 4.3: Çakışma örneği

Şekil 4,3'te verilen PA grafında p_1 yeri hem t_1 hem de t_2 geçişleri tarafından paylaşılmakta olan ortak bir kaynağı temsil etmektedir. Bu durumda t_1 ve t_2 geçişleri aynı anda gerçekleşemez. Başka bir deyişle ya t_1 geçişi ya da t_2 geçişi ateşlenecektir. Bunun seçimi sistemle ilgili harici koşullara bağlı olacaktır[5].

4.1.4 Sıralama (sequentiality)

PA'ları ile sistemlerin modellenmesinde olaylarının sırasının önemli olduğu durumlar olabilir. Bu durumda sıralama (sequentiality) adı verilen PA yapısı kullanılır. Böylelikle olayların gerçekleşme sırası belirlenip modellenebilir[5].

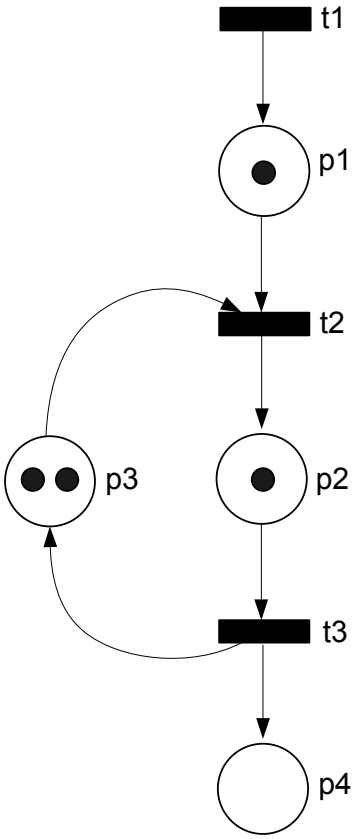


Şekil 4.4: Sıralama örneği

Şekil 3,4'te verilen PA grafında t_1 geçişi her zaman t_2 geçişinden önce gerçekleşir. Böylelikle olayların gerçekleşme sırası modellenmiş olur[5].

4.1.5 Kaynak sınırlama (limited resources)

Dağıtık (distributed) ve üretim (manufacturing) gibi sistemlerin performansını etkileyen bir faktör mevcut kaynakların sınırlı olmasıdır. Kaynakların tükenmesi işlemlerin yürümesini engelleyip sistemlerin çalışmasını durdurabilir. Bu tür sistemlerin modellenmesinde kaynak sınırlama (limited resources) adı verilen PA yapısı kullanılır[5].



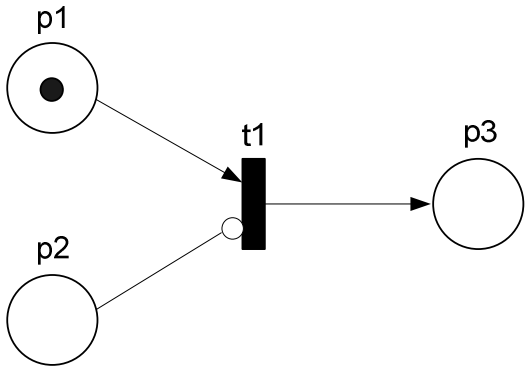
Şekil 4.5: Kaynak sınırlama örneği

Şekil 4.5'te verilen PA grafında kaynakları sınırlı bir kuyruk modellenmiştir. Bu kuyruğun bir alışveriş merkezine ait alışveriş kuyruğu olduğu düşünülebilir. O halde PA grafında p_3 yeri bu kuyrukta boşta olan pozisyon sayısını belirtirken p_2 yeri ise dolmuş pozisyon sayısını göstermektedir. Eğer dikkat edilecek olursa p_2 ve p_3 yerlerindeki toplam jeton sayısı sabit olup kuyruktaki toplam boş pozisyon sayısını ifade etmektedir. Bu örnekte kuyruktaki toplam boş pozisyon sayısı üçtür. p_1 yeri gelen müşterileri ve p_4 yeri ise işini bitirip kuyruktan ayrılan müşterileri temsil etmektedir. t_2 geçişi bir müşterinin kuyruğa girmesini ya da başka bir deyişle

bir pozisyonun bir müşteri tarafından doldurulmasını temsil etmektedir. t_2 geçişinin ateşlenebilmesi için p_1 ve p_3 yerlerinde en az bir jeton bulunmalıdır. Başka bir deyişle kuyruğa giren bir müşteri olmalı ve kuyrukta bu müşteri için boşta bir pozisyon bulunmalıdır. t_3 geçişi bir müşterinin işini bitirip kuyruktan ayrılmasını temsil etmektedir. Bunun için p_2 yerinde en az bir jeton bulunmalıdır, yani kuyrukta en az bir müşteri olmalıdır. t_3 geçişi ateşlendiğinde bir müşteri kuyruktan ayrılır ve boş pozisyon sayısı bir artar, yani p_2 yerinden p_3 yerine bir jeton geçmiş olur[5].

4.1.6 Bozucu (inhibitor) dallar

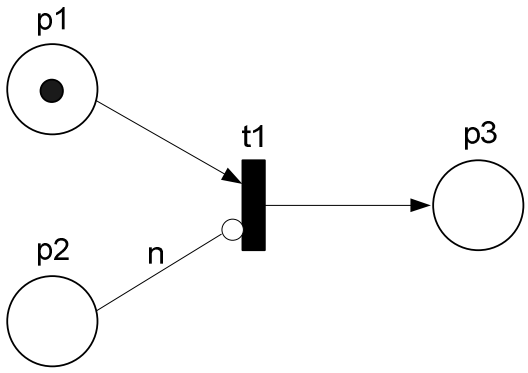
PA grafında p_i yerinden t_j geçişine çizilmiş bir bozucu (inhibitor) dal t_j geçişinin ateşlenme mekanizmasını şu şekilde etkiler; eğer p_i yerinde en az bir jeton varsa, t_j geçişi izinli hale gelmiş bile olsa ateşlenemez. Bozucu dallar, ucu boş daire şeklinde biten dallar ile gösterilir[5].



Şekil 4.6: Bozucu dal örneği

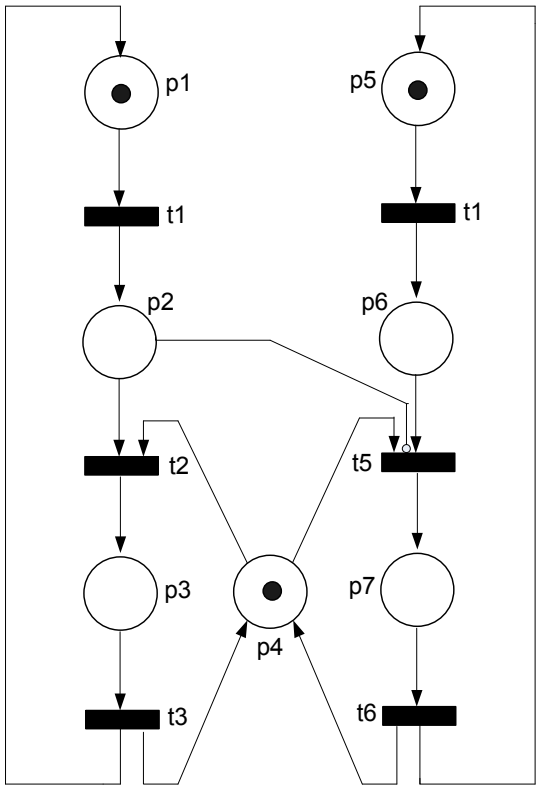
Şekil 4,6'da verilen örnekte p_2 yerinden t_1 geçişine çizilmiş ucu boş daire şeklinde biten dal bir bozucu daldır. p_1 yerinde bir jeton bulunduğu için t_1 geçişi izinli durumdadır ve p_2 yerinde herhangi bir jeton bulunmadığı için t_1 geçişi ateşlenebilir. Eğer p_2 yerinde en az bir jeton bulunsaydı t_1 geçişi ateşlenemeyecekti[5].

Şekil 4.7'de verilen örnekte ise p_2 yerinden t_1 geçişine çizilmiş olan n adet bozucu dal mevcuttur. Bu durumda eğer p_2 yerinde en az n adet jeton bulunursa t_1 geçişi ateşlenemez. Diğer durumlarda t_1 geçişi ateşlenebilecektir[5].



Şekil 4.7: Çoklu bozucu dal örneği

Bozucu dalların kullanılmasındaki asıl amaç çakışma durumunda olan iki ya da daha fazla geçişe bir öncelik atamak ve atanan öncelik sırasına göre bu geçişlerin ateşlenmesini sağlamaktır. Şekil 4.8'de bu durumu ele alan bir örnek verilmiştir[5].



Şekil 4.8: Geçişlere öncelik atama

Şekil 4.8'de verilen PA grafında p_2 yerinden t_5 geçişine çizilmiş bir bozucu dal görülmektedir. İlk durumda sadece t_1 ve t_4 geçişleri izinli durumdadır ve paralel olarak ateşlenebilir. t_1 ve t_4 geçişleri ateşlendikten sonra p_2 ve p_6 yerleri birer jeton

kazanacağından ve p_4 yerinde bir jeton bulunduğundan hem t_2 hem de t_5 geçişi izinli durumda olacaktır. Bu durumda t_2 ve t_5 geçişi p_4 yerindeki kaynağı ortak olarak kullandığından çakışma içindedir. Çakışma durumundaki bu geçişlerden birine öncelik vermek için bozucu dal kullanılmaktadır. Örnekteki bozucu dal p_2 yerinde bir jeton bulunmasından dolayı t_5 geçişinin ateşlenmesini engeller ve t_2 geçişinin ateşlenmesine olanak sağlar. Başka bir deyişle bozucu dal t_2 geçişini t_5 geçişinden öncelikli konuma getirir. Çakışma durumunun olması durumunda öncelikli t_2 geçişi ateşlenir, t_5 daha sonra ateşlenecektir[5].

4.2 Petri Ağlarının Sınıflandırılması

PA'ları yapısal özelliklerine göre bir takım sınıflara ayrılmıştır. Böylelikle PA'ların analizi, ait oldukları sınıflara özel yaklaşımlar kullanılarak daha etkin bir şekilde yapılabilir. Bu sınıfları sıralayacak olursak[6]:

- **Durum Makinesi-DM (State Machine-SM):** Bu sınıftaki PA'larında tüm geçişlerin sadece tek bir giriş ve çıkış yeri mevcuttur.

  Durum makinesi sınıfında bulunan PA'larında yerler önem kazanmaktadır. Bu tip ağlarda bir yerde bulunan tek bir jeton birden fazla geçişi aktif duruma getirip çakışmaya neden olabilir[6].

- **İşaretli Graf-İG (Marked Graph - MG):** Bu sınıftaki PA'larında tüm yerlerin sadece tek bir giriş ve çıkış geçişi mevcuttur[6].

  İşaretli graf sınıfında bulunan PA'larında geçişler önem kazanmaktadır. Bu tip ağlarda tek bir geçişin birden fazla giriş ve çıkış yeri olabilir. Bu sınıfa ait PA'ları senkronizasyona izin verirler[6].

- **Serbest Seçim Ağları-SSA (Free Choice - FC):** Bu sınıftaki PA'larında bir p yerini t geçişine bağlayan her dal için aşağıdaki koşullardan biri sağlanır:

  $\Rightarrow$ t geçişi, p yerine ait tek çıkış geçişidir (çakışma yok), veya

  $\Rightarrow$ p yeri, t geçişine ait tek giriş yeridir (senkronizasyon yok).

Durum makinesi ve işaretli graf sınıflarına giren bütün PA'ları aynı zamanda serbest seçim ağları sınıfına da girer. Durum makinesi sınıfına giren PA'larında senkronizasyon söz konusu değil iken işaretli graf sınıfına giren PA'larında ise

çakışma durumu söz konusu değildir. Serbest seçim ağları sınıfına giren PA'larında hem senkronizasyon hemde çakışma durumları söz konusu olabilir. Serbest seçim ağları sınıfında bulunan PA'larında eğer bir p yerine ait bir çıkış geçişi izinli hale gelmişse söz konusu p yerine ait tüm çıkış geçişleri izinli hale gelmiş olur[6].

- **Genişletilmiş Serbest Seçim Ağları-GSSA (Extendent Free Choice - EFC):** Bu sınıftaki PA'larında bir p yerini t geçişine bağlayan her dal için, t geçişine ait bütün giriş yerlerindeki p yerine ait bütün çıkış geçişlerine çizilmiş bir dal mevcuttur[6].

  Eğer ;

$$O(p_1) \cap O(p_2) \neq 0 \quad ise \quad O(p_1) = O(p_2) \tag{4.1}$$

  olur.

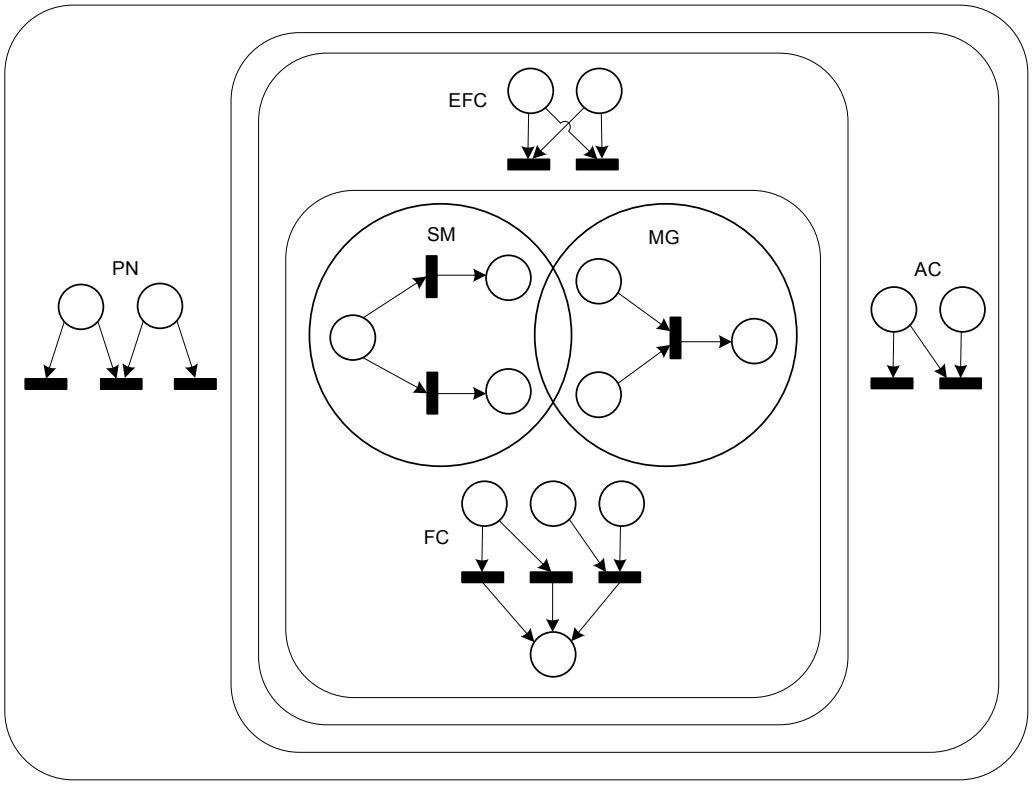
- **Asimetrik Seçim Ağları-ASA (Asymmetric Choice - AC):** Bu sınıftaki PA'larında bütün p_1 ve p_1 yer çiftleri için aşağıdaki koşul sağlanır[6].

  Eğer**;**

$$O(p_1) \cap O(p_2) \neq 0 \quad ise \quad O(p_1) \subseteq O(p_2) \quad veya \quad O(p_1) \supseteq O(p_2) \tag{4.2}$$

  olur.

Şekil 4.9' da PA'rı sınıflarına ilişkin PA'ları ile bu sınıfların birbiri ile olan ilişkisi gösterilmiştir.



Şekil 4.9: PA sınıfları

5. PETRİ AĞLARINDA SAVAK(SIPHON) VE KAPAN(TRAP) YAPILARI

Savak ve kapanlar PA'da belli kuralları sağlayan yerlerin bir araya gelmesiyle oluşan yer kümeleridir[3].

Savaklar ve kapanlar PA'nın işaretlemesi ile ilgili yapısal özellikleri tanımlamada kullanılırlar. Bu nedenle PA'nın dinamik davranışını belirleyen yapılar oldukları söylenebilir.

Bu bölümde Petri Ağlarının yapısal özelliklerinden olan savak(siphon) ve kapanlar(trap) hakkında tanımlamalar ile canlılık ve ulaşılabilirlik için savak ve kapanlara dayanan sonuçlar verilecektir.

PA'larını bölüm 2.3'te anlatılan giriş ve çıkış ifadeleri ile (P, T, I, O, X_0) olarakta ifade edebiliriz[11].

Burada;

P: m $yerden$ $oluşan$ $küme$
T: n $geçişten$ $oluşan$ $küme$
$I = \{i_{kj}\}$ mxn $boyutlu$ $giriş$ $matrisi$
$O = \{o_{kj}\}$ mxn $boyutlu$ $çıkış$ $matrisi$
$X_0 : P \to N$ ilk $işaretleme$

I giriş matirisinin i_{kj} elemanları p_k yerinden t_j geçişine olan dalın ağırlığını gösterir. Aynı şekilde O çıkış matrisinin o_{kj} elemanlarıda t_j geçişinden p_k yerine olan dalın ağırlığını gösterir.

5.1 Petri Ağlarında Savak(Siphon) ve Kapan(Trap) Yapıları

Tanım 5.1: N petri Ağının boş olmayan alt kümesi S;

Eğer;

$$\begin{aligned} &\bullet S \subseteq S \bullet \\ &\bullet S = \{t_j \in T \mid o_{kj} \neq 0, p_k \in S\} \\ &S \bullet = \{t_j \in T \mid i_{kj} \neq 0, p_k \in S\} \end{aligned} \qquad \textbf{(5.1)}$$

koşulunu sağlıyorsa **savak(siphon)** olarak adlandırılır[9,10]. S'te çıkış yerine sahip her bir geçiş aynı zamanda S'te giriş yerinede sahiptir.

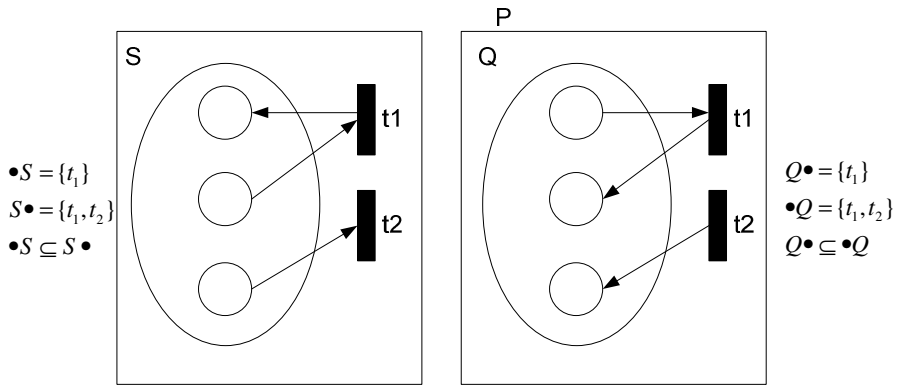
N petri Ağının boş olmayan alt kümesi Q;

Eğer;

$$Q\bullet \subseteq \bullet Q$$
$$\bullet Q = \{t_j \in T \mid o_{kj} \neq 0, p_k \in S\} \quad \textbf{(5.2)}$$
$$Q\bullet = \{t_j \in T \mid i_{kj} \neq 0, p_k \in S\}$$

koşulunu sağlıyorsa **kapan(trap)** olarak adlandırılır[9,10]. Q'da giriş yerine sahip her bir geçiş aynı zamanda Q'da çıkış yerinede sahiptir.

Şekil 5.1 a'da savak görülmektedir[10]. Jeton sayısı t_1 geçişinin ateşlenmesi ile değişmemekte, fakat t_2 geçişinin ateşlenmesi ile azalmaktadır. Bu yüzden savaklar, eğer herhangi bir X_k işaretlemesi altında jetonsuz kalırlarsa takip eden işaretlemelerde jetonsuz kalmayı devam ederler. Bu nedenle savaklar PA'da geçişlerin ölü olduğunu göstermek amacıyla kullanılabilirler



Şekil 5.1: PA yapıları (a) savak ve (b) kapan

Şekil 5.1 b'de gösterilen kapan da ise Jeton sayısı t_1 geçişinin ateşlenmesi ile değişmemekte, fakat t_2 geçişinin ateşlenmesi ile artmaktadır. Bu yüzden kapanlar, eğer X_k işaretlemesi altında işaretli ise (en az 1 tane jetona sahipse) takip eden işaretlemelerde işaretli kalmaya devam eder. Bu nedenle kapanlar PA'nın canlılık analizinde kullanılabilirler.

Savaklar ve kapanlar genel olarak sıradan Petri Ağlarının yapısal analizlerinde kullanılırlar. Burada savak ve kapanlara bağlı birkaç özellik verecek olursak:

- Sıradan bir PA'da kilitli(deadlock) bir sistem varsa bu durumda ağdaki jetonsuz yerlerin oluşturduğu yerler kümesi bir savak tanımlar.
- Kapanların bir defa jeton kazanması durumunda her zaman işaretli kaldıkları göz önüne alınırsa ve sıradan bir PA'da ilk X_0 işaretlemesi altında her kapan işaretli ise bu durumda sistem kilitlenmesizdir.
- X_0 işaretlemesi canlı bir sistemin ev işaretlemesi ise bu durumda her kapan işaretli olmalıdır, aksi durumda bir defa işaretlenen bir kapan her zaman canlı kalacağından asla ilk duruma ulaşılamaz[12].

Teorem 5.1: İşaretli bir Petri S savağı tanımlı olsun. Eğer S savağı X_0 ilk işaretlemesinde işaretli değil ise bu durumda erişilebilir hiçbir işaretleme altında işaretli duruma getirilemez[12].

İspat: $R(X_0)$**,** S'de hiçbir yeri izinli hale getirmeyen işaretlemelerin kümesi olsun. $R(X_0)$**'**da bir M_k işaretlemesini ve buna bağlı bir t geçişi tanımlansın, öyle ki t geçişi $X_k \rightarrow X_{k+1}$ durum değişimini sağlasın. Bu durumda $t \notin S\bullet$'tir. Çünkü X_k t geçişini izinli hale getirir ve X_k işaretlemesi S'teki hiçbir yeri işaretleyemez. S bir savak olduğundan $t \notin \bullet S$'i gerektirir. Bu yüzden hiçbir $p \in S$ yeri t geçişinin ateşlenmesiyle jeton kazanamaz.

Teorem 5.2: En az bir yeri içinde barındıran bir S savağı PA'da tanımlı olsun. İlk işaretlemede eğer hiçbir geçiş ölü değil ise bu durumda S'de yer alan ve ayrık olmayan bir yer işaretlidir[12].

İspat: S'de ilk işaretleme altında ayrık olmayan hiçbir yerin işaretli olmadığı varsayılsın. p, S'de yer alan bir yeri tanımlasın($p \in S$). Teorem 5.1 gereği p yeri hiçbir erişilebilir işaretleme altında işaretlenemez. Bu yüzden $(\bullet p \cup p\bullet)$'de yer alan tüm geçişler ölüdür. p ayrık bir yer olmadığından ilk işaretleme altında ölü bir geçiş vardır ki bu teoremin koşullarıyla çelişir.

Teorem 5.3: İşaretli bir PA'da Q kapanı tanımlı olsun. Eğer Q kapanı ilk işaretlemede işaretli ise bu durumda her erişilebilir işaretlemede de işaretli olacaktır[12].

İspat: $R(X_0)$, Q'da en az bir yeri izinli hale getiren işaretlemelerin kümesi olsun. Varsayım gereği ilk X_0 işaretlemeside $R(X_0)$'da yer almalıdır. X_k'da $R(X_0)$'da yer alan bir işaretleme olsun ve buna bağlı bir t geçişi tanımlansın, öyle ki t geçişi $X_k \rightarrow X_{k+1}$ durum değişimini sağlasın. Eğer $t \notin Q\bullet$ ise bu durumda X_k'da işaretli olan ve Q'da yer alan p yerleri işaretli kalır. Eğer $t \in Q\bullet$ ise aynı zamanda $t \in \bullet Q$ olmalıdır. Bu koşul kapanın tanımından ileri gelir($Q\bullet \subseteq \bullet Q$). Bu yüzden X_{k+1} işaretlemesi altında $(t\bullet \cap Q)$'da yer alan en az bir yer işaretli olmalıdır. Sonuç itibarıyla $R(X_0)$ erişilebilir tüm işaretlemeleri barındırır.

Teorem 5.4: Tersinir(reversible) ve ayrık yerler barındırmayan bir S kapanın bulunduğu PA göz önüne alınsın. İlk işaretlemede hiçbir geçiş ölü değil ise bu durumda Q kapanı işaretlidir[12].

İspat: $p \in Q$ olan bir yeri, t'de $(\bullet p \cup p\bullet)$ koşuluna uyan bir geçiş tanımlansın. t geçişi ilk işaretlemede ölü olmadığından, p yerini işaretleyen bir X_k işaretlemesi bulunmalıdır. Bu durumda Q kapanı erişilebilir bir işaretleme altında işaretli hale gelecektir. X_0 ilk işaretlemesi X_k işaretlemesine tersinirlik gereği erişilebilir olduğundan Q kapanı X_0 işaretlemesi altında işaretli olmalıdır.

Teorem 5.5: En az bir geçişi barındıran ve işaretli bir PA göz önüne alınsın. Ayrık yerler barındırmayan ve boş olmayan her savak ilk işaretlemede işaretlenmiş bir kapan barındırıyorsa bu durumda işaretlenmiş ağ kilitlenmesizdir(deadlock free)[12].

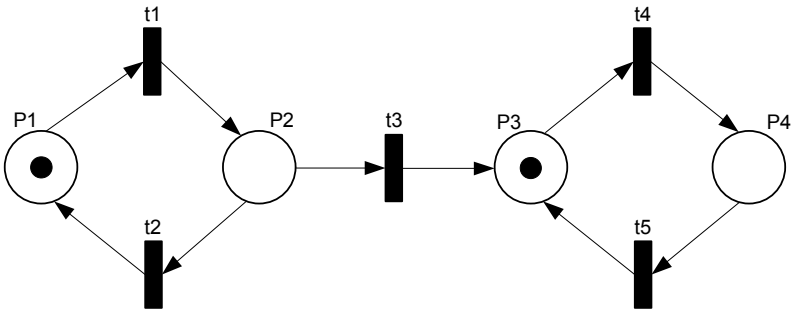
İspat: İşaretli bir PA'nın kilitlenebilir olduğu varsayılsın. X_k bu PA'da erişebilen ölü işaretlemeyi tanımlasın. S ayrık olmayan ve X_k'da işaretli olmayan yerlerin tanımladığı küme olsun. Bu durumda S'in boş olmayan ve ilk işaretlemede işaretli olmayan bir savak barındırdığı gösterilecektir.

X_k işaretlemesi altında her t geçişi ölüdür, bu yüzden işaretli olmayan giriş yerleri bulundurur. $S\bullet$ aynı durumda olan tüm t geçişlerinin kümesini tanımlar. Bu yüzden $(\bullet S \subseteq S\bullet)$'tir. S varsayım gereği ağda bir geçiş ve bu geçişe öncü bir yer bulundurduğundan boş küme değildir. Bu durumda savak tanımından S'in boş olmayan ve ayrık yerler barındırmayan bir savak olduğu sonucuna varılır. Yine varsayım gereği S savağı X_k işaretlemesi altında işaretli değildir. Bu yüzden S

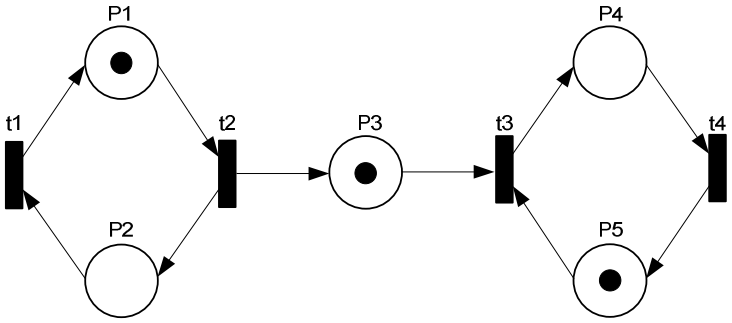
savağı X_k işaretlemesi altında bir kapan barındırmaz. İlk işaretlemede işaretli olan kapanlar her zaman canlı kaldığından, S ilk işaretleme altında bir kapan barındıramaz. Bu çelişki aranan sonucu verir.

Örnek 5,1'de savak ve kapanlara ait farklı özelliklere sahip iki PA görülmektedir[12].

Örnek 5.1:



Şekil 5.2: Savak-kapan yapılarına ait örnek PA1



Şekil 5.3: Savak-kapan yapılarına ait örnek PA2

Örnek 5.1'de verilen PA'nın savak ve kapanları tablo 5.1'de görülmektedir.

Şekil 5.2'deki $\{p_1, p_2\}$ savağı sırasıyla t_1 ve t_3 geçişlerinin ateşlenmesi sonucu işaretsiz halde kalır ve bir daha hiçbir işaretleme altında jeton kazanamaz. Aynı zamanda teorem 5.5'e göre işaretli bir kapan barındırmadığı için ölü hale gelir.

Şekil 5.3'teki PA2, teorem 5.4'e göre tersinirdir ve tüm kapanlar ilk işaretlemede işaretli olduğundan hiçbir ölü geçiş yoktur. Aynı zamanda teorem 5.5'in koşullarını sağladığı için kilitlenmesizdir.

Tablo 5.1. PA1 ve PA2'ye ait savak ve kapanlar.

PA1		**PA2**	
Savaklar	Kapanlar	Savaklar	Kapanlar
$\varnothing$ $\{p_1, p_2\}$ $\{p_1, p_2, p_3, p_4\}$	$\varnothing$ $\{p_3, p_4\}$ $\{p_1, p_2, p_3, p_4\}$	$\varnothing$ $\{p_1, p_2\}$ $\{p_4, p_5\}$ $\{p_1, p_2, p_3\}$ $\{p_1, p_2, p_3, p_4\}$ $\{p_1, p_2, p_3, p_4, p_5\}$	$\varnothing$ $\{p_1, p_2\}$ $\{p_3, p_4, p_5\}$ $\{p_2, p_3, p_4, p_5\}$ $\{p_1, p_2, p_3, p_4, p_5\}$

5.1.1 Petri ağlarında baz savak ve kapan yapıları

Tanım 5.2: Eğer savak(kapan) diğer savak(kapan)'ların birleşimi olarak ifade edilemiyorsa **Baz Savak(Kapan)**'dır[9,11]. PA'daki her savak(kapan) uygun baz savak(kapan)'ların birleşiminden elde edilebilinir.

Diğer yandan iki savağın(kapanın) birleşimi yine savak(kapan)'dır.

5.1.2 Petri ağlarında minimal savak ve kapan yapıları

Tanım 5.3: Eğer savak(kapan) diğer herhangi bir savak(kapan) içermiyorsa **Minimal Savak(Kapan)**'dır[9,11].

Her minimal savak(kapan) bir baz savak(kapan)'dır. Fakat tüm baz savak(kapan)'lar minimal değildir.

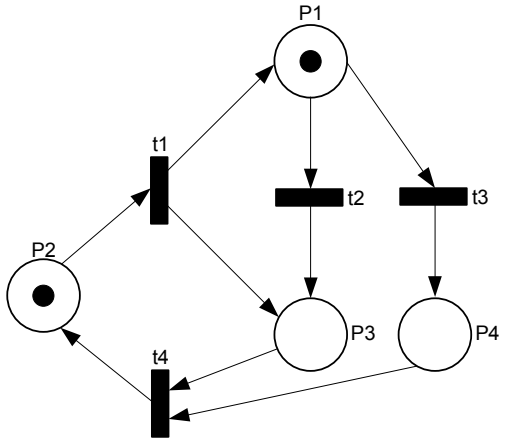
5.1.3 Petri ağlarında yer minimal savak ve kapan yapıları

Tanım 5.4: p_i yerini içeren savak(kapan)'ların p_i yerine göre minimallerinin oluşturduğu kümeye **Yer Minimal Savak(Kapan)** denir[11].

Bütün yer minimal savak(kapan)'lar aynı zamanda baz savak(kapan)'dır. Fakat, eğer savak(kapan) sadece ve sadece bazı yerlere göre yer minimal ise baz savak(kapan)'dır.

Örnek 5.2'de baz ile minimal savak(kapan) yapılarına ilişkin PA görülmektedir.

Örnek 5.2: Şekil 5.4'te görülen PA'da $S_1 = \{p_1, p_2, p_3\}$, $S_2 = \{p_1, p_2, p_4\}$, $S_3 = \{p_1, p_2, p_3, p_4\}$, $S_4 = \{p_2, p_3\}$ ve $S_5 = \{p_2, p_3, p_4\}$ olsun.



Şekil 5.4: Baz ve minimal savak(kapan) örneği[3].

$\bullet S_1 = \{t_1, t_2, t_4\} \subseteq S_1 \bullet = \{t_1, t_2, t_3, t_4\}$ olduğundan savak, $S_4 \bullet = \{t_1, t_4\} \subseteq \bullet S_4 = \{t_1, t_2, t_4\}$ ise kapandır. Aynı şekilde analiz edilerek S_2'nin savak, S_3'ün hem savak hem de kapan ve S_5'in kapan olduğu görülür.

Burada S_1 ve S_2 minimal ve baz savaklardır, S_3, S_4 ve S_5 ise baz kapanlardır. S_3 ve S_5 kapanları ise minimal değildir. Tablo 5.2'de bu yer kümelerine ait yapılar görülmektedir.

Tablo 5.2. Şekil 5.4'teki yerlere ait yapılar

Yerler Kümesi	Geçişler		Savak	Kapan	Min Savak	Min Kapan	Baz Savak	Baz Kapan
	Giriş	Çıkış						
$S_1 = (p_1, p_2, p_3)$	(t_1, t_2, t_4)	(t_1, t_2, t_3, t_4)	X		X		X	
$S_2 = (p_1, p_2, p_4)$	(t_1, t_3, t_4)	(t_1, t_2, t_3, t_4)	X		X		X	
$S_3 = (p_1, p_2, p_3, p_4)$	(t_1, t_2, t_3, t_4)	(t_1, t_2, t_3, t_4)	X	X				X
$S_4 = (p_2, p_3)$	(t_1, t_2, t_4)	(t_1, t_4)		X		X		
$S_5 = (p_2, p_3, p_4)$	(t_1, t_2, t_3, t_4)	(t_1, t_4)		X				X

5.2 Savak ve Kapanlara Dayanan Özellikler

Özellik 1: Serbest seçim ağları SSA (N, X_0),

Eğer N'deki her savağı işaretli kapan içeriyorsa canlıdır.

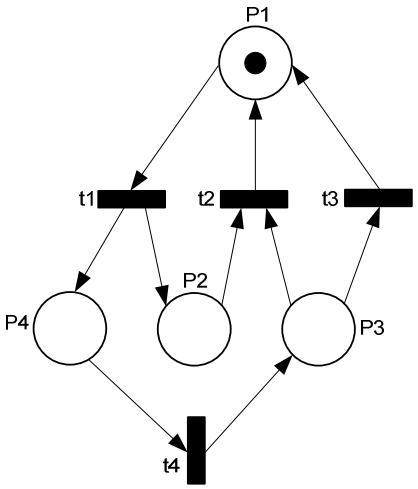
Örnek 5.3: Şekil 5.4'te görülen serbest seçim ağı (SSA) canlı değildir[3].

Çünkü, $S_2 = \{p_1, p_2, p_4\}$ savağı kapan içermez, bundan dolayı aynı zamanda işaretli kapanda içermemiş olur.

Özellik 2: Asimetrik seçim ağları ASA (N, X_0),

Eğer N'deki her savak işaretli kapan içeriyorsa canlıdır. Bu yeter koşuldur, aynı zamanda gerek koşul değildir.

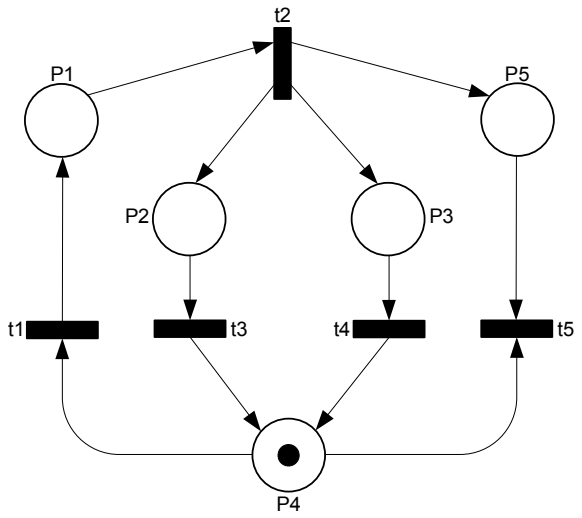
Örnek 5.4: Şekil 5.5'te görülem asimetrik seçim ağı ASA canlıdır[3].



Şekil 5.5: Canlı asimetrik seçim ağı (ASA)

Çünkü $\{p_1, p_3, p_4\}$ minimal savağı, işaretli $\{p_1, p_3, p_4\}$ kapanını, $\{p_1, p_2, p_3, p_4\}$ savağıda, işaretli $\{p_1, p_2\}$ kapanı ile işaretli $\{p_1, p_3, p_4\}$ kapanını içerir.

Örnek 5.5: Şekil 5.6'da $\{p_1, p_2, p_3, p_4\}$ savağının işaretli kapan içermemesine rağmen canlı olduğunu görüyoruz (özellik-2)[3].



Şekil 5.6: Canlı asimetrik seçim ağı (ASA)

Özellik 3: Kapan devreli ağda KD,

X_0 işaretlemesinden X_d işaretlemesine erişilebilmesi için:

Ancak ve ancak $X_d = X_0 + A^T x$ denklemini sağlayan pozitif tamsayı çözümlerinin olması ve (N_x, X_{0x}) 'te jetonsuz savaklarının bulunmaması gerekir[3].

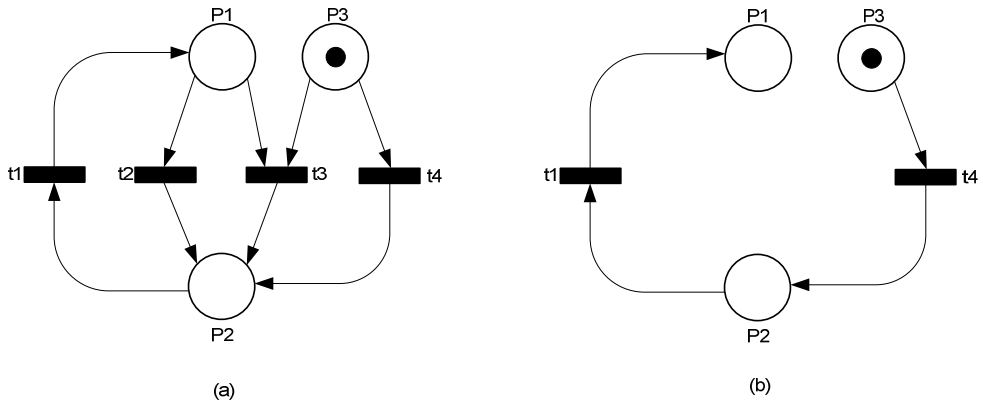
Burada;

N_x: t geçişleri $x(t) > 0$ ve bu geçişlerin giriş-çıkış yerleri ile bunların bağlantı dallarından oluşmuş N ağının alt ağını ifade eder

X_{0x}: N_x 'teki yerleri belirten X_0 vektörünün alt vektörüdür.

KD: Petri ağının yönlendirilmiş bölümlerindeki bütün yerler kapan ise kapan devreli ağ belirtir.

Örnek 5.6: Şekil 5.7 (a)'daki (ASA olmayan) ağı göz önüne alalım[3].



Şekil 5.7: Özellik 3'e ilişkilendirilen örnek (a) KDA (N, X_0) ve

(b) alt ağı (N_x, X_{0x})

Şekil 5.7 (a)'dan iki tane yönlendirilmiş devre olduğunu görüyoruz. $p_1 t_2 p_2 t_1 p_1$ ve $p_1 t_3 p_2 t_1 p_1$. Bu yönlendirilmiş devrelerdeki her bir yer kapandır $\{p_1, p_2\}$. Bu yüzden kapan devresidir KD.

$X_d = (1 \quad 0 \quad 0)^T$ olsun. $X_d = X_0 + A^T x$ denkeleminden $x = (1 \quad 0 \quad 0 \quad 1)^T$ çözümü kolaylıkla bulunabilir. Şekil 5.7 (b)'de alt ağ (N_x, X_{0x}) görülmekte ve $\{p_3\}$, $\{p_2, p_3\}$ ile $\{p_1, p_2, p_3\}$ oluşan bütün savakların jetona sahip olduğu görülür. Bu yüzden özellik 3'ün şartları sağlanmış olur. X_0 işaretlemesinden X_d işaretlemesine sırası ile t_4 ve t_1 geçişlerinin ateşlenmesi ile ulaşılabildiği kolaylıkla görülebilmektedir.

Özellik 4: Savak devreli ağda SD,

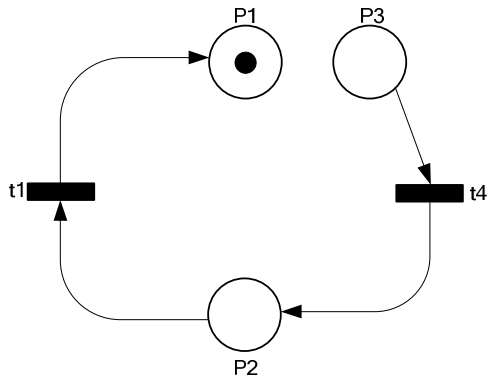
X_0 işaretlemesinden X_d işaretlemesine erişilebilmesi için:

Ancak ve ancak $X_d = X_0 + A^T x$ denklemini sağlayan pozitif tamsayı çözümlerinin olması ve (N_x, X_{dx})'te jetonsuz kapanlarının bulunmaması gerekir[3].

X_{dx}: N_x'teki yerleri belirten X_d vektörünün alt vektörüdür.

SD: Petri ağının yönlendirilmiş bölümlerindeki bütün yerler savak ise savak devreli ağ belirtir.

Örnek 5.6'da verilen PA'nın yönlendirilmiş yerlerinin giriş-çıkış ilişkileri incelendiğinde aynı zamanda kapan olduğu görülür. $x = (1 \quad 0 \quad 0 \quad 1)^T$ çözümüne ait $X_d = (1 \quad 0 \quad 0)^T$ vektörünün alt işaretlemesindeki PA'ı şekil 5.8'de görülmektedir.



Şekil 5.8: Özellik 4'e ilişkilendirilen örnek 5.6'ya ait PA'nın alt ağı (N_x, X_{dx})

Bu alt PA incelendiğinde $\{p_1\}$, $\{p_1, p_2\}$ ile $\{p_1, p_2, p_3\}$'ten oluşan bütün kapanların jetona sahip olduğu görülür. Yine, X_0 işaretlemesinden X_d işaretlemesine sırası ile t_4 ve t_1 geçişlerinin ateşlenmesi ile ulaşılabilinir.

Özellik 5: Kapan içeren devreli ağda KİD,

X_0 işaretlemesinden X_d işaretlemesine erişilebilmesi için:

Eğer $X_d = X_0 + A^T x$ denklemini sağlayan pozitif tamsayı çözümlerinin olması durumunda, (N_x, X_{0x})'teki her savak işaretli kapana sahip olması gerekir[3].

KİD: Petri ağının yönlendirilmiş bölümlerindeki yerler kapan içeriyorsa kapan içeren devreli ağ belirtir.

Özellik 6: Savak içeren devreli ağda SİD,

X_0 işaretlemesinden X_d işaretlemesine erişilebilmesi için:

Eğer $X_d = X_0 + A^T x$ denklemini sağlayan pozitif tamsayı çözümlerinin olması durumunda, (N_x, X_{dx}) jetonsuz kapan içermemesi gerekir.

Takip eden iki teorem yukarıda verilen özellikler ile minimal savak ve kapanlar arasındaki ilişkiyi göstermektedir.

Teorem 5.6: Eğer PA'da savak(kapan) varsa, bu savak(kapan)'da en az bir minimal savak(kapan) vardır[9].

İspat: Tanım gereği savak, baz savak içerir. Baz savak, minimal savak içermelidir. İçermiyorsa eğer, baz savak minimal savak olmalıdır, çünkü minimal savak diğer herhangi bir savak içermez. Bu yüzden her savak en az bir tane minimal savak içerir. Aynı tanımlamalar kapan içinde geçerlidir.

Teorem 5.7: PA'ı işaretsiz savağa sahipse eğer, işaretsiz minimal savağada sahiptir[9].

İspat: "$\Rightarrow$" savak, minimal savak içermelidir.

"$\Leftarrow$" terside geçerlidir.

Teorem 5.7: Her savak işaretli kapan içeriyorsa eğer, her minimal savakta işaretli minimal kapan içerir[9].

İspat: "$\Rightarrow$" Savak, minimal savak içermelidir. Eğer her savak işaretli kapan içeriyorsa, her minimal savakta işaretli kapan içermeli. Diğer yandan kapanda minimal kapan içermeli. Bu durumda her minimal savak, minimal kapan içermelidir.

"$\Leftarrow$" terside geçerlidir.

Sonuç olarak, eğer biz işaretsiz savakların olup olmadığını veya her savağın işaretli kapan içerip içermediğini kontrol etmek istersek, bu işlemleri minimal savak ve kapanlarda yapmamız yeterlidir. PA ağlarında savak ve kapan sayılarının yer ve geçişlere bağlı olarak çok sayıda olacağını göz önünde bulundurularak, minimal savak ve kapanlar üzerinde canlılık ve erişilebilirlik analizleri yapmak daha verimli olmaktadır.

6. YER MİNİMAL İLE MİNİMAL SAVAK VE KAPAN PROGRAMLARI VE ÖZELLİKLERİ

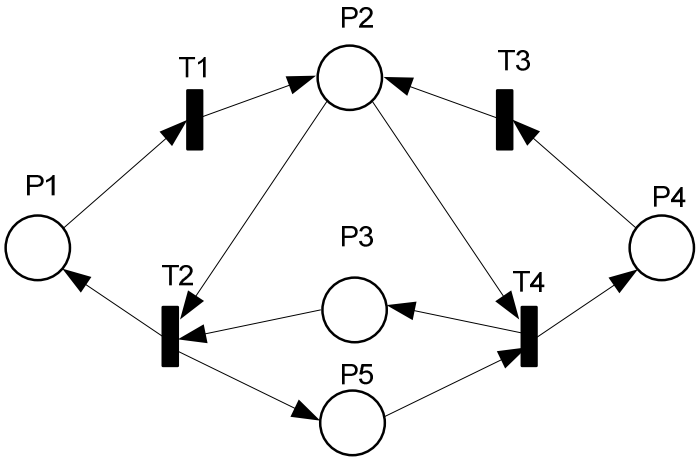
Bu bölümde, yer minimal ile minimal savak (kapan) yapılarını bulmaya yönelik farklı yöntemlere dayanarak geliştirilmiş programların özellikleri ile bu yapıların bulma adımları anlatılacak, değişik boyutlarda analiz edilmiş ağların bu yapıları bulma zamanları verilecektir.

6.1 Programların Çalışma Ortamı

Farklı yöntemlere dayalı bu programlar MATLAB (sürüm 7.2.0.232-R2006A) programlama dili kullanılarak geliştirilmiştir. Zaman analizleri elde ederken üzerinde koşturulan sistem özellikleri ise CPU: AMD Athlon XP 2500+ 1.8 GHz, 1 GB RAM ve Windows XP kurulu kişisel bilgisayardır.

6.1.1 Programların dosya giriş ve çıkış formatları

Her iki programda da aynı giriş ve çıkış yapısı kullanılmaktadır. Giriş dosyası olarak daha önce bölüm 2.6.2'de nasıl elde edildiği gösterilen durum-geçiş matrisi kullanılarak oluşturulan **.xls** uzantılı Excel dosyası, program dosyalarının kayıtlı olduğu MATLAB dizinine kopyalanır.



Şekil 6.1: Giriş dosyası için örnek PA

Şekil 6.1'de görülen PA'na ilişkin durum-geçiş matrisini yazacak olursak;

$$A = \begin{bmatrix} -1 & 1 & 0 & 0 & 0 \\ 1 & -1 & -1 & 0 & 1 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & -1 & 1 & 1 & -1 \end{bmatrix}$$

Elde edilen durum geçiş matrisinde satırlar geçişleri, sütunlar ise yerleri ifade eder. Durum-geçiş matrisindeki değerler boş bir Excel sayfasına tablo 6.1'deki gibi girilerek isim.xls giriş dosyası oluşturulmuş olur.

Tablo 6.1. Örnek PA'na ait giriş excel dosyası

-1	1	0	0	0
1	-1	-1	0	1
0	1	0	-1	0
0	-1	1	1	-1

Oluşturulan giriş dosyasına verilen .xls uzantılı isim MATLAB arayüzündeki program satırına girilerek program çalıştırılır ve de .txt uzantılı MinimalSiphons ile PlaceMinimalSiphons dosyaları MATLAB tarafından ekrana gösterilerek program dosyalarının kayıtlı olduğu dizine kopyalanır.

Her iki programın çalıştırılması ile şekil 6.1'e ait çıkış dosyaları Şekil 6.2 ve Şekil 6.3'te gösterilmiştir.

```
Minimal Siphons

{ 3  5 }

{ 1  2  4 }
```

Şekil 6.1: Minimal savak (siphon) çıkış dosyası

```
Place Minimal Siphons

{ 1  2  4 }

{ 1  3  5 }

{ 3  5 }

{ 1  2  3  4 }

{ 3  4  5 }

{ 1  2  4  5 }
```

Şekil 6.2: Yer minimal savak (siphon) çıkış dosyası

6.2 Durum-Geçiş Matrisine Dayalı Programlama Yöntemi

Bu yöntem, durum-geçiş matrisinden yararlınarak savakları bulmadan doğrudan baz savakları bulma algoritmasına dayanır. Şekil 6.3'te görülen diyagramda 2. adımda görülen A matrisinin yerlerine denk gelen p_j kolonlarındaki 1'lerin yok edilmesi mantığına dayanır.

Vjr 'ye atanan kolonlardaki 1'lerin yok edilmesi ile özyineli $V_{j(r+1)}$ vektörü şu şekilde elde edilir.

$V_{j(r+1)} \Rightarrow$ 2, Eğer $V[i,j]=1$ ve $A[i,k]=-1$;

1, Eğer $V[i,j]=1$ ve $A[i,k]\neq-1$;

2, Eğer $V[i,j]=-1$ ve $A[i,k]=1$;

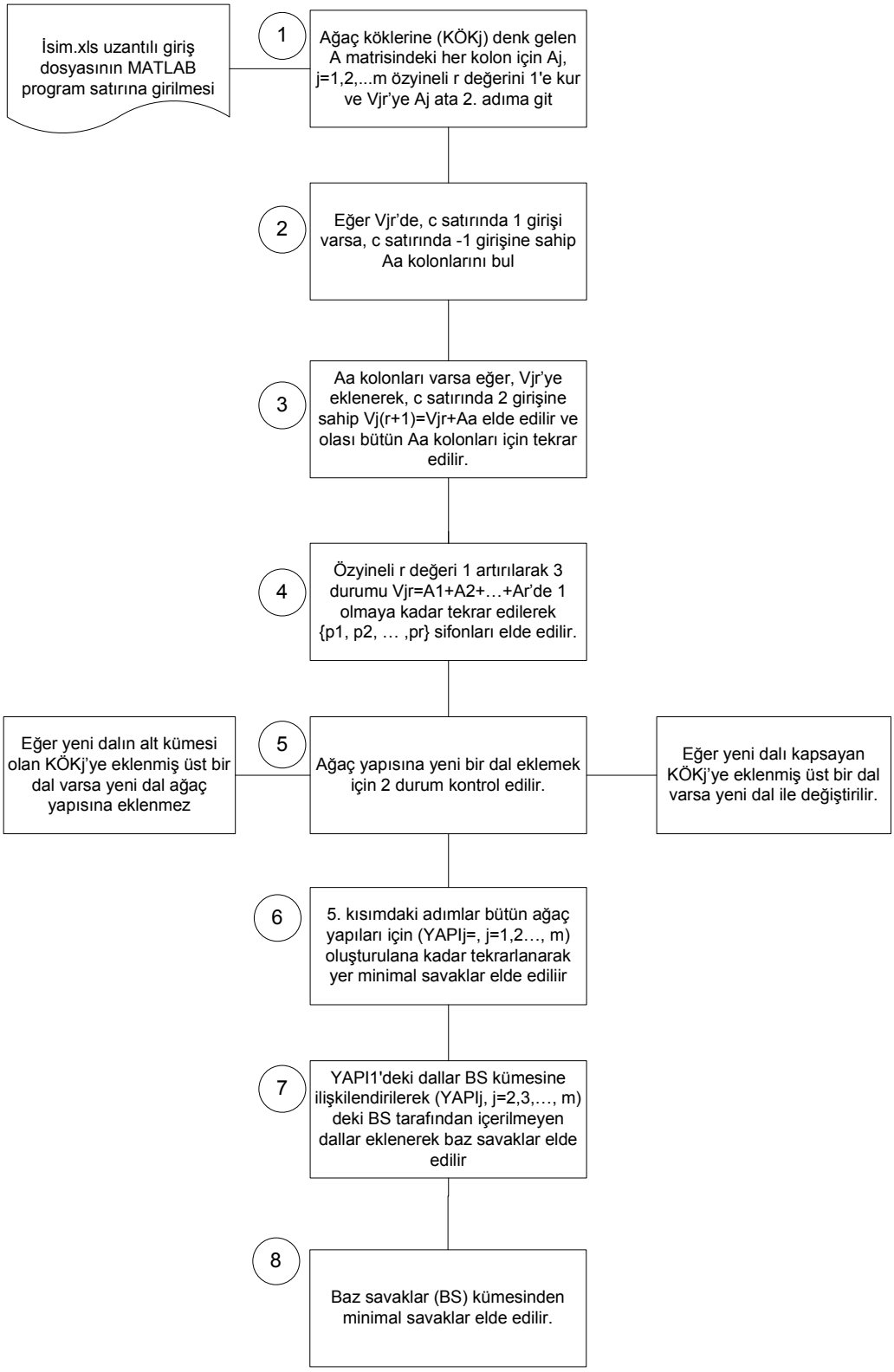
$A[i,k]$, Eğer $V[i,j]=0$ ve $A[i,k]\neq0$;

2, Eğer $V[i,j]=2$

i=1,…, T

j=V_j aday sütün

k $\in \{Aa \mid p_j, c=1$ *satırına karşılık gelen sütün vektörleri*$\}$



Şekil 6.3: Durum-geçiş matrisi algoritması

Şekil 6.1'de verilen örnek giriş PA için algoritmanın adımlarını adım adım uygularsak;

Adım 1: 1. sütün seçilerek *Vjr* 'ye atanır ve r özyineli değerine 1 atanır.

$$A_1 = \begin{bmatrix} -1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad ve \quad V_{11}^1 = \begin{bmatrix} -1 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \quad S_{11}^1 = \{p_1\}$$

Burada alt indislerin ilki oluşturulan ağaç yapılarının köklerine denk gelen yerlere(yer p_1) karşılık gelir. İkincisi ise özyineli değere karşılık gelir(burada 1). Üst indis ise aynı alt indislere sahip farklı V vektörlerini ifade eder.

Adım 2-3:

$c = 2 \Rightarrow sütün \quad 2,3$

$$V_{12}^1 = \begin{bmatrix} -1 \\ 1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 1 \\ -1 \end{bmatrix}, S_{12}^1 = \{p_1, p_2\} \quad \text{ve} \quad V_{12}^2 = \begin{bmatrix} -1 \\ 1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ -1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 2 \\ 0 \\ 1 \end{bmatrix}, S_{12}^2 = \{p_1, p_3\}$$

V_{11}^1'in 2. satırında 1 olduğu için bu satıra karşılık gelen sütünlarda -1 bulunan yerler $p_2 \quad ve \quad p_3$ tür.

Adım 4: V_{12}^1 ve V_{11}^2 vektörlerinde 1 bulunduğu için adım 2 ve 3 tekrar edilir.

$c = 3 \Rightarrow sütün \quad 4$

V_{12}^1 için: $$V_{13}^1 = V_{12}^1 + A_4 = \begin{bmatrix} 2 \\ 2 \\ 1 \\ -1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 2 \\ 1 \end{bmatrix}, S_{13}^1 = \{p_1, p_2, p_4\}$$

$c = 4 \Rightarrow sütün \quad 2,5$

V_{12}^2 için: $$V_{13}^2 = V_{12}^2 + A_2 = \begin{bmatrix} -1 \\ 2 \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 1 \\ 2 \end{bmatrix}, S_{13}^2 = \{p_1, p_3, p_2\}$$

ve

$$V_{13}^3 = V_{12}^2 + A_5 = \begin{bmatrix} -1 \\ 2 \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \\ -1 \end{bmatrix} = \begin{bmatrix} -1 \\ 2 \\ 0 \\ 2 \end{bmatrix}, S_{13}^3 = \{p_1, p_3, p_5\}$$

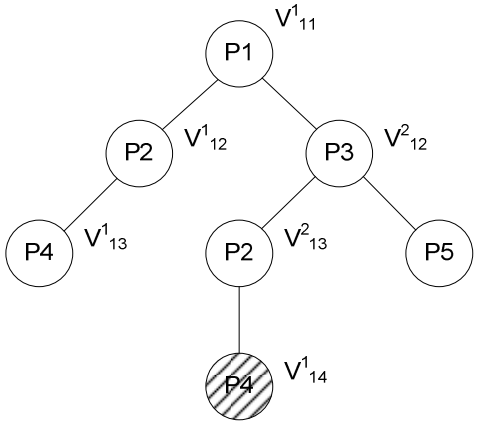
3. iterasyon adımında V_{13}^1 ile V_{13}^3 vektörlerine baktığımızda 1'den arındırıldıklarını görürüz. Bu vektörler birbirlerinin alt veya üst kümesi olmadıkları için savak oluştururlar. 5. adım ile p_1'e göre oluşturulan ağaç yapısına dal olarak ilave edilebilirler. Bu savaklar, aynı zamanda p_1'e göre yer minimal savaklardır. Burada dikkat edilecek nokta yer minimal savaklar p_1'e göre olan özyineli yapı bitene kadar elde edilemiyeceğidir.

V_{13}^2 vektöründe 1 olduğu için adım 2 ve 3 tekraralanır.

$c = 3 \Rightarrow sütün \;\; 4$

V_{13}^2 için : $V_{14}^1 = V_{13}^2 + A_4 = \begin{bmatrix} 2 \\ 2 \\ 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 2 \\ 2 \end{bmatrix}, S_{14}^1 = \{p_1, p_3, p_2, p_4\}$

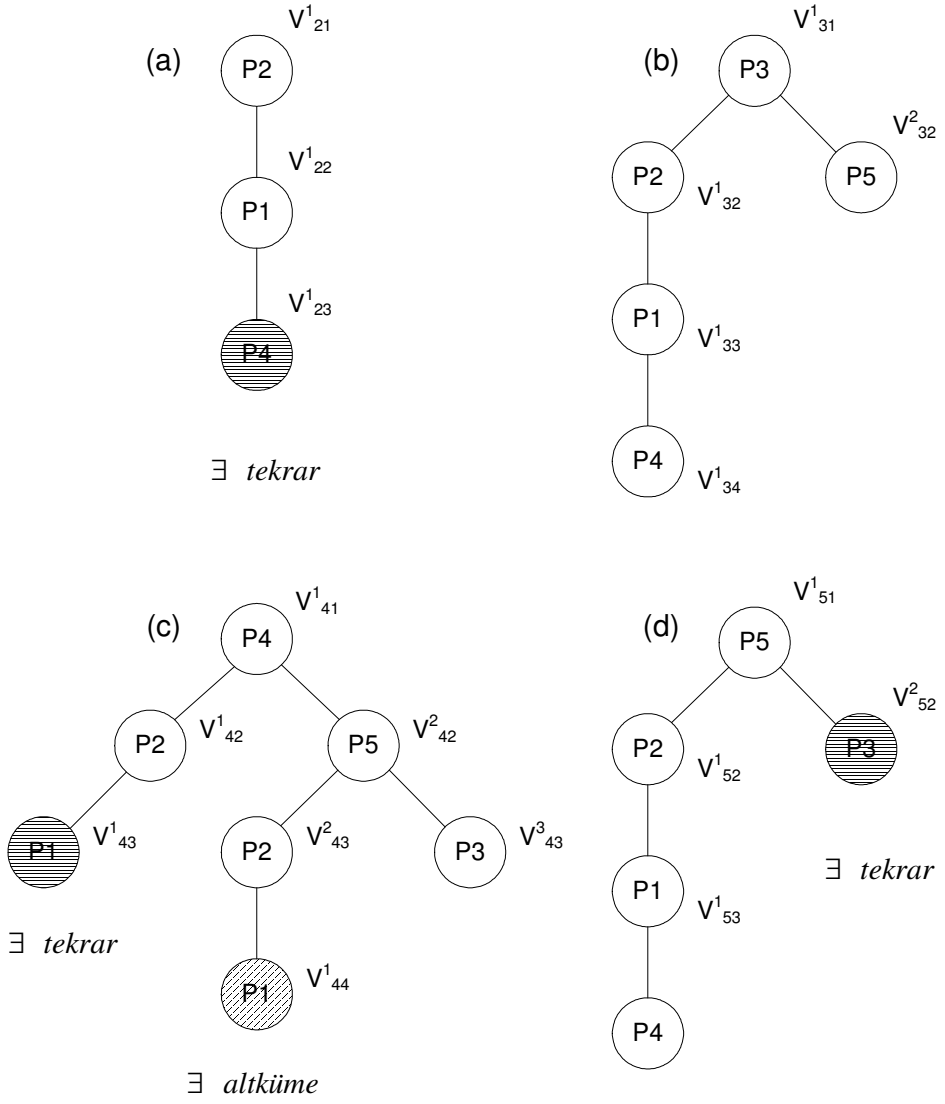
Adım 5: V_{14}^1 vektörü 1 içermediği için 5. adım uygulanır. 5. adımda V_{13}^1'ün $\{p_1, p_2, p_4\}$, V_{14}^1'ün alt kümesi olduğunu görüyoruz. Bu yüzden p_1'e göre yer minimal savak değildir ve yapı1'e eklenmez.



$\exists$ *altküme*

Şekil 6.4: p_1 yerine göre üretilen yer minimal savaklar

Şekil 6.4'te p_1're göre üretilen ağaç yapısı görülmektedir. Diğer yerlere göre olan yer minimal savaklarda aynı yöntemle elde edilir. Şekil 6.5'te ise diğer dört yere göre üretilen ağaç yapıları ve yer minimal savaklar görülmektedir.



Şekil 6.5: p_2, p_3, p_4 *ve* p_5 yerlerine göre üretilen yer minimal savaklar

Adım 6: Şekil 6.5 (a)'da $\{p_2, p_1, p_4\}$, (c)'de $\{p_4, p_2, p_1\}$ ve de (d)'de $\{p_5, p_3\}$'ün yerlere göre üretilen minimal savaklarda tekrarladığından dolayı elde edilen baz savak kümesine eklenmediğini görüyoruz. Bu nedenle son olarak elde edilen baz savak kümesi;

$$BS = \left[\{p_1, p_2, p_4\}, \{p_1, p_3, p_5\}, \{p_3, p_2, p_1, p_4\}, \{p_3, p_5\}, \{p_4, p_5, p_3\}, \{p_5, p_2, p_1, p_4\}\right]$$

elemanlarından oluşur.

Adım 7: Adım 6'da elde edilen baz savak kümesinde diğer elemanlar tarafından içerilmeyen kümeler analiz edilerek minimal savak kümesi elde edilir. Buradan hareketle

$Minimal \quad savaklar = \left[\{p_1, p_2, p_4\}, \{p_3, p_5\}\right]$ olarak elde edilir.

6.3 Lojik Tabanlı İkili Tamsayı Lineer Programlama Yöntemi

Bu yöntem PA'nın lojik ile ifade edilmesi ile lineer programlamada kısıtlamaların oluşturulması ve de minimal savaklara denk gelen objektif fonksiyonunun formülize edilmesine dayanır. Eğer çözümün bulunması durumunda ilk olarak formülize edilmiş kısıtlamalara yenilerinin eklenmesi ile bütün minimal savaklar iteratif bir şekilde elde edilir. Bu kısıtlamaların eklenmesi ile bir önceki yönteme göre tarama uzayı büyük ölçüde azaltılmış iteratif bir yöntem elde edilmiş olur. Yer minimal savakların bulunması içinde, sırası ile yerlere ait $p_i = 1$ kısıtlamaları eklenerek yer minimal savaklar üretilir.

İkili Tamsayı Lineer Programlamanın tanım bağıntılarını verecek olursak[7]:

$$\begin{aligned} &x \in \{0,1\}, \quad x \geq 0 \\ &A \cdot x \leq b: \quad A \in R^{nxm}, b \in R^m \\ &\min c^T \cdot x: \quad c \in R^m \end{aligned} \qquad \textbf{(6.1)}$$

$A \cdot x \leq b$: kısıtlamaları

$\min c^T \cdot x$: objektif fonksiyonunu ifade eder.

6.3.1 Savakların lojik ifadelerle tanımlanması

Savaklar lojik terimlerinin eşitlikleri olarak ifade edilebilir[3]. Buradaki mantığı şu şekilde açıklayabiliriz;

S'nin bir savak kümesi olduğunu farz edelim, p_i yerinin S'e ait olabilmesi için savak tanımı gereği $\bullet p_i$'ye gelen bütün geçişlerin $S\bullet$'in çıkış kümesinde olması gerekmektedir. Yani, eğer $p_i \in S$ ise diğer hangi yerlerin S'de olması gerektiğini tanımlamamız gerekir. Bu yerler I ve O giriş ile çıkış matrislerinin uygun denetlenmesi ile üretilebilir.

Şekil 6,1'deki örnek PA'da şunu söyleyebiliriz:

$$Eğer \quad p_1 \in S, \quad ise \quad (p_2 \in S) \quad veya \quad (p_3 \in S), \tag{6.2}$$

Çünkü $\bullet(p_1) = \{t_2\}$ ve $t_2 \in (p_2)\bullet$, $t_2 \in (p_3)\bullet$.

Yukarıda belirtmiş olduğumuz durumu lojik ifade ile yazacak olursak;

x_i: p_i yerine karşılık düşen Boolean değişkenini ifade etsin, bu durumda (6.2) eşitliği

$$x_1 \Rightarrow x_2 \vee x_3 \tag{6.3}$$

şeklinde yazılabilir.

p_2 yeri için lojik ifade değişik formda olur:

$$Eğer \quad p_2 \in S, \quad ise \quad (p_1 \in S) \quad ve \quad (p_4 \in S),$$

Çünkü $\bullet(p_2) = \{t_1, t_3\}$ ve $t_1 \in (p_1)\bullet$, $t_3 \in (p_4)\bullet$

Şekil 6.3'teki örnek PA için bütün koşulları yazarsak:

$$\begin{aligned}
&Eğer \quad p_1 \in S, \quad ise \quad (p_2 \in S) \quad veya \quad (p_3 \in S),\\
&Eğer \quad p_2 \in S, \quad ise \quad (p_1 \in S) \quad ve \quad (p_4 \in S),\\
&Eğer \quad p_3 \in S, \quad ise \quad (p_2 \in S) \quad veya \quad (p_5 \in S),\\
&Eğer \quad p_4 \in S, \quad ise \quad (p_2 \in S) \quad veya \quad (p_5 \in S),\\
&Eğer \quad p_5 \in S, \quad ise \quad (p_2 \in S) \quad veya \quad (p_3 \in S),
\end{aligned}$$

Olarak elde edilir. Bu koşullara denk düşen kural tabanlı model ise:

$$\begin{aligned}
&x_1 \Rightarrow x_2 \vee x_3,\\
&x_2 \Rightarrow x_1 \wedge x_4,\\
&x_3 \Rightarrow x_2 \vee x_5,\\
&x_4 \Rightarrow x_2 \vee x_5,\\
&x_5 \Rightarrow x_2 \vee x_3,
\end{aligned} \tag{6.4}$$

Şeklinde ifade edilir. Burada dikkat edilmesi gereken $\forall i, \quad x_i = yanlış$ çözümünün çıkarılması gerekir, çünkü bir savak yapısını belirtmez.

6.3.2 Savak yapısının ikili tamsayı lineer programlama ile modellenmesi

Eğer x Boolean değişkeni yerine ikili değişken $x \in \{0,1\}$ ise (6.4)'te elde edilen kurallar aşağıdaki eşitsizliklere dönüşür:

$$\begin{aligned} &x_1 \le x_2 + x_3, \\ &x_2 \le x_1 * x_4, \\ &x_3 \le x_2 + x_5, \\ &x_4 \le x_2 + x_5, \\ &x_5 \le x_2 + x_3, \end{aligned} \qquad \textbf{(6.5)}$$

Lineer olmayan kısıtlama $x_2 \le x_1 * x_4,$ lineer olarak aşağıdaki kısıtlamalara eşit olur.

$$\begin{aligned} &x_2 \le x_1, \\ &x_2 \le x_4, \end{aligned}$$

İkili lineer programlama için kısıtlamalarımızı özetleyecek olursak;

$$\begin{aligned} &x_1 \le x_2 + x_3, \\ &x_2 \le x_1, \\ &x_2 \le x_4, \\ &x_3 \le x_2 + x_5, \\ &x_4 \le x_2 + x_5, \\ &x_5 \le x_2 + x_3, \end{aligned} \qquad \textbf{(6.6) (a)}$$

Sistemin sıfır çözümü savak belirtmeyeceği için ilave olarak,

$$\sum_{i=1}^{5} x_i \ge 1 \qquad \textbf{(6.6)(b)}$$

Kısıtlamasını eklemeli ve de x_i değişkenlerini ikili olarak tanımlamalıyız.

$$x_i \in \{0,1\}, \quad i = 1,...,5 \qquad \textbf{(6.6)(c)}$$

(6.5a)'daki kısıtlamalar, O(çıkış) ve I(giriş) matrislerininin tanımlarından hareketle analiz edilmeleri ile elde edilinebilinir.

Temiz PA'da (öz çevrim yok) durum-geçiş matrisi :

A=O-I giriş ve çıkış matrislerinin farkı olarak yazılabilir. Şekil 6.1'deki örnek PA'nın giriş ve çıkış matrislerini yazacak olursak;

$$O = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix} \text{ ve de } I = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix} \text{ olarak elde edilir.}$$

O matrisini sıfırdan farklı olan yerlerine karşılık düşen yerleri indis değerleri ile gösterelim bu durumda:

$O_{indis} = \{(1,2),(2,1),(3,2),(4,3),(4,4),(2,5)\}$ olarak ifade edilir,

O_{indis} matrisinin (i, j) elemanlarına karşılık kısıtlamaları aşağıdaki ilişki ile elde edebiliriz.

$$x_j \leq I_{\bullet i} * X$$

Burada,

$I_{\bullet i}$: I matrisinin i satırlarını ifade eder.

X : $X = [x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5]^t$ vektörünü belirtir.

Sistem (6.6)'in çözümleri PA'na ait bütün savakları verir. Bizim burada ilgilendiğimiz analiz zamanını azaltan, savaklara göre çok daha az sayıda olan minimal savaklar ile baz savakları bulmaktır. (6.6)' in çözümündeki minimum boyuta sahip olan savaklar aynı zamanda minimal savaklardır. Bu çözümü sağlamak için lineer programlamada(6.1):

$\min c^T \cdot x : \quad c \in R^m$ objektif fonksiyonuna denk gelen şu bağıntıyı tanımlamamamız gerekmektedir:

$$\min \quad = \sum_{i=1}^{n} x_i \qquad \textbf{(6.7)}$$

İkili tamsayı lineer programlamanın objektif fonksiyonu (6.7) ile kısıtlamaları (6.6)'nın çözümünün olması durumunda minimal savağı verir. Bütün minimal savaklar kümesini bulmak için bulunan her minimal savak $\min S$ için iteratif olarak yeni kısıtlamalar eklememiz gerekmektedir. Böylelikle $\min S$ 'i ve $\min S$ 'i içeren daha geniş savakları çözüm kümesinden çıkarmış oluruz. Kısıtlama şu şekilde formüle edilebilinir:

$$\sum_{x_i \in \min S} x_i \leq |\min S| - 1, \qquad \textbf{(6.8)}$$

$|\min S|$: $\min S$ 'teki eleman sayısını belirtir.

(6.8) kısıtlaması ile orijinal problemde $\min S$ tarafında içerilen yerler çözüm kümesinden çıkarılmış olur. Bu kısıtlamalar iteratif olarak her bulunan yeni minimal savak için lineer programlamanın kısıtlamalarına eklenir.

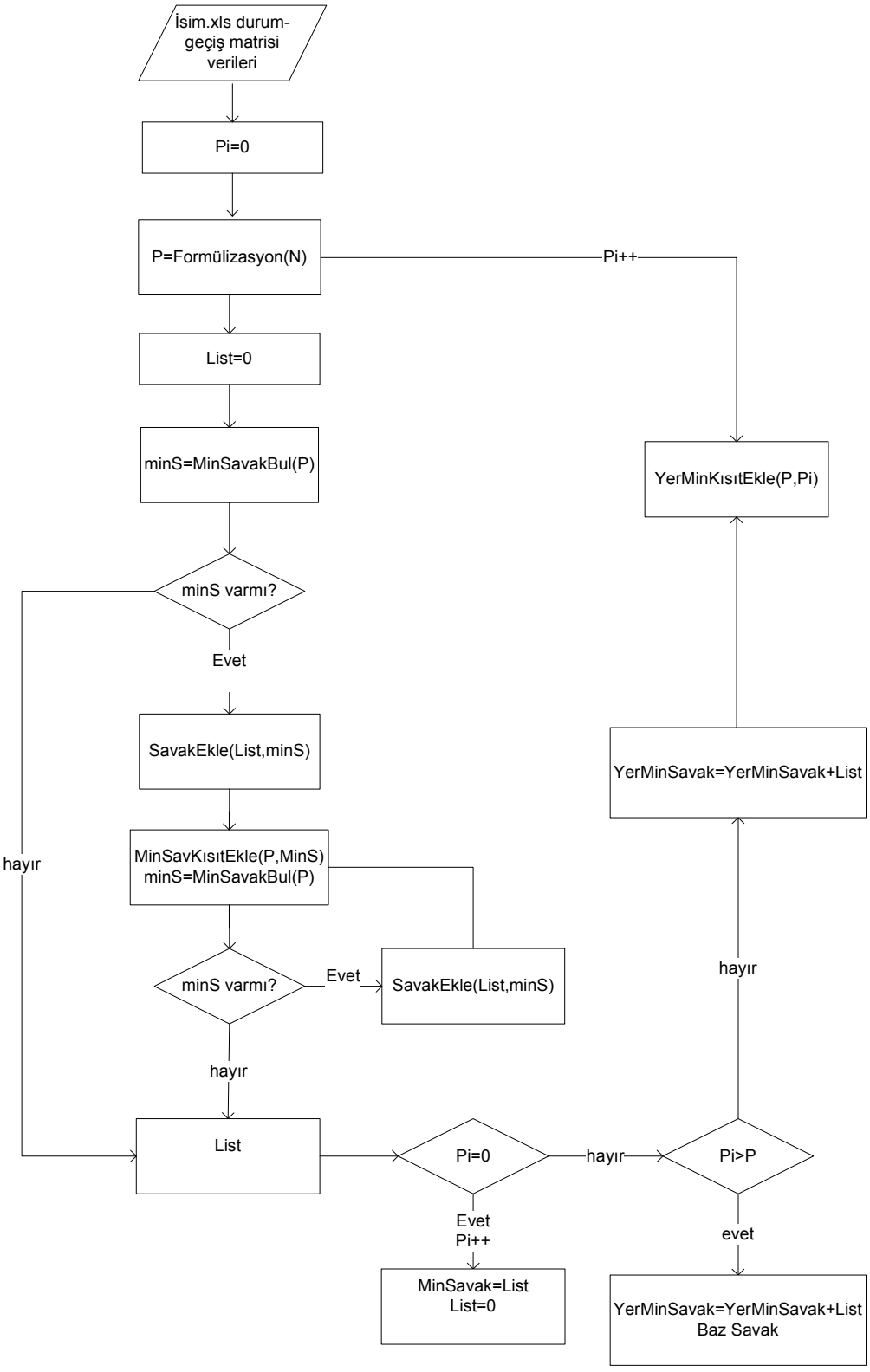
(6.6), (6.7) ve (6.8) ile oluşturulan ikili tamsayı lineer programlama problemi aynı yapı kullanılarak iteratif bir şekilde hesaplanır. Her iterasyonda çözümün olması durumunda minimaldir ve bir önceki bulunan savağı içermez. Bu şekilde çözümün olmadığı noktaya kadar iterasyon devam ettirilerek bütün minimal savak kümesi oluşturulur.

Yer minimal savakları bulmak için orjinal probleme (6.6) ek kısıtlama eklememiz gerekmektedir. p_i yerine göre bulunacak minimal savakları bulmak için

$$x_i = 1, \quad i = 1, ..., m \qquad \textbf{(6.9)}$$

Kısıtlaması eklenerek yukarıda anlatılan adımlar sistemin çözümsüz olana kadar devam ettirilir. $i = 1, ..., 5$ kadar olan bütün yerlere göre bulunan yer minimal savaklardan eşit olanlardan sadece biri alınarak baz savak kümesi oluşturulur.

Şekil 6,6'da lojiğe dayalı ikili tamsayı lineer programlama algoritması görülmektedir.



Şekil 6.6: Lojiğe dayalı ikili tamsayı lineer programlama algoritması

Şekil 6.6'da Formülizasyon fonksiyonu verilen PA'na uyarlanan ikili tamsayı lineer programlamaya ait P problemini oluşturur, MinSavakBul fonksiyonu problem P'den minimal boyutta MinS savağını bulur, MinSavKısıtEkle ile YerMinKısıtEkle minimal ve yer minimal savaklar bulunurken gereken kısıtlamaları ekler ve SavakEkle fonksiyonuda bulunan savakları Liste ekler.

6.3.3 Lojik tabanlı yöntemin VHDL benzetimi

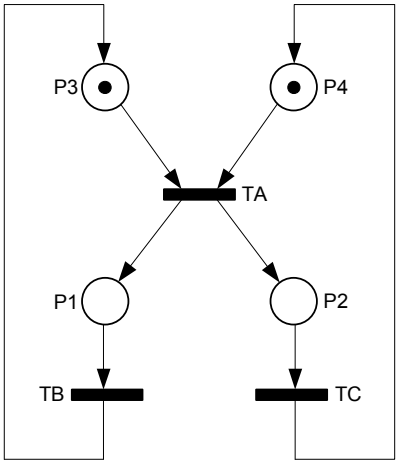
VHDL donanım tasarlama dilinde komutlar, diğer programlama dillerinde olduğu gibi sırası ile değilde eşzamanlı işletildiği için, PA'larının davranışsal özelliği olan eşzamanlı olayların tasarlanmasında uyumluluk göstermektedir. VHDL ile PA'nın değişik tasarımlarını yapmak mümkündür[13]. Örneğin;

- Sadece geçişlerin açıkça tanımlı olduğu özet model, burada yer bilgileri geçişlerle olan dal ilişkilerinden çıkarılmaktadır.
- Geçiş ile yerlerin her ikisininde tanımlı olduğu davranışsal model, PA'nın davranışsal özellikleri tanımlamada büyük esneklik sağlar.
- Geçiş ve yerlerin sayısal devre olarak düşünülüp tasarlandığı donanım modeli.

VHDL componentlerin tanımlanmasına ve aralarındaki bağlantılara izin verir. VHDL'i diğer programlama dillerinden farklı kılan, yeni veri türlerinin tanımlanmasına olanak sağlaması ve bunlar üzerinde işlem yapılmasına izin vermesidir. Bu yüzden modellenen sisteme ait veri paketlerinin, objelerin ve de fonksiyonların tanımlanmasında oldukça kullanışlıdır. PA'da olayların oluşması, ayrık zamanlı modelleme olan VHDL'in davranışsal özelliği ile örtüşmektedir[13].

VHDL'de her bir komponent tanımlamalarının bulunduğu iki bölümde teşkil edilir. '*entity*' komponentlerin dış bağlantıları ile temel parametrelerini tanımlar. '*Architecture*' ise içyapıyı belirtir veya komponentin davranışsal yapısını tanımlayan durumlardan oluşur.

Örnek 6.1: Şekil 6.7'de görülen PA'na ait geçişleri VHDL 'process' yapısı ile yazılmış örnek program verilmiştir[13]. Process yapısının özelliğinden dolayı durumlar sırası ile değerlendirilmektedir. 'Process'ler kendileri eşzamanlı VHDL durumları oldukları için PA'nın elemanlarını tanımlanmasında kullanılabilirler.



Şekil 6.7: VHDL ile programlanan PA örneği

```
use work.std_logic_1164.all;
entity net is
end net;

architecture abstract of net is

component place
generic (initial_token : integer :=0);
port(s,r : in std_logic; q : out std_logic);
end component;

signal p1q , p2q , p3q , p4q : std_logic ; --place outputs
signal sp1 , sp2 , sp3 , sp4 : std_logic ; --set signals
signal rp1 , rp2 , rp3 , rp4 : std_logic ; --reset signals

begin

P1: place port map (sp1 , rp1 , p1q); --instances of places
P2: place port map (sp2 , rp2 , p2q);
 --P3 and P4 contain initial tokens
P3: place generic map (1) port map (sp3 , rp3 , p3q);
P4: place generic map (1) port map (sp4 , rp4 , p4q);
```

```
TA: process            --transitions represented by processes
    begin
      -- wait until input places active
      wait until p3q='1' and p4q='1' ;
      assert false report "TA enabled" severity node;
      wait for 3 ns;       --'wait ' represents transition time
      -- reset input places, set output places
      rp3 <='1' , '0' after 1 ns; rp4 <='1' , '0' after 1 ns;
      sp1 <='1' , '0' after 1 ns; sp2 <='1' , '0' after 1 ns;
      assert false report "TA terminated" severity node;
   end process

TB:  process
    begin
      wait until p1q='1';
      assert false report "TB enabled" severity node;
      wait for 5 ns;
      assert false report "TB terminated" severity node;
      rp1 <='1' , '0' after 1 ns;
      sp3 <='1' , '0' after 1 ns;
   end process;
TC:  process
    begin
      wait until p2q='1';
      assert false report "TC enabled" severity node;
      wait for 8 ns;
      rp2 <='1' , '0' after 1 ns;
      sp4 <='1' , '0' after 1 ns;
      assert false report "TC terminated" severity node;
   end process;

end abstract;
```

Programda dikkat edilirse signal **sp1**, **rp1** gibi tanımlanan sinyaller yerlere ve bu yerlerdeki jetonlara denk gelmektedir. Set ve reset durumları iki farklı durumlu yerleri temsil etmektedir.

Şekil 6,3'teki örnek PA için yazılan koşullar lojik ifadelerden oluşmaktadır. VHDL ile bu lojik ifadelerin tanımlanması çok kolaydır. 6.4 ile elde edilen lojik tabanlı kuralları yeniden bakacak olursak;

$$x_1 \Rightarrow x_2 \vee x_3,$$
$$x_2 \Rightarrow x_1 \wedge x_4,$$
$$x_3 \Rightarrow x_2 \vee x_5,$$
$$x_4 \Rightarrow x_2 \vee x_5,$$
$$x_5 \Rightarrow x_2 \vee x_3,$$

Bu kuralların VHDL ifadeleri şu şekilde verilebilinir:

entity PetriAğı **is**

end PetriAğı

architecture kısıtlamalar **of** PetriAğı **is**

begin

X_1 <= X_2 or X_3;

X_2 <= X_1 and X_4;

X_3 <= X_2 or X_5;

X_4 <= X_2 or X_5;

X_5<= X_2 or X_3;

and architecture kısıtlamalar;

yukarıda görülen boolean ifadelerin çözümünde 'true' olanlardan gerekli döngüler tanımlanarak elde edilecek minimum boyuttaki yapı minimal savağa verecektir. İteratif bir yapı tanımlanarak bu elde edilen minimal yerler, çözümden çıkarılarak aynı şekilde yine minumum yapıda çözümler geriye kalan minimal savakları verecektir.

Yer minimal savakların bulunmasında ise architecture kısmına;

$X_i = \text{true},\ i = 1,..., p$ kısıtlamalarının eklenmsei gerekecektir.

$X_1 = 1;$ yerine ait olan basit VHDL kodunu şu şekilde verebiliriz:

entity PetriAğı **is**

end PetriAğı

architecture kısıtlamalar **of** PetriAğı **is**

begin

$\mathbf{X}_1$:boolean:=true;

$\mathbf{X}_1$ **<=** $\mathbf{X}_2$ **or** $\mathbf{X}_3$;

$\mathbf{X}_2$ **<=** $\mathbf{X}_1$ **and** $\mathbf{X}_4$;

$\mathbf{X}_3$ **<=** $\mathbf{X}_2$ **or** $\mathbf{X}_5$;

$\mathbf{X}_4$ **<=** $\mathbf{X}_2$ **or** $\mathbf{X}_5$;

$\mathbf{X}_5$**<=** $\mathbf{X}_2$ **or** $\mathbf{X}_3$;

and architecture kısıtlamalar;

verilen bu denklem sistemlerinin yukarıda belirtildiği gibi çözüm olmayana kadar iteratif bir şekilde tekrarlanması ile $X_1 = P1$ yerine göre minimal savaklar elde edilmiş olur.

$X_i,\ i = 2,..., p$ diğer yerler için yukarıdaki belirtilen yapılan döngüler kurularak bütün P yerleri için yer minimal savaklar bulanabilir. Bulunan bu yer minimal savaklardan tekrar edilenler çözümden çıkarılarak baz savak kümesi oluşturulur.

Minimal savak ve yer minimal savak kümeleri anlatılan yöntemler aynı zamanda kapan kümeleri üzerinde yapılarak minimal kapan ve yer minimal kapan kümeleri elde edilebilinir. Savak tanımında $\bullet S \subseteq S \bullet$ olarak kullanılan yapı programların tanım kısımlarında $Q\bullet \subseteq \bullet Q$ kapan tanımına dönüştürülerek istenen kümeler elde edilir.

6.4 Programların Analiz Sonuçları

Farklı yapılara sahip değişik boyutlardaki PA'nın minimal ve baz savak yapılarının sayısı ile bunları üretme süreleri tablo 6.2'de görülmektedir. Tablonun ilk sütununda analizleri yapılan PA'larının geçiş ve yerlerinin sayısından oluşan [t,p], durum-geçiş matrisinin boyutları verilmiştir. İkinci sütünda bu PA'larından elde edilen minimal savakların sayısı, üçüncü sütunda da baz savakların sayısı görülmektedir.

Dördüncü ve beşinci sütunlarda sırası ile lojik ile durum-geçiş matrisine dayalı yöntemlerin minimal ve baz savakların analiz zamanları görülmektedir. Verilen hesaplama süreleri saniye cinsindendir.

Tablo 6.2. Programların analiz sonuçları

Petri Ağı Boyutu	Minimal Savak	Baz Savak	Lojik Yöntemi zamanı(sn)	Durum-geçiş yöntemi zamanı(sn)
01-[t , p] = [4 , 5]	2	4	4.931682	3.242596
07-[t , p] = [4 , 4]	2	2	4.827006	4.204768
03-[t , p] = [5 , 5]	3	5	5.400813	4.348383
02-[t , p] = [4 , 5]	2	6	5.741535	4.470450
08-[t , p] = [9 , 10]	1	3	5.708900	5.543644
04-[t , p] = [7 , 7]	2	6	6.359729	5.520327
05-[t , p] = [10 , 9]	4	16	10.834193	-
06-[t , p] = [20 , 18]	34	198	263.004087	-

Tablo 6.2'den görüldüğü gibi analiz zamanı açısından durum-geçiş matrisine dayalı yöntemin lojik yöntemine göre sonuca daha kısa sürede ulaşmaktadır. Fakat ağın boyutunun büyümesi ile minimal ile baz savak yapılarını bulamamaktadır. Lojiğe dayalı yöntem ise büyük boyutlu ağlarda da çözüme ulaşmakta, fakat ağın çok büyük olması durumunda sonuç vermemektedir. Dikkat edilirse ağın boyutuna paralel olarak analiz zamanında da büyük oranda artış olduğunu görülmektedir.

Tablo 6.2'de kullanılan PA'nın durum-geçiş matrislerinden oluşan giriş dosyaları ile programların ürettiği minimal ve baz savak çıkış dosyaları EK-A'da verilmiştir.

7. SONUÇLAR VE ÖNERİLER

Bu çalışmada, paralel, eşzamanlı ve ayrık olaylı sistemler gibi problemlerin modellenmesinde kullanılan Petri ağlarının yapısal özelliklerinden olan minimal ile baz savak ve kapan yapısı incelenmiş ve bu yapıları bulmaya yönelik iki farklı yöntem gerçekleştirilerek MATLAB programlama dilinde kodlanmıştır. Petri ağlarının tanımlamaları, özellikleri ve analiz yöntemleri incelenmiştir.

PA'da canlılık, ulaşılabilirlik gibi analizlerinde kullanılan minimal ve baz savak(kapan) yapılarını bulan bu iki yöntemin birbirlerine göre olan avantaj ve dezavantajları incelenmiş, bu yapıları bulma süreleri verilmiştir. Lojik tabanlı yöntemin VHDL ile programlanabilirliği incelenmiş, PA'nın basit VHDL örnek kodları verilmiştir. Diğer yandan bu yapılar kullanılar diğer tüm savak(kapan) yapıları türetilebilir.

Çok büyük yapılı PA'larında sonuça ulaşamayan bu yöntemlerin eksiklerini giderecek yöntemlerin geliştirilmesi ile analiz süresini kısaltacak yöntemlerin geliştirilmesi ileriki çalışmaların amaçlarındandır.

Bir diğer çalışmada literatürde mevcut olan diğer savak üretme algoritmalarının bu çalışmadakiler ile birlikte hepsinin performans ve analiz sürelerini karşılaştırmak olabilir.

Bu çalışmada MATLAB ile kodlanan bu iki farklı yöntemin VHDL ile kodlanarak büyük PA'rındaki performansı ve analiz sürelerini karşılaştırmak diğer bir çalışma olabilir.

KAYNAKLAR

[1] **Cassandras, G.C. and Lafortune, S.,** 1999. Introduction to Discrete Event Systems, Kluver Academic Publishers, MA.

[2] **Peterson , James L.,** 1981. Petri Net Theory and The Modelling of Systemts, Prentice-Hall, Inc., Englewood Cliffs

[3] **Murata, T.,** 1989. Petri Nets: Properties, Analysis and Applications, *Proceedings of The IEEE*, **77**, No.4, 541-580

[4] **Pastore, E.,** 1996. Structural methods for the synthesis of asynchronus circuits from signal transition graphs, *PhD Thesis*, Universitat Politecnica de Catalunya, Barcelona, Spain.

[5] **Bobbio,A.,** 1990. Sysytem Modelling With Petri Nets, *Kluwer p.c.,pp 102-143*, Istitut Elettrotecnico Nazionale Galileo Ferraris Strada delle Cacce 91, 10135 Torino, Italy

[6] **Moody , J.O. and Antsaklis, P.J.,** 1998. Supervisory Control of Discrete Event Systems Using Petri Nets, Kluver Academic Publishers, MA.

[7] **Platzner, M. and Danne, K.,** 2006. Optimization by Integer Linear Programming, Universitat Paderborn, Germany.

[8] **Johnsonbough, R., and Murata, T.,** 1982, Petri Nets and Marked Graphs – Mathematical Models of Concurrent Computing, *The American Monthly*, **89**, Issue 8, 552-566

[9] **Jeng, M.D. and Peng, M.Y.,** 1996, Generating Minimal Siphons and Traps for Petri Nets, *1996 IEEE International Conference on Systems, Man, and Cybernetics*, Vol. 4, 2996 - 2999

[10] **Boer, E.R. and Murata, T.,** 1992, Sign Incidence Matrix and Generation of Basis Siphons and Traps of Petri Nets , *1992 IEEE International Conference on Systems, Man, and Cybernetics*, **1**, 632 – 637

[11] **Cordone, R., Ferrarini, L. And Piroddi, L.,** 2002. Characterization of Minimal and Basis Siphons with Predicate Logic and Binary Programming , *2002 IEEE International Conference on Computer Aided Control System Design Proceedings* , Glasgow, Scotland, U.K.

[12] **Çayır, S.,** 2004. Petri Ağlarında Değişmez Ananlizi, *Yüksek Lisans Tezi*, İ.T.Ü. Fen Bilimleri Enstitüsü, İstanbul

[13] **Prothero, D.,** 2000, Hardware Design And Petri Nets, , *Modelling And Implementation Of Petri Nets Using VHDL* , Kluver Academic Publishers, pp. 223-236

[14] http://www.informatik.uni-hamburg.de/TGI/PetriNets/introductions/aalst/

EKLER

EK-A: Analizleri yapılan PA'rı ve bölüm6'da bu ağlardan elde edilen minimal savak ile baz savak yapıları (aynı zamanda CD'ye kaydedilmiştir.)

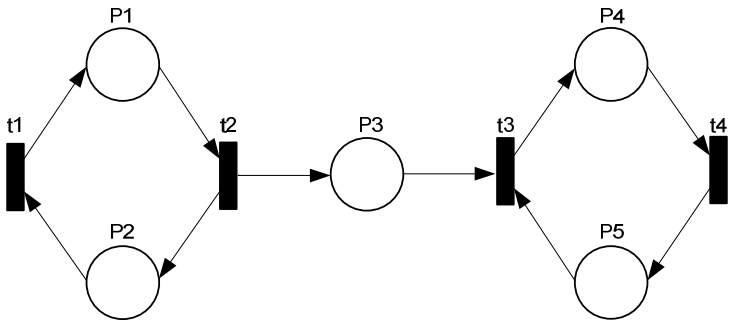
EK-B: Programların kaynak kodları (CD'ye kaydedilmiştir.)

EK-C: Tezin metin kısmına ait dosyalar (CD'ye kaydedilmiştir.)

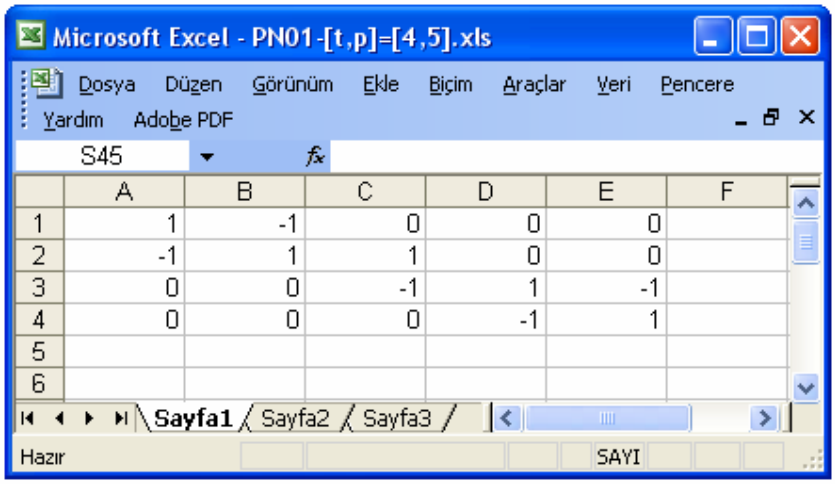
EK A

1)

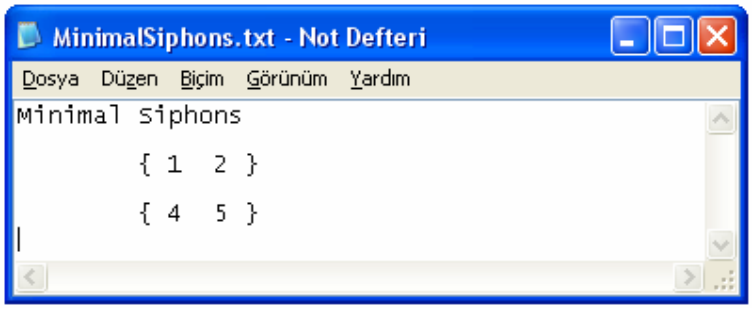
- Petri Ağı- PN01-[t,p]=[4,5].xls



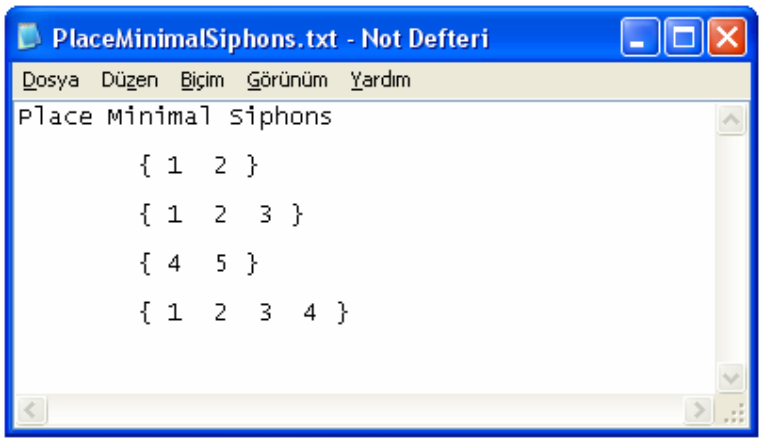
- Durum-Geçiş Matrisi (Matlab dizinindeki .xls dosyası)



- Minimal Savaklar (Matlab dizinindeki .txt dosyası)

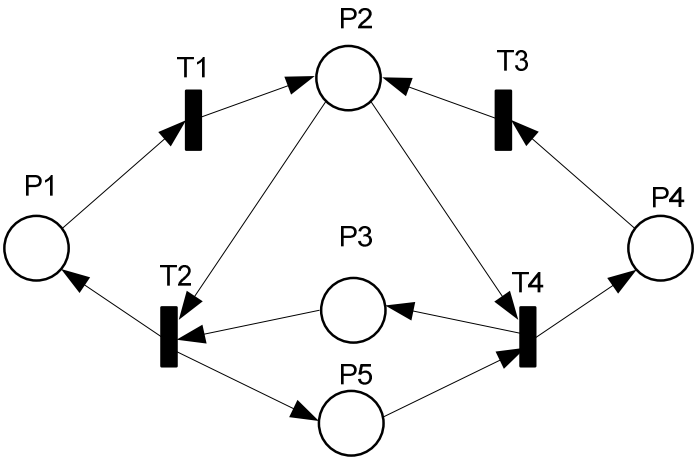


- Yer Minimal Savaklar (Matlab dizinindeki .txt dosyası)

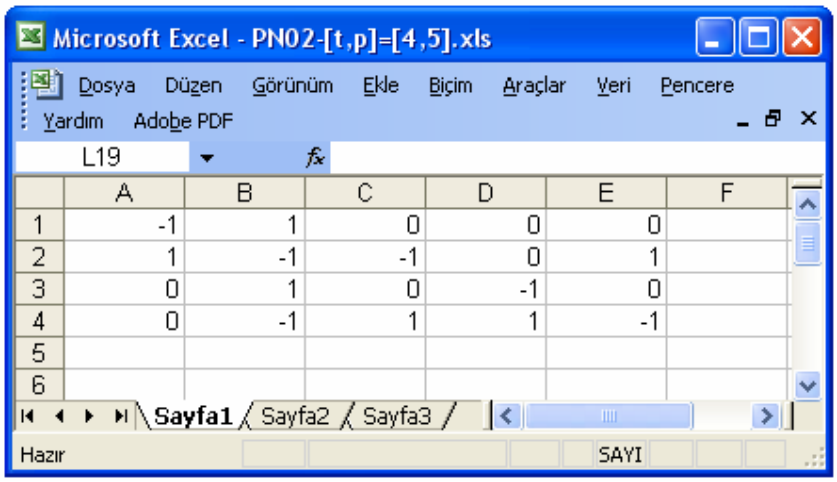


2)

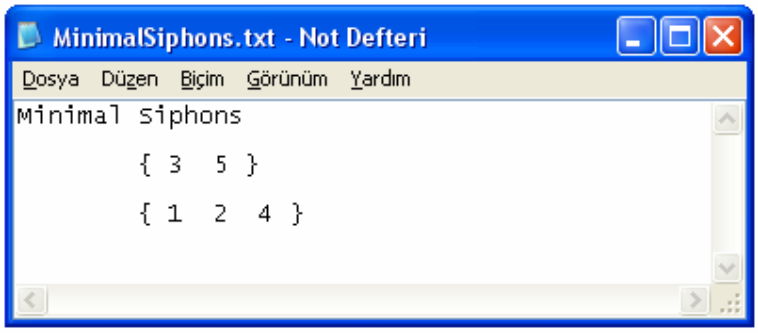
- Petri Ağı- PN02-[t,p]=[4,5].xls



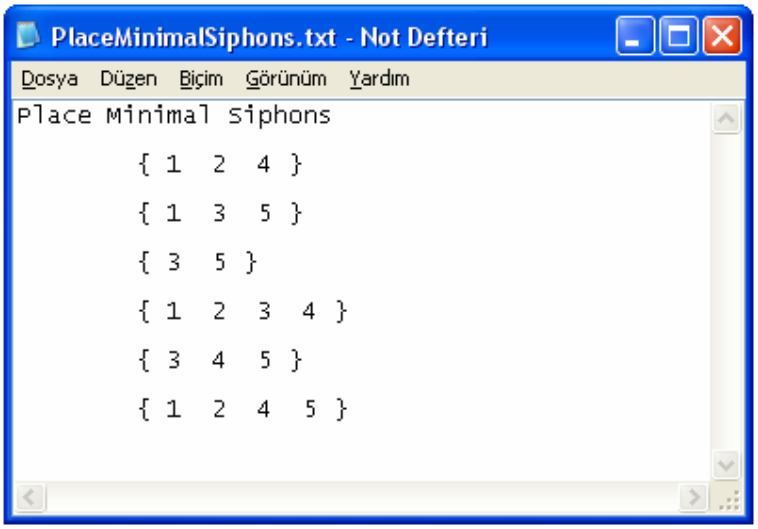
- Durum-Geçiş Matrisi



- Minimal Savaklar

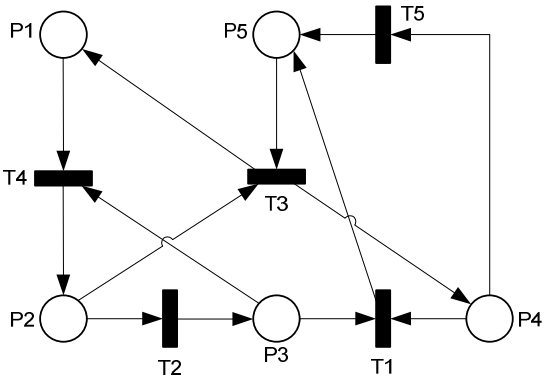


- Yer Minimal Savaklar

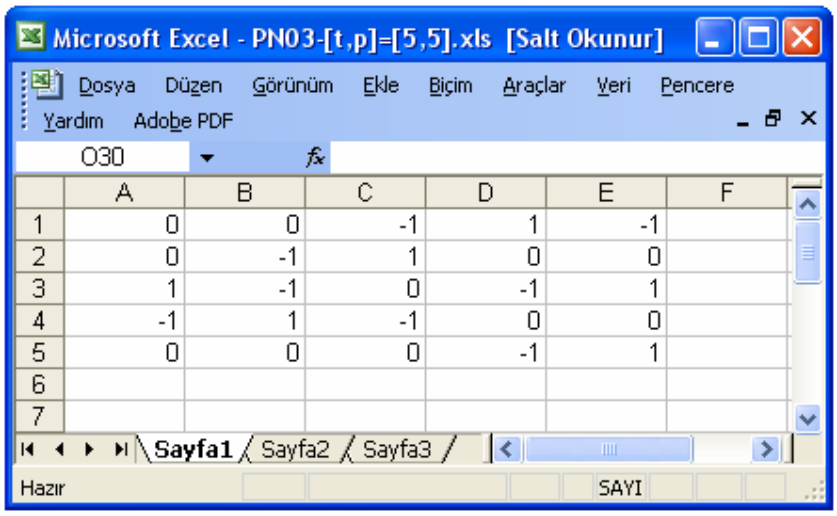


3)

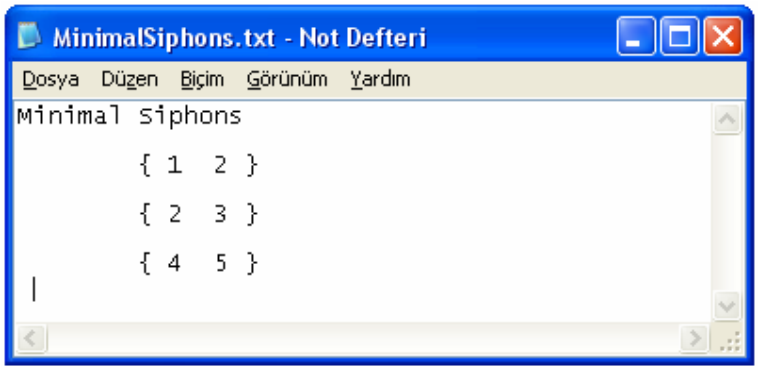
- Petri Ağı- PN03-[t,p]=[5,5].xls



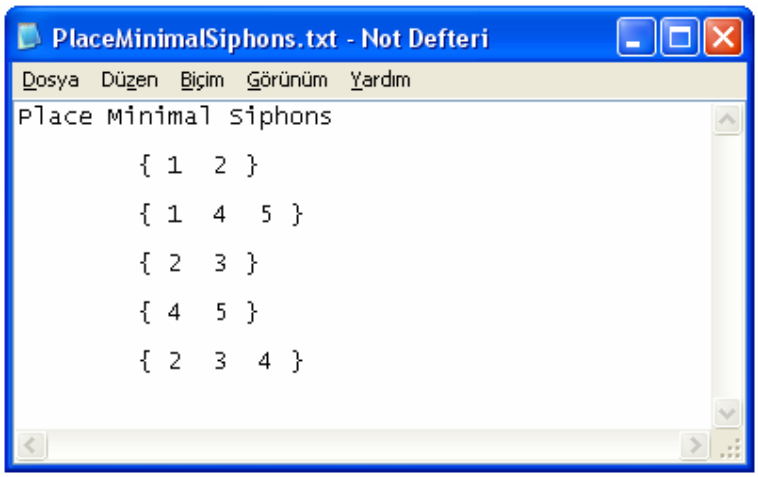
- Durum-Geçiş Matrisi



- Minimal Savaklar

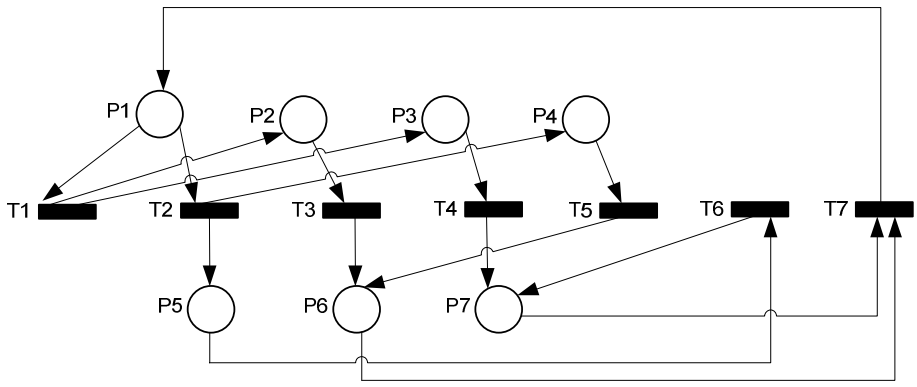


- Yer Minimal Savaklar



4)

- Petri Ağı- PN04-[t,p]=[7,7].xls



- Durum-Geçiş Matrisi

Microsoft Excel - PN04-[t,p]=[7,7].xls

Dosya Düzen Görünüm Ekle Biçim Araçlar Veri Pencere Yardım Adobe PDF

O25

	A	B	C	D	E	F	G	H
1	-1	1	1	0	0	0	0	
2	-1	0	0	1	1	0	0	
3	0	-1	0	0	0	1	0	
4	0	0	-1	0	0	0	1	
5	0	0	0	-1	0	1	0	
6	0	0	0	0	-1	0	1	
7	1	0	0	0	0	-1	-1	
8								
9								

Sayfa1 Sayfa2 Sayfa3

Hazır BH SAYI

- Minimal Savaklar

MinimalSiphons.txt - Not Defteri

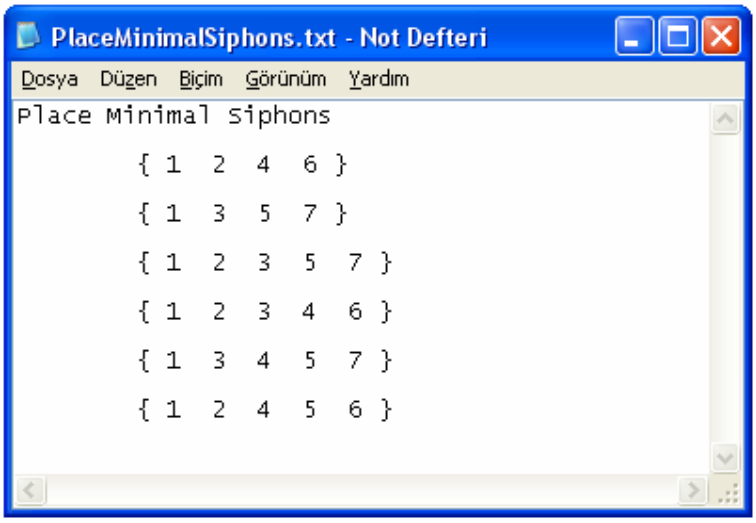
Dosya Düzen Biçim Görünüm Yardım

```
Minimal Siphons

        { 1  2  4  6 }

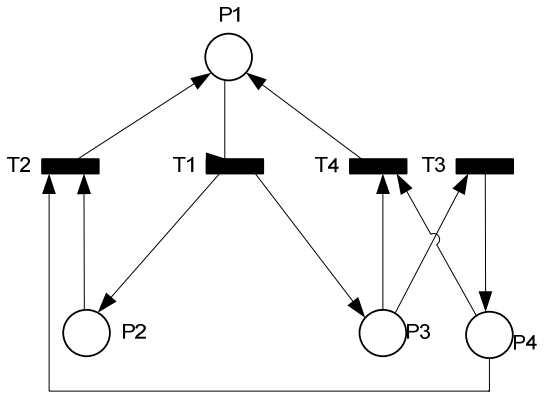
        { 1  3  5  7 }
```

- Yer Minimal Savaklar

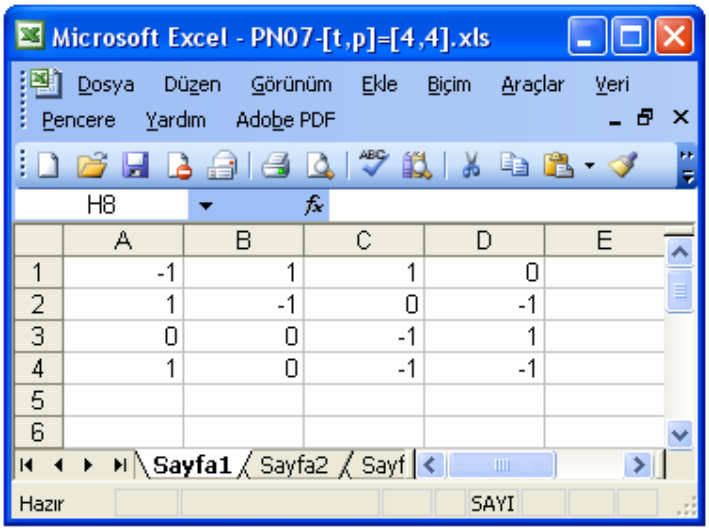


5)

- Petri Ağı- PN07-[t,p]=[4,4].xls



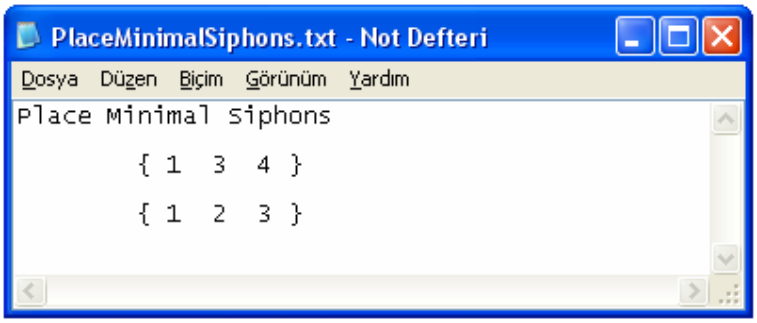
- Durum-Geçiş Matrisi



- Minimal Savaklar



- Yer Minimal Savaklar



ÖZGEÇMİŞ

Taner Varol 1981 yılında, Kırcaali/Bulgaristan'da doğmuştur. İlk ve orta eğitimini sırası ile Bursa Zafer İlk Okulu ile Yeni Mahalle İlköğretim okulunda, lise eğitimini ise Bursa Erkek Lisesinde tamamlamıştır. 1999 yılında Kocaeli Üniversitesi Elektronik ve Haberleşme Mühendisliği bölümünde başladığı lisans eğitimini 2001 yılında Uludağ Üniversite Elektronik Mühendisliği bölümüne geçerek 2004 yılında Elektronik Mühendisi ünvanı ile mezun olmuştur. 2005 yılında İstanbul Teknik Üniversitesi, Fen bilimleri Enstitüsü, Elektronik Mühendisliği programında başlamış olduğu yüksek lisans eğitimini 2007 yılında tamamlamış, bu süre zarfında Ericsson Telekominikasyon A.Ş.'de çalışmıştır.