T.C.
NİĞDE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

PETRİ AĞLARI VE RAMADGE WONHAM YÖNTEMLERİNİN AYRIK OLAY SİSTEMLERİNİN KONTROLÜNE UYGULANMASINDA YENİ MELEZ YAKLAŞIMLAR VE ÜRETİM SİSTEMLERİNE UYGULANMASI

GÖKHAN GELEN

Aralık 2010

T.C.
NİĞDE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

PETRİ AĞLARI VE RAMADGE WONHAM YÖNTEMLERİNİN AYRIK OLAY SİSTEMLERİNİN KONTROLÜNE UYGULANMASINDA YENİ MELEZ YAKLAŞIMLAR VE ÜRETİM SİSTEMLERİNE UYGULANMASI

GÖKHAN GELEN

Doktora Tezi

Danışman

Prof. Dr. Murat UZAM

Aralık 2010

Gökhan GELEN tarafından Prof. Dr. Murat UZAM danışmanlığında hazırlanan "Petri Ağları ve Ramadge-Wonham Yöntemlerinin Ayrık Olay Sistemlerinin Kontrolüne Uygulanmasında Yeni Melez Yaklaşımlar ve Üretim Sistemlerine Uygulanması" adlı bu çalışma jürimiz tarafından Niğde Üniversitesi Fen Bilimleri Enstitüsü Elektrik-Elektronik Mühendisliği Anabilim Dalında Doktora tezi olarak kabul edilmiştir.

Başkan : Prof. Dr. Galip CANSEVER, Yıldız Teknik Üniversitesi

Üye : Prof. Dr. Kenan DANIŞMAN, Erciyes Üniversitesi

Üye : Prof. Dr. Tankut YALÇINÖZ, Melikşah Üniversitesi

Üye : Prof. Dr. Murat UZAM, Niğde Üniversitesi

Üye : Doç. Dr. Saffet AYASUN, Niğde Üniversitesi

ONAY:

Bu tez, Fen Bilimleri Enstitüsü Yönetim Kurulunca belirlenmiş olan yukarıdaki jüri üyeleri tarafından …./…./20.... tarihinde uygun görülmüş ve Enstitü Yönetim Kurulu'nun …./…./20.... tarih ve ..... sayılı kararıyla kabul edilmiştir.

...../...../20...
Doç. Dr. Nurettin ACIR
MÜDÜR

ÖZET

PETRİ AĞLARI VE RAMADGE WONHAM YÖNTEMLERİNİN AYRIK OLAY SİSTEMLERİNİN KONTROLÜNE UYGULANMASINDA YENİ MELEZ YAKLAŞIMLAR VE ÜRETİM SİSTEMLERİNE UYGULANMASI

GELEN, Gökhan

Niğde Üniversitesi

Fen Bilimleri Enstitüsü

Elektrik-Elektronik Mühendisliği Anabilim Dalı

Danışman: Prof. Dr. Murat UZAM

Aralık 2010, 302 Sayfa

Son yirmi yılda bilgisayar teknolojisinde meydana gelen hızlı gelişmeler üretim sistemleri, trafik sistemleri gibi Ayrık Olay Sistemleri (AOS) (Discrete Event Systems-DES) olarak sınıflandırılan birçok sistemi de beraberinde getirmiştir. AOS'lerin kontrolünde yaygın olarak, otomata temelli Gözetimli Kontrol Teorisi ve Petri ağı temelli yöntemler olmak üzere iki yöntem kullanılmaktadır. Bu iki yaklaşımın sunduğu avantajlardan aynı anda faydalanan melez bir yaklaşım da mevcuttur.

Tez kapsamında Petri ağları ve Gözetimli kontrol teorisinin sunduğu avantajlardan aynı anda yararlanmak için yeni yekpare, indirgenmiş ve modüler yaklaşımlar ortaya konmuştur. İlave olarak Petri ağlarından otomatalara dönüşüm metotları da önerilmiştir. Petri ağı formunda hesaplanmış bir AOS denetleyicisinin doğruluk analizinin TCT yazılımı kullanılarak yapıldığı bir yöntem önerilmiştir. Kontrollü Petri ağları için derlenmiş melez denetleyicilerin hesaplanabileceği yekpare, indirgenmiş ve modüler yaklaşımlar da önerilmiştir. Bu yeni yaklaşımların tamamının gerçek sistemlerin kontrolünde kullanılabileceği PLC kontrollü bir deneysel endüstriyel üretim sistemi kullanılarak gösterilmiştir.

Bu tez kapsamında önerilen yöntemlerin, endüstrinin formal ve pratik ayrık olay kontrol sistemi tasarımındaki ihtiyaçlarını karşılama konusunda faydalı olacağı değerlendirilmektedir.

Anahtar sözcükler: Ayrık olay sistemlerin kontrolü, Otomata, Petri ağları, Kontrollü Petri ağları, PLC, Üretim sistemleri, Gözetimli kontrol teorisi,

SUMMARY

NEW HYBRID APPROCHES BASED ON PETRI NETS AND RAMADGE WONHAM METHODS FOR THE SYNTHESIS OF DISCRETE EVENT SYTEM CONTROLLERS AND THEIR APPLICATION TO MANUFACTURING SYSTEMS

GELEN, Gökhan

Nigde University

Institute of Natural Sciences

Department of Electrical and Electronics Engineering

Supervisor: Prof. Dr. Murat UZAM

December 2010, 302 pages

The last two decades have witnessed rapid developments in computer technology, which in return, has found widespread applications in manufacturing systems, traffic systems, etc. such systems fall into the category of Discrete Event Systems (DES). Two fundamental methods namely automata based Supervisory Control Theory (SCT) and Petri net based methods are widely used in control of DESs. Recently a hybrid method, making use of both SCT and Petri nets has been proposed in order to take advantage of both methods.

In this thesis, new monolithic, reduced and modular methods are proposed in order to take advantage of both Petri nets and SCT. In addition, mappings form Petri nets to automata are also proposed. A novel and general methodology is proposed to carry out the correctness analysis for the computed PN-based controllers by using the TCT implementation tool of SCT. New monolithic, reduced and modular methods are proposed to synthesize compiled supervisors for Controlled Petri nets. The applicability of all proposed methods has been demonstrated by means of an experimental manufacturing system controlled by programmable logic controller (PLC).

It is expected that the methods proposed in this thesis will be very useful to satisfy the requirements of industry in designing formal and practical discrete event control systems.

Keywords: Control of DES, Automata, Petri nets, Controlled Petri nets, PLCs, Manufacturing Systems, Supervisory Control Theory (SCT).

ÖNSÖZ

Tez kapsamında, günümüzde pek çok alanda karşılaşılan ve ayrık olay sistemi olarak sınıflandırılan sistemlerin etkin bir şekilde kontrolü konusunda araştırmalar yapılmıştır. Ayrık olay sistemlerinin kontrolü konusunda Gözetimli Kontrol Teorisi ve Petri ağlarını birleştiren melez bir yöntem Uzam ve Wonham tarafından önerilmiştir. Bu tez kapsamında, önerilen bu yöntemin gerçek sistemlere uygulanabilirliği gösterilmiştir. Ayrıca, literatürde bulunan Kontrollü Petri ağları için derlenmiş denetleyicilerin hesaplanabildiği yeni melez bir yöntem de önerilmiştir. Önerilen yöntemin etkinliğini arttırmak amacıyla indirgenmiş ve modüler yaklaşımlar da ortaya konmuştur.

Bu Doktora Tezi, TÜBİTAK 107E125 kodlu *"Ayrık Olay Sistemleri için Denetleyicilerin sentezlenmesi ve Üretim Sistemlerine Uygulanmasında Petri Ağları ve Ramadge-Wonham Yöntemlerinin Teorik ve Deneysel İncelenmesi"* adlı proje kapsamında desteklenmiştir. Bu proje kapsamında oluşturulan Siemens S7-300 PLC kontrollü deneysel endüstriyel üretim sistemi kullanılarak tezde önerilen tüm yöntemlerin gerçek sistemlerin kontrolünde de kullanılabilirliği gösterilmiştir.

TEŞEKKÜR

Doktora çalışmalarım süresince her konuda bilgi, tecrübe ve desteğini benden esirgemeyen danışmanım Prof. Dr. Murat UZAM'a, bölüm hocalarına, maddi ve manevi destekleri ile her zaman yanımda olan başta eşim Ayetül GELEN olmak üzere tüm aileme sonsuz teşekkürlerimi sunarım.

Bu Doktora tezi, 107E125 kodlu *"Ayrık Olay Sistemleri için Denetleyicilerin sentezlenmesi ve Üretim Sistemlerine Uygulanmasında Petri Ağları ve Ramadge-Wonham Yöntemlerinin Teorik ve Deneysel İncelenmesi"* adlı proje kapsamında desteklenmiştir. TÜBİTAK Kurumu ve çalışanlarına da teşekkürlerimi sunarım.

İÇİNDEKİLER DİZİNİ

ÇİZELGELER DİZİNİ

ŞEKİLLER DİZİNİ

KISALTMALAR

AGV: Automatic Guided Vehicle – Otomatik Yönlendirmeli Araç

AOS: Discrete Event System - Ayrık Olay Sistemi

APN: Automation Petri Net - Otomasyon Petri Ağı

CtlPN: Controlled Petri Net - Kontrollü Petri Ağı

FMS: Flexible Manufacturing System – Esnek Üretim Sistemi

FSA: Finite State Automata - Sonlu Durum Otomatası

PLC: Programmable Logic Controller - Programlanabilir Lojik Denetleyici

PN: Petri Net - Petri Ağı

RW: Ramadge Wonham

SCIE: Science Citation Index Expanded – Genişletilmiş Bilimsel Atıf İndeksi

SCT: Supervisory Control Theory - Gözetimli Kontrol Teorisi

SOPN: Safe and Ordinary Petri Net - Güvenilir ve Sıradan Petri Ağı

TPL: Token Passing Logic - Jeton Aktarma Lojiği

TTA: Timed Transition Automata – Zamanlı Geçişli Otomata

BÖLÜM I

GİRİŞ

Bilim ve teknolojideki hızlı gelişmeler alışılagelmiş diferansiyel eşitliklerle açıklanamayan birçok sistemi de beraberinde getirmiştir. Esnek imalat sistemleri, trafik sistemleri, haberleşme protokolleri, bilgisayar ağ sistemleri, çeşitli nakil (ulaşım) sistemleri bu sistemlere örnek olarak verilebilir. Bu sistemlerin davranışları çoğunlukla, kendi içlerinde işlemekte olan ayrık (discrete) olaylarla belirlenir. Karakteristikleri aşağıdaki gibi saptanabilen böyle sistemlere, ayrık olay sistemleri (AOS) (Discrete Event Systems-DES) veya ayrık olay dinamik sistemleri (Discrete Event Dynamic Systems-DEDS) denir [1]:

Eş zamanlılık (güdümlülük): Bir ayrık olay sisteminde, birçok işlem aynı anda meydana gelebilir.

Asenkron işlemler: Her değişimin ve adımın, global bir saat tarafından senkronize edildiği sistemlerden farklı olarak, ayrık olay sistemlerinde olaylar, çoğu kez asenkron olarak meydana gelir.

Olay sürümlülük (event-driven): AOS'ler olay oluşumlarının sistemi ifade eden durumlarda değişimler meydana getirdiği, ayrık olaylar uzayıyla karakterize edilebilir. Bu durumda, herhangi bir olay diğer olayların oluşumuna bağımlı olabilir, örneğin bir işlemin bitişi diğer bir işlemi başlatabilir.

Belirsizlik (non-determinism): Belirsizlik kesin olmayan olay oluşumları sonucunda meydana gelir, örneğin: verilen bir durumdan çok farklı gelişmeler oluşması mümkün olabilir.

AOS'ler günümüzün modern endüstriyel sistemlerinin modellemesinde, analizinde, tasarımında ve kontrolünde kullanılan yeni bir araştırma disiplini olarak ortaya çıkmıştır. Bu disiplin öncesinde karşılaşılan basit problemlerin çözümünde buluşsal (heuristic) metotların kullanımı yeterli olmaktaydı. Maalesef bu eğilim halen güncelliğini korumakta ve AOS'lerdeki kontrol problemlerinin, kontrol mühendislerinin

tecrübeye dayalı yöntemleriyle çözülmesine çalışılmaktadır. Fakat AOS'lerin gün geçtikçe daha karmaşık hale gelmesiyle bu tip sistemlerin kontrolünde kullanılmak üzere etkin bir formal yönteme olan ihtiyaç daha da artmaktadır.

Bu ihtiyacın karşılanması amacıyla, AOS'lerle ilgili yapılan formal çalışmalar genel olarak dört farklı yöntemle gerçekleştirilmektedir: otomata (automata), Petri ağları (Petri nets), minimaks ve diğer cebirsel yöntemler (minimax and other algebras), ve son olarak sıralama ağları (queuing networks) [2]. Literatürde yapılan çalışmaların yoğunluğu dikkate alındığında otomata (automata) ve Petri ağları kullanılarak gerçekleştirilen çalışmaların diğer iki yönteme göre çok fazla yer aldığını görmek mümkün olacaktır. Bu sebepten burada sadece otomata ve Petri ağları yöntemleri incelenecektir.

Sonlu Durum makinesi - (Finite State Machine - FSM) olarak da bilinen otomata temelli yöntem, sürekli sistemler için geliştirilmiş olan kontrol teorisinin AOS'lere uygulanması yönünde yapılmış en ciddi çalışmaları kapsamaktadır. FSM'ler AOS'ler için lojik modeller gerçekleştirmekte kullanılır. AOS'lerin bu yöntemle kontrolü konusunda yapılan çalışmalara öncülük eden Profesör W.M. Wonham, doktora öğrencisi P.J. Ramadge ile yapmış olduğu yayınlarla AOS'lerin kontrolü konusunda bir çığır açmıştır. O günden bu yana ***gözetimli kontrol teorisi*** olarak bilinen (Supervisory Control Theory - SCT) teori geliştirile gelmektedir. Bu teori kısaca **Ramadge-Wonham yöntemi** olarak da isimlendirilmektedir. Ramadge ve Wonham tarafından AOS'lerin kontrol edilebilirliği (controllability) konusunda çalışmalar yapılmış ve bu özellikler formal dil kavramları kullanılarak tanımlanmıştır [3-5]. Benzer çalışmalar Lin ve Wonham tarafından AOS'lerin gözlenebilirliği (observability) konusunda da yapılmıştır [6]. Elde edilen gözetici yapısı boyutunun azaltılmasına ve daha etkin kontrol sağlanabilmesine yönelik modüler, dağıtık (decentralized) ve hiyerarşik (hierarchical) kontrol yöntemleri önerilmiştir [6-9]. Bu referanslarda belirtilen çalışmalardan sonra pek çok kişi bu konuda araştırmalar yapmıştır. AOS'lerin kontrolü için ortaya konan Ramadge-Wonham yöntemi otomata kullanımı ve formal diller (formal languages) kavramları üzerine kurulmuş bir yöntemdir. Bu yöntemin sunduğu avantajları incelersek: bu yöntem AOS'lerin tamamına uygulanabilen genel bir yöntemdir. Teorinin uygulanması için geliştirilen bir yazılım mevcuttur ve genel kullanıma açıktır. Öte yandan bu yöntemin kullanımında bir kaç dezavantaj mevcuttur. Öncelikle bu

yöntemde kullanılan otomatalarla sistem modellemesi yapıldığından, genel olarak model ile modellenen sistem arasında birebir ilişki kurmak pek mümkün olmamaktadır. Ayrıca bu yöntemdeki matematik araçların kullanımını öğrenmek için başlangıçta çok zaman harcamak gerekmektedir. En önemli dezavantaj, durum patlaması (state explosion problem) adı verilen problemdir. Bu problem, AOS'yi oluşturan tüm alt modellerin artmasıyla elde edilen sonuç modeldeki durum sayısının üstel olarak artması ve sonuç modelin elde edilmesiyle ilgili hesaplamaların da zorlaşması demektir. Son yirmi yıldır yapılan çok yoğun akademik çalışmalar sonucunda yöntem iyi bir şekilde geliştirilmiş ve geliştirilmeye devam edilmektedir. Literatürde bu yöntem kullanılarak gerçekleştirilmiş uygulamaya dönük çalışmaları da görmek mümkündür [10-14]. Fakat bu yöntemin gerçek endüstriyel sistemlerin kontrolü uygulamalarında hak ettiği yeri tam olarak alamadığı da dikkate alınması gereken çok önemli bir noktadır.

AOS'lerin kontrolü ile ilgili olarak çok yaygın olarak kullanılan ikinci yöntem grubunda **Petri ağları**nın kullanımı söz konusudur. Grafiksel ve matematiksel bir araç olarak Petri ağları, ayrık olay sistemlerinin modellemesinde, analizinde, tasarımında ve kontrolünde giderek artan bir hızla kullanılagelmektedir [15]. Petri ağları, otomata ile haberleşme çalışmaları için ağ (net) şeklinde matematiksel bir araç ortaya koyan Alman matematikçi Carl A. Petri'den sonra bu ismi almıştır [16]. O zamandan bu yana üretim sistemleri, bilgisayar bilimi, haberleşme sistemleri v.b. gibi çeşitli bilim dallarında büyük çapta araştırmalar yapılmıştır [17]. Petri ağları herhangi bir türden ayrık olayın modellenmesini mümkün kılar. Petri ağları iki önemli karakteristik sunar. Öncelikle, eş zamanlılık, senkronizasyon ve kaynak paylaşımını içeren durumların modellenmesini ve göz önünde canlandırılmasını mümkün kılarlar. İkinci olarak, Petri ağları yardımıyla verimli kuramsal sonuçlar ortaya konulmuştur [18]. Petri ağlarının modellemede, analizde, simülasyonda ve üretim sistemlerinin kontrolünde çok faydalı oldukları ispatlanmıştır. Petri ağları aşağıdaki nedenlerden dolayı çok faydalı modeller sunarlar [19]:

- Petri ağları öncül ilişkilere (precedence relations), stokastik, eş zamanlı ve asenkron olayların yapısal etkileşimlerine hâkim olurlar. Buna ek olarak grafiksel tabiatları böyle karmaşık sistemleri göz önünde canlandırmaya yardımcı olur.
- İkilemler (conflict) ve tampon (buffer) boyutları kolayca ve etkin bir şekilde modellenebilir.
- Sistemdeki tıkanmalar (deadlocks) tespit edilebilir.

- Petri ağı modelleri iyi geliştirilmiş matematiksel ve pratik temeller ile hiyerarşik bir modelleme aracını temsil ederler.
- Zamanlı Petri ağları, stokastik Petri ağları, renkli Petri ağları ve yüklemli (predicate) geçiş Petri ağları gibi çeşitli genişletilmiş Petri ağları, kaynak kullanımının, hataların etkisinin ve aktarım (throughput) oranının hem kalitatif hem de kantitatif analizinin yapılmasına olanak sağlar.
- Petri ağı modelleri, hem karmaşık sistemlerin sistematik analizinin yapılmasında hem de ayrık olay kontrolörlerinin sistematik tasarımında kullanılabilir.
- Son olarak, Petri ağı modelleri esnek ve hızlı imalat sistemlerinin gerçek-zamanlı (real-time) kontrolunun gerçekleştirilmesinde de kullanılabilir.

Sıradan Petri ağları, karmaşık endüstriyel sistemleri ifade etmekte her zaman yeterli olamazlar. Bu sebepten yeni Petri ağı sınıfları geliştirilmektedir. Örneğin, birçok benzer etkileşimli aktivitelerden oluşan karmaşık sistemleri modellerken, sıradan Petri ağları, modelin grafiksel karmaşıklığını arttırırlar. Bu konuyu tanımlamak amacıyla, farklı özdeşlikler elde etmek için Petri ağlarında kullanılan ***jetonlara*** özel nitelik veren Petri ağları önerilmiştir. Bu ağlar, Genrish ve Lautenbach tarafından önerilen yüklemli geçiş ağlarını [20], Jensen'in önerdiği renkli ağlar [21] ve kendine has jetonları olan ağları [22] içeren yüksek seviyeli Petri ağlarındır. Sibertin-Blanc tarafından obje yöneltili (object oriented) Petri ağlarının ortaya konması yüksek seviyeli Petri ağları alanında önemli bir ilerleme olmuştur [23]. Yaklaşık olan ve kesinlik taşımayan bilgileri ifade etme ihtiyacı çeşitli Fuzzy Petri ağı tiplerinin ortaya çıkmasına neden olmuştur [24-27]. Sistemlerin zamana bağlı analiz ihtiyacı, zamana bağlı Petri ağların (timed Petri nets) ortaya konmasıyla sonuçlanmıştır [28].

AOS'lerin kontrolü için ortaya konan Ramadge-Wonham yöntemine alternatif olarak Petri ağı temelli yöntemler de geliştirilmiştir [30-42]. Bu yöntemlerin ortaya konmasının asıl nedeni Petri ağı kullanılarak geliştirilen yöntemlerin sonlu durum makineleri (FSM) kullanılarak geliştirilen yöntemlere göre bir takım avantajlar sunuyor olmasıdır [29]. Öncelikle, bir Petri ağının durumu o an ağdaki ***jetonların*** dağılımı tarafından belirlenirken FSM'de sistemin içinde bulunduğu birbirinden farklı her bir olasılığı ifade etmek için bir durum (state) kullanılmak zorundadır. Böylece Petri ağının yapısı sistemdeki ***jeton*** sayısı artsa bile sisteme ait daha derli toplu (compact) bir model

sunabilir. İkinci olarak, FSM'ler anlaması zor pek çok karmaşık metinsel tanımlamalar ve matematik notasyonların kullanılmasını gerektirirken; Petri ağlarının kullanımında ise modellenen sistemi ve spesifikasyonları anlaşılır ağ yapıları şeklinde ifade etmek mümkün olduğundan fiziksel sistemle elde edilen Petri ağı modeli arasında bağ kurmak ve modeli çok iyi anlayıp kullanmak mümkün olmaktadır. Son olarak, Petri ağı modellerinin kullanımında, aynı model sisteme ait özelliklerin analiz edilmesi, performans gelişimi ve ayrık olay kontrolörlerinin sistematik tasarımı için kullanılabilmektedir. Literatürde, Petri ağı temelli denetleyicilerin (Petri net based supervisors) genelde iki farklı şekli mevcuttur [29]. Birinci gruptaki denetleyiciler sistem tarafından üretilen her bir yeni olay (event) için hesaplanan bir fonksiyon şeklinde ifade edilen ***gönderim denetleyicileri*** (*mapping supervisors)* olarak isimlendirilirler. ***Derlenmiş denetleyiciler*** (*compiled supervisors*) olarak isimlendirilen ikinci gruptaki denetletleyicilerde ise, kontrol mekanizması bir ağ yapısı şeklinde gösterilmektedir. Denetleyicilerin bir ağ şeklinde ifade edilmesinin pek çok faydası vardır [29]: Öncelikle, çevrim-içi (on-line) bir hesaplama gerektirmediğinden kontrol işlemi için gereken hesaplama daha hızlıdır. İkinci olarak, hem kontrol mekanizması hem de sistem modeli ağ yapısı şeklinde ifade edildiğinden aynı Petri ağı sistem yürütme algoritmaları her iki ağ yapısını gerçekleştirmek için kullanılabilir. Son olarak, kontrol edilecek sisteme ait kapalı çevrim (closed-loop) kontrol modeli standart ağ elde etme yapıları kullanılarak gerçekleştirilebilir. Bu açıklanan üstünlüklerinden dolayı *derlenmiş denetleyiciler*, *gönderim denetleyicileri*ne göre daha fazla tercih edilmektedir.

Holloway ve Krogh tarafından AOS'lerin kontrolü için denetleyicilerin *gönderim denetleyicisi* olarak hesaplandığı ve sadece belirli bir Petri ağı alt kümesine (cyclic controlled marked graphs) uygulanabilen yöntemler önerilmiştir [30-31]. Daha sonra Petri ağı mevki değişmezleri (Petri net place invariants) kavramına dayalı olarak *derlenmiş denetleyicileri*n etkin bir biçimde hesaplanması için çok önemli bir metot Yamalidou ve diğerleri tarafından ortaya kondu [36]. Maalesef bu yöntemde de bazı yetersizlikler söz konusudur. Öncelikle sisteme ait spesifikasyonların ya da kısıtların (constraints) bir *mevki değişmezi* (place invariant) olarak verildiği varsayılmaktadır. Fakat bu durum her zaman mümkün olmamaktadır. Ayrıca bu yöntemde kontrol edilemeyen geçişlerin (uncontrollable transitions) dikkate alınması mümkün olmamaktadır. Son olarak, bu yöntemde kısıtların istenilen formatta ifade edilmesi sonucunda denetleyici boyutları aşırı derecede büyümektedir.

Literatürde, Ramadge-Wonham yönteminin ortaya koyduğu yaklaşıma benzer bazı yöntemlerin Petri ağlarıyla birlikte kullanımına ait örnekleri de görmek mümkündür [36-37]. Fakat burada problem, önerilen yöntemin uygulanması için mevcut bir bilgisayar yazılımının bulunmamasıdır. Bu noktada adı geçen problemin çözümü için Ramadge-Wonham yönteminin uygulanması için geliştirilen TCT bilgisayar programının [68] kullanımı faydalı olacaktır.

Badouel ve Darondeau tarafından ortaya konan bölgeler teorisi (theory of regions) [43] adında bir teori yardımıyla otomata şeklinde verilen bir modelden Petri ağı elde edilebileceği anlaşıldı ve böylelikle bu teorinin AOS'lerin kontrolünde kullanımı da gündeme gelmiş oldu [44-47]. Bu çalışmalar yardımıyla Petri ağı şeklindeki bir AOS için verilen kısıtları ifade etmek için *derlenmiş denetleyicilerin* elde edilmesi mümkün olabildi. Fakat bu yöntemlerin kullanımında bir kaç problem mevcuttur. Öncelikle bu teorinin çıktı olarak bir Petri ağı üretebilmesi için verilen otomata modelini ifade eden bir Petri ağı modelinin bulunması gerekir. Fakat bazı durumlarda verilen bir otomata modelini ifade edecek bir Petri ağı bulunmamaktadır [48 - Figure 7]. Ayrıca verilen otomata modeli büyüdükçe bu modelin Petri ağı karşılığını elde etmek için gereken hesaplama işlemi de üstel olarak zorlaşmaktadır. Ramadge-Wonham yönteminin sunduğu en önemli avantajlardan biri, bir AOS'deki hem kontrol edilebilir (controllable) hem de kontrol edilemez (uncontrollable) olayların (events) birlikte dikkate alınabilmesi ve ilgili kontrol probleminin en az kısıtlayıcı (maximally permissive) ya da optimum olarak çözülebilmesidir. Petri ağı temelli bazı problemlerde [49-50] verilen bir kısıtlama problemi için *derlenmiş denetleyicilerin* kullanımıyla optimum olarak çözülmesi imkansızdır. Bu tip durumlarda optimum-altı (sub-optimal) çözüm yöntemlerinin kullanımı önerilmiştir [51]. Bu tip problemlerin çözümüne alternatif başka bir yaklaşım olarak Petri ağı ve otomata modelleri yardımıyla elde edilen bir melez yöntem Uzam ve Wonham tarafından önerilmiştir [52]. Bu yöntemde, kontrol edilecek AOS'nin Petri ağı modeli ve ilgili kısıtlar öncelikle otomata modellerine dönüştürülüp daha sonra Ramadge-Wonham yöntemi kullanarak ilgili problem için bir otomata modeli elde edilir. Daha sonra bu modelin a*uto-net* karşılığı oluşturulur. Son olarak elde edilen *auto-net* ile Petri ağı modeli kontrol poliçesini ifade etmek için kullanılan yasaklama okları (inhibitor arcs) yardımıyla birbirine bağlanarak kapalı çevrim kontrol sistemi elde edilmiş olur. Bu melez yöntemin özelliği verilen bir

kısıtlama problemi için *derlenmiş denetleyicilerin* kullanımıyla optimum olarak çözülmesi imkansız olan problemlere uygulanabilir nitelikte olması ve sonuçta bu tip problemlerin optimum çözümüne olanak vermesidir. [52]'de önerilen yöntemin uygulanabilirliği, literatürde daha önce incelenmiş bir problemin çözümü verilerek başarılı bir şekilde gösterilmiştir. Ancak, bu yöntemin bir AOS'nin gerçek-zamanlı (real-time) kontrolüne uygulanması bu doktora tezi kapsamında yapılan çalışmalar öncesine kadar literatürde rapor edilmemiştir.

Buraya kadar AOS'lerin kontrolünde çok yaygın olarak kullanılan iki formal yöntem olan Petri ağı ve otomata temelli Ramadge-Wonham yöntemleri ile ilgili literatür analizi ele alınmıştır. Bu yöntemlerin endüstride kullanılan gerçekleştirilme (implementation) şekilleri üzerinde bir inceleme aşağıda sunulmaktadır:

Bilgisayar teknolojisinin ortaya çıkmasından önce endüstriyel sistemlerin kontrolünde yaygın olarak kullanılmakta olan sistemler elektro-mekanik kumanda sistemleri olarak bilinen ve kontaktör ya da rölelerden oluşan sistemlerdi. Bilgisayar teknolojisinin gelişmesiyle beraber elektro-mekanik kumanda sistemlerinin yerini programlanabilir lojik denetleyici (Programmable Logic Controller - PLC) adı verilen yeni ürünler almıştır [53]. PLC'lerin elektro-mekanik kumanda sistemlerine göre pek çok avantajı vardır. Az yer kaplamaları, az enerji harcamaları, esnek yapıları, endüstriyel ortamlarda (nemli, sıcak, titreşime maruz) çalışabilecek şekilde üretilmeleri, bilgisayar teknolojisinde kullanılan tüm imkânların PLC'lerde de kullanılabilmesi gibi pek çok özelliklerinden dolayı günümüzde PLC'ler endüstriyel sistem kontrolünde çok yaygın olarak kullanılmaktadır. PLC'lerde kullanılan programlama dilleri IEC 61131-3 numaralı uluslararası bir standart tarafından tanımlanmıştır [54]. Bu diller arasında daha önceleri elektro-mekanik kumanda sistemlerinde kullanılan ve merdiven basamakları şeklinde çizilen diyagramlara benzeyen programlama dili olan merdiven diyagramı (ladder diagram) dili en yaygın olarak kullanılan programlama dilidir [55].

Literatürde Ramadge-Wonham yöntemiyle elde edilen denetleyicilerin PLC'ler yardımıyla gerçekleştirilmesi ile ilgili olarak yapılmış olan çalışmalar mevcuttur. Bu çalışmalara bir kaç örnek şu referanslarda görülebilir: [56-57]. Konu ile ilgili yapılan çalışmalar incelendiğinde Ramadge-Wonham yöntemiyle elde edilen denetleyicilerin PLC'ler yardımıyla ve merdiven diyagramı dili ile gerçekleştirilmesine dair kabul

görmüş ve kullanılan genel bir yöntemin bulunmadığı anlaşılacaktır. Bunun en önemli sebeplerinden birisi Ramadge-Wonham yönteminde elde edilen denetleyicilerin yapısıdır. Çünkü bu yapıda denetleyici (supervisor) bir otomato modeli şeklindedir ve denetleyicinin farklı durumlarında yasaklanması gereken olaylar (events) bir liste halinde verilmektedir. Bu model ve yasaklama listesi endüstride (pratikte) uygulana gelen işleyişle tam olarak örtüşmediği için bir haritalama problemi ve dolayısıyla da gerçekleştirmede bir takım sıkıntılar yaşanmaktadır.

Her ne kadar merdiven diyagramı ile program yazmak kolay olsa da karmaşık sistemlerin kumandasında yapısal özelliklerinden dolayı merdiven diyagramı ile program yazmak oldukça zorlaşır. Ayrıca merdiven diyagramı ile program geliştirirken hata ayıklama işlemi karmaşık sistemlerde hiç te kolay bir işlem değildir. Merdiven diyagramıyla yazılmış bir programın bir başka programcı tarafından anlaşılması ise neredeyse imkânsızdır. Bu gibi problemlerin en temel nedeni PLC programlarının endüstrideki yaygın kullanım şekli olan programcının kendine has bilgi, beceri ve yeteneğine bağlı olarak geliştirilmesidir. Bir başka deyişle PLC programlamada yaygın ve genel bir metot eksikliği günümüz endüstrisinin yaşadığı en büyük problemlerden biridir. Bu problemin çözümü için son yıllarda pekçok çalışma yapılmıştır. Petri ağlarının genel yapısı gereği ayrık işlemler olarak tanımlanabilen otomasyon sistemlerinin modellenmesi ve bu modeller yardımıyla kontrol sistemlerinin oluşturulması mümkün olmaktadır. Böylelikle kontrol edilecek olan bir sisteme ait kontrol sistem modeli bir Petri ağı olarak elde edilebilir ve daha sonra da bu Petri ağından kontrol koduna dönüşüm yapılabilir olduğu literatürde son 10-15 yıldır üzerinde çalışılan konular olmuştur. Petri ağlarının PLC'lerde kullanılmak üzere merdiven diyagramına dönüşümü ile ilgili olarak genel bir metot [58]'de ortaya konulmuştur. Bu yöntemin gerçek sistemlere uygulanabilirliği ile ilgili çalışmalar da rapor edilmiştir [59-61]. Konu ile ilgili yapılan araştırmalar literatürde önemli bir dönüm noktası teşkil etmiştir [55, 62-64]. AOS'lerin Petri ağı temelli kontrolü için merdiven diyagramı kullanımı hakkında kapsamlı bir çalışma [55]'te mevcuttur. Her ne kadar gerçek sistemlerin kontrolünde Petri ağlarının kullanımı söz konusu olsa da [65], ortaya konan yöntemlerin halen PLC'leri geleneksel yöntemlerle kullanmakta olan endüstri tarafından tam olarak kabul edildiğini söylemek mümkün değildir. Bu sebepten konuyla ilgili ortaya konan bu yöntemlerin daha da geliştirilmesi ve gerçek sistemlerde uygulanması konusunda yeni çalışmalar yapılmasına ihtiyaç vardır.

Bu tezin amacı endüstriyel sistemlerin kontrolünün gerçekleştirilmesi için yeni ve etkin yöntemler geliştirmek ve bu yöntemleri gerçek bir sistem üzerinde test etmektir. Bu amacı gerçekleştirmek üzere ayrık olay sistemlerinin (AOS) kontrolünde Petri Ağları ve Ramadge-Wonham Yöntemlerinin sunduğu imkânlardan aynı anda faydalanmak için bu yöntemlerin beraberce kullanımını öngören araştırmalar yapılmıştır. Geliştirilen yöntemlerin tabiatları gereği ayrık olan üretim sistemlerine ve benzer diğer ayrık sistemlere uygulanması beklenmektedir. Tez kapsamında gerçekleştirilen çalışmalar genel başlıklar halinde şunlardır:

1. Ayrık olay sistemlerinin kontrolünde literatürde ilk kez Petri Ağları ve Ramadge-Wonham yöntemlerinin birlikte kullanımının öngörüldüğü, tez danışmanı Prof. Dr. Murat Uzam ve Prof. Dr. W. M. Wonham tarafından daha önce önerilmiş olan yöntemin [52] gerçek sistemlerin kontrolünde uygulanabilirliği gösterilmiştir.
2. Ayrık olay sistemlerinin kontrolünde Petri Ağları ve Ramadge-Wonham yöntemlerinin sunduğu imkânlardan aynı anda faydalanmak için bu yöntemlerin beraberce kullanımını öngören yeni yaklaşımlar ortaya konmuştur.
3. Genişletilmiş bir Petri ağı sınıfı olan Kontrollü Petri ağları (CtlPN) [74, 75] için derlenmiş denetleyicilerin sentezlenmesine olanak sağlayan yeni melez yaklaşımlar ortaya konmuştur.
4. Geliştirilen yöntemlerin uygulanabilirliği bir PLC tarafından kontrol edilen gerçek bir sistem üzerinde gösterilmiştir.

İlk olarak yapılan çalışmada [52]'de önerilen yöntemin gerçek sistemlere uygulanabilirliği, Niğde Üniversitesi Elektrik Elektronik Mühendisliği Bölümü Laboratuarında bulunan uzaktan kumandalı otomatik kapı modeli kullanılarak ve deneysel endüstriyel üretim sistemi kullanılarak başarılı bir biçimde gösterilmiştir. Bu çalışmaların sonuçları [66-67]'de yayınlanmıştır.

İkinci olarak, AOS'lerin kontrolünde Petri Ağları ve Ramadge-Wonham yöntemlerinin sunduğu imkânlardan aynı anda faydalanmak için bu yöntemlerin beraberce kullanımını öngören yeni yöntemler üzerinde çalışılmıştır. Bu kapsamda ilk olarak, [52]'de

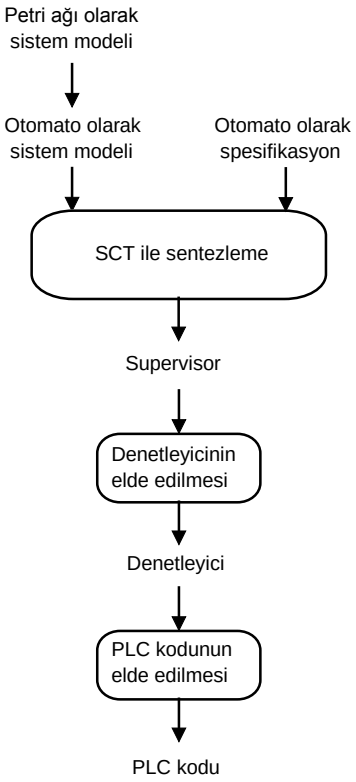
önerilmiş olan melez yöntemin indirgenmiş gözetici yapısına sahip olan biçimi incelenmiştir. Daha sonra, [52]'de önerilen yöntemin modüler biçimi önerilmiştir.

Üçüncü olarak, literatürde bulunan Kontrollü Petri ağları (CtlPN) [74,75] ile Ramadge-Wonham yöntemlerinin birlikte kullanımını öngören yeni melez yöntemler önerilmiştir. Bu yöntemler kullanılarak CtlPN'ler için derlenmiş denetleyicilerin hesaplanması mümkündür.

Son olarak, önerilen bu yeni melez yaklaşımların deneysel bir endüstriyel üretim sistemi kullanılarak gerçek sistemlerin kontrolünde uygulanabilirliği gösterilmiştir.

Tez kapsamında ikinci ve üçüncü adımda yapılan çalışmaların genel yapısı Şekil 1.1'de görülmektedir. Buna göre kontrol edilecek olan bir AOS ile ilgili Petri ağı modeli öncelikli olarak elde edilmektedir. Daha sonra bu modeldeki Petri ağı mevkilerinin her biri ilgili otomata modellerine dönüştürülür. AOS ile ilgili spesifikasyon da bir otomata modeli olarak ifade edildikten sonra Ramadge-Wonham yöntemi (Supervisory Control Theory - SCT) kullanılarak sisteme ait gözetici, bir automata modeli olarak elde edilir. Bundan sonraki aşamada elde edilen gözeticiden hareketle bir denetleyici elde edilmesine dönük araştırmalar tez kapsamında yapılan çalışmaları içermektedir. Elde edilen denetleyici yapısının PLC merdiven diyagramı koduna dönüşümü için daha önce tez danışmanının da yer aldığı bir ekip tarafından önerilen yöntemden [58] faydalanılmıştır.

Tez kapsamında yapılan çalışmalarda, gözetimli kontrol teorisinin uygulamaları için TCT yazılımı [68] , PLC kodlarının yazılması, derlenmesi ve PLC'ye yüklenmesi için SIEMENS SIMATIC Manager 5.4 Programı ve yazılan bazı exe programlarda C derleyicisi olarak CodeBlocks (http://www.codeblocks.org/) yazılımı kullanılmıştır.



Şekil 1.1 Tez kapsamında yapılan çalışmaların genel yapısı

Tezin organizasyonu aşağıdaki gibidir. İkinci bölümde, ayrık olay sistemleri, gözetimli kontrol teorisi, otomatalar, Petri ağları, Kontrollü Petri ağları, Petri ağlarının PLC ile gerçekleştirilmesi ve deneysel endüstriyel üretim sistemi hakkında temel bilgiler sunulmuştur.

Üçüncü bölümde, [52]'de önerilen yöntemin deney düzenekleri kullanılarak gerçek sistemlere uygulanabilirliğinin gösterilmesi konusunda yapılan çalışmalar yer almaktadır.

Dördüncü bölümde, [52]'de önerilen yöntemle elde edilmiş gözeticinin durum sayısının azaltılması ile elde edilen indirgenmiş gözetici içeren denetleyici yapısının gerçekleştirilmesi konusunda yapılan çalışmalar açıklanmıştır. Ayrıca bu kısımda, indirgenmiş gözeticinin PLC ile gerçekleştirilmesi esnasında karşılaşılan çığ etkisi problemi (avalanche effect problem) ve bu problemin çözümüne yönelik önerilen metot açıklanmaktadır.

Beşinci bölümde, [52]'de önerilmiş olan yöntemde kullanılan Petri ağından otomata

modellerine dönüşüm konusunda önerilen genel yaklaşımlar ve bu yaklaşımların uygulamaları açıklanmıştır.

Altıncı bölümde, [52]'de önerilen yöntemin etkinliğini arttırmak amacı ile önerilen modüler melez yaklaşımlar ve gerçek sistemlere uygulanabilirliği konusunda yapılan çalışmalar yer almaktadır.

Yedinci bölümde, Kontrollü Petri ağları (CtlPN) için derlenmiş denetleyicilerin sentezlenmesine olanak sağlayan yeni melez yaklaşımlar konusunda yapılan çalışmalar açıklanmıştır. Ayrıca bu bölümde önerilen yöntemlerin gerçek sistemlere uygulanabilirliği de gösterilmiştir.

Son olarak, tez kapsamında tamamlanan çalışmalardan elde edilen sonuçlar ve öneriler sekizinci bölümde sunulmuştur.

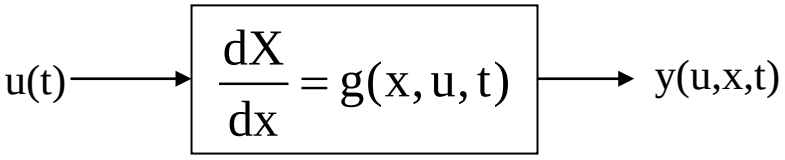
BÖLÜM II

TEMEL KONULAR

Bu bölümde, tez kapsamında incelenen araştırma konuları hakkında tanıtıcı temel bilgiler sunulmuştur. Bu konular, ayrık olay sistemleri, gözetimli kontrol teorisi, otomatalar, Petri ağları, Kontrollü Petri ağları, Petri ağlarının PLC ile gerçekleştirilmesi ve tez kapsamında oluşturulan deney düzeneğini kapsamaktadır. Ayrık olay sistemlerinin diğer sistemlerden farklarının görülebilmesi açısından ilk olarak klasik sistemler tanıtılmıştır.

2.1. Klasik Sistemler ve Sistem Kontrolü

Klasik bir sistem, birbiriyle etkileşimli alt bileşenlerden oluşan ve çıkışı bir fonksiyon olarak tanımlanabilen yapıdır. Bir sistemin giriş ve çıkışlarını gösteren blok diyagram Şekil 2.1'de görülmektedir [112].



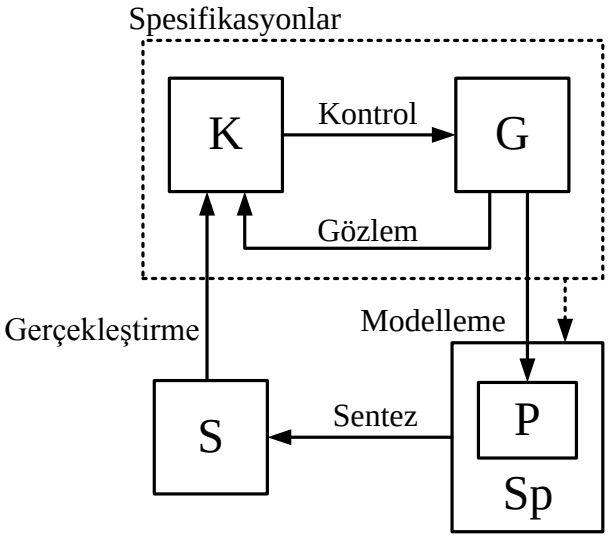
Şekil 2.1 Klasik sistem modeli

Burada u giriş işareti, y çıkış işareti, x ise sistemin durumlarını belirten değişkenlerdir. Lineer bir sistem için y çıkış ifadesi; girişler, sistemin durumları ve zamanın bir fonksiyonudur. Lineer bir sisteminin giriş, çıkış ve durumları arasındaki ilişki aşağıda görüldüğü gibi ifade edilir.

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{2.1}$$

Bu denklemlerden de görülebileceği gibi klasik sistemlerde girişler (u), çıkışlar (y) ve durumlar (x) sürekli fonksiyonlardır.

Gerçek bir sistemin kontrolü için izlenmesi gereken adımlar Şekil 2.2'de görülmektedir. Bu aşamalardan *modelleme* ile gerçek bir sistem olan G'nin P isimli modeli oluşturulur. Gerçek bir sistem olan G ve gerçek denetleyici K'nın birlikte oluşturduğu kapalı çevrim, sistemin özellikleri Sp *spesifikasyon* olarak modele aktarılmaktadır. P ve Sp'yi dikkate alarak bir S kontrolörünün hesaplanması işlemine *Sentez* denilmektedir. Sentezlenmiş bir S denetleyicisinin reel karşılığı olan fiziksel bir donanımın yapılması işlemine ise *gerçekleştirme* denilmektedir. Şekilde gösterilmeyen bir aşamada *doğrulama*dır. Bu aşama ile oluşturulan model (P) ve sentezlenen denetleyicinin (S) spesifikasyonları (Sp) sağladığı test edilmektedir [112].



Şekil 2.2 Kontrol aşamaları

2.2 Ayrık Olay Sistemleri

Giriş ve çıkış uzayları ayrık kümelerden oluşan, durumları tam sayı (0, 1, 2, 3, ..) veya sembolik etiket (boşta, çalışıyor, arızalı, ..) olan sistemler ayrık olay sistemleri olarak adlandırılmaktadır. Bu sistemlerin en belirgin özellikleri sistemin durumlar arası geçişlerinin asenkron ve olay sürümlü olmasıdır [1]

Ayrık olay sistemlerinde durumlar arası geçişlerin anlık gerçekleştiği kabul edilir. Anlık gerçekleşmeyen olaylar da modellenebilir. Böyle sistemler diferansiyel veya fark denklemleri kullanılarak modellenemezler. Ayrık olay sistemlerine örnek olarak aşağıdaki sistemler sayılabilir:

- Sıralama sistemleri
- Haberleşme sistemleri
- Üretim sistemleri
- Trafik sistemleri
- Yığın işlemleri
- Veri tabanı sistemleri
- Kontrol sistemlerinin gözlenmesi

2.2.1 Ayrık olay sistemlerinin karakteristik özellikleri

Ayrık olay sistemlerinin karakteristik özellikleri aşağıdaki gibi belirlenmektedir [1]:

Eş zamanlılık: Bir ayrık olay sisteminde, birçok işlem aynı anda meydana gelebilir.

Asenkron işlemler: Her değişimin ve adımın, global bir saat tarafından senkronize edildiği sistemlerden farklı olarak, ayrık olay sistemlerinde olaylar, çoğu kez asenkron olarak meydana gelir.

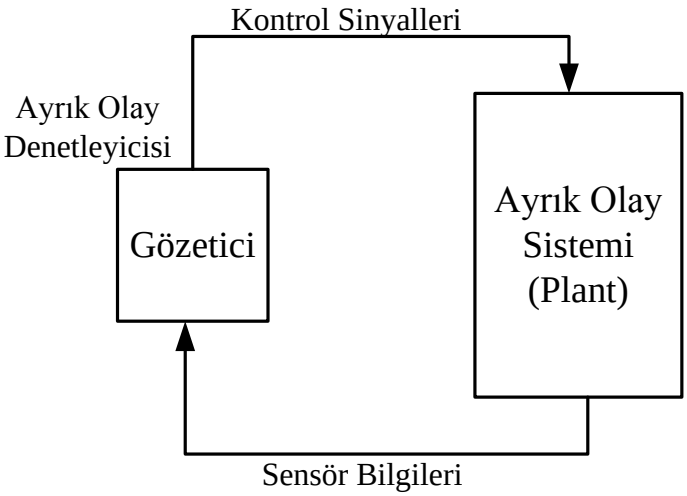
Olay sürümlülük: AOS'ler olay oluşumlarının sistemi ifade eden durumlarda değişimler meydana getirdiği, ayrık olaylar uzayıyla karakterize edilebilir. Bu durumda, her hangi bir olay diğer olayların oluşumuna bağımlı olabilir, örneğin bir işlemin bitişi diğer bir işlemi başlatabilir.

Belirsizlik: Belirsizlik, kesin olmayan olay oluşumları sonucunda meydana gelir, örneğin: verilen bir durumdan çok farklı gelişmeler oluşması mümkün olabilir.

2.3 Ayrık Olay Sistemlerinin Kontrolü

Spesifikasyonların yerine getirilmesi işlemine ayrık olay sistemlerinin kontrolü denilmektedir. AOS'lerin kontrolü Gözetimli Kontrol ile eş anlamlı olarak kullanılmaktadır. AOS'lerde meydana gelebilecek olan olaylar ikiye ayrılmaktadır. Bunlar; kontrol sinyalleri tarafından *kontrol edilebilen olaylar* ve kontrol sinyalleri

tarafından *kontrol edilemeyen olaylar*dır. AOS'lerde kontrolör tasarımı; verilen bir sistem modeli ve kontrol edilmiş sistemin sağlaması istenen spesifikasyonlar dikkate alınarak, bir gözetici sentezlenmesi işlemidir. Sentezlenecek olan gözeticinin bloklanmasız (nonblocking) olması gerekmektedir. AOS'lerin kontrolünü gösteren blok şema Şekil 2.3'te görülmektedir [1].



Şekil 2.3 AOS'lerin kontrolünü gösteren blok şema

Şekilden de görüldüğü gibi AOS'lerde gözetici, sistemden sensör vb elemanlardan aldığı bilgileri işleyerek arzulanan spesifikasyonları yerine getirmek üzere sisteme kontrol sinyalleri ile müdahale etmektedir. Gözetici içerisinde AOS'nin bir modeli ve spesifikasyonlar bulunmaktadır. AOS, otomata modelleme aracı kullanılarak modellenebilir. Gözetici ise çeşitli yöntemlerle sentezlenmektedir. En yaygın ve genel sentezleme yöntemi Ramadge ve Wonham'ın önerdiği Gözetimli Kontrol Teorisi'nin kullanılmasıdır. Bu yöntem formal diller ve otomata teorilerine dayanmaktadır.

2.3.1 Gözetimli kontrol teorisi

AOS'lerin denetimi için Ramadge ve Wonham'ın (R&W) önermiş olduğu yöntem Gözetimli Kontrol Teorisi olarak bilinmektedir. Bu teori, formal diller ve otomata kavramları üzerine kurulmuştur. Denetlenecek AOS, alfabesi sistemin olaylar kümesi olan bir otomata tarafından üretilmiş bir formal dil jeneratörü ile modellenmektedir. Bu yöntemin özünü, ilgili jeneratörün ürettiği olayların sentezlenen gözetici tarafından etkinleştirilmesi ya da yasaklanması işlemi oluşturur. Arzulanan spesifikasyonlar (specifications) formal dil ile ifade edildikten sonra, gözetici bir dil tanıyıcı (recognizer)

olarak sentezlenebilir. Sentezlenen gözetici, sistemi bazı özel durumlardan veya bazı olay oluşum sıralarından sakındırır. [4,7] referanslarında Ramadge ve Wonham'ın önermiş olduğu yöntemin detayları ve ispatları görülebilir.

Bu kısımda, Ramadge ve Wonham'ın önermiş olduğu yöntem tanıtılacak ve bu yöntem yardımıyla AOS'lerin kontrolü anlatılmaktadır. Formal diller ve otomata teorisi hakkında kısa bilgi verildikten sonra Gözetimli Kontrol Teorisi ve adı geçen yöntemin uygulanmasında kullanılan TCT yazılımı [68] tanıtılacaktır.

2.3.1.1 Formal diller ve otomata

Sonlu sayıda ve birbirinden farklı sembollerden (α, σ, τ, ...) oluşan $\sum$ kümesine *alfabe* denilmektedir. Elemanları bu sembollerin ardı ardına eklenmesi ile oluşturulabilecek sembol sıralamaları ($\sigma_1\sigma_2\sigma_3$, ..., σ_k) olan küme $\sum^+$ ile gösterilmektedir. Hiç bir sembol içermeyen sembol dizisi ise boş dizi (empty string) olarak adlandırılır ve ε ile gösterilir. Boş dizi ve sembol sıralamalarının birleşimi ile oluşturulan $\sum^*=\sum^+\cup\{\varepsilon\}$ kümesinin her bir elemanına $\sum$ alfabesi üzerinde kelime (word) veya dizi (string) denilmektedir.

Sembollerin ardı ardına eklenmesi işlemi katenasyon (catenation) işlemi olarak adlandırılmaktadır. Bu işlemin temel özellikleri aşağıda görülmektedir:

$$cat : \sum^* \times \sum^* \to \sum^* \tag{2.2}$$

$$cat(\varepsilon, s) = cat(s, \varepsilon) = s, \quad s \in \sum^* \tag{2.3}$$

$$cat(s,t) = st, \quad s,t \in \sum^+ \tag{2.4}$$

$\sum$ alfabe kümesinin herhangi bir $\sum^*$ dizi alt kümesine *dil* denilmektedir. Bir alfabe üzerinde tanımlanan bir dil, bu alfabedeki sembollerden oluşan $\sum^*$ dizi alt kümelerini tanınan ve tanınmayan olarak ikiye ayırır. Dil jeneratörünü son duruma ulaştıran diziler tanınan, ulaştırmayan diziler tanınmayan dizilerdir. Bir dil jeneratörü aşağıda görüldüğü gibi bir beşli ile ifade edilmektedir:

$$G = (Q, \sum, \delta, q_0, Q_m) \tag{2.5}$$

Burada;

Q: Durumlar kümesi

$\sum$: Sonlu sayıda giriş simgesinden oluşan giriş alfabe kümesi

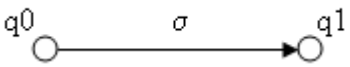
δ: Geçiş fonksiyonu, (Q x $\sum$)'dan Q'ya bir eşleşme

q_0: Başlangıç durumu

Q_m: İşaretli durumlar kümesini (marker) ifade etmektedir.

Q_m durumlar kümesinin bir alt kümesi ($Q_m \subseteq Q$) olarak adlandırılır. Burada $\sum$ alfabe kümesinin sonlu olduğu kabul edilir. Q ve Q_m için böyle bir durum söz konusu değildir. δ geçiş fonksiyonu parçalı fonksiyon olup, $\delta(q,\sigma)$ şeklinde ifade edilmektedir. Bu fonksiyon parçalı fonksiyon olduğu için tüm q ve σ değerleri için tanımlı değildir.

Yukarıdaki beşli ile tanımlanan dil jeneratörü G aynı zamanda, düğüm kümesi (node set) Q ve kenar kümesi (edge set) (q, q') ve $\delta(q,\sigma) = q'$ ikilileri ile tanımlanan bir çizgeye (graph) eşittir. Bu çizge, q ve q' durumları arasındaki etiketi σ olayı olan kenar anlamına gelmektedir. Bu σ olayı sistemi q durumundan q' durumuna götürmektedir. Bu çizge genellikle bir durum makinesi (state machine) veya otomata (özdevinir-automaton) olarak tanımlanır [69]. Basit bir otomata çizgesi Şekil 2.4'te görülmektedir. Bu çizgede içi boş daireler durumları, yönlenmiş ok ise durumlar arası bir geçişi ifade etmektedir.



Şekil 2.4 Basit bir otomata

G dil jeneratörü tarafından oluşturulan dil aşağıda görüldüğü gibi tanımlanmaktadır.

$$L(G)=\{s: s \in \sum^{*} \text{ ve } \delta(s,q_0) \text{ tanımlı ise}\} \quad (2.6)$$

Markalanmış bir dil (marked language) ise aşağıda görüldüğü gibi ifade edilebilir.

$$L_m(G)=\{s: s \in L(G) \text{ ve } \delta(s,q_0) \in Q_m\} \quad (2.7)$$

Burada, G'nin markalanmış dili Lm(G), L(G)'nin alt kümesidir. Markalanmış dilin kelimeleri bir durumu belirten işaretlenmiş ya da kaydedilmiş olayları karakterize etmektedir.

$\sum_1$, $\sum_2$, …, $\sum_n$, birbirinden farklı alfabeler ve her bir alfabe üzerinde tanımlı diller L_1, L_2, …, L_n olmak üzere bu dillerin senkron kompozisyonu (shuffle language) $L_1||L_2||\ldots||L_n$ olarak gösterilir ve aşağıda görüldüğü gibi tanımlanmaktadır. $\sum = \sum_1 \cup \sum_2 \cup \ldots \cup \sum_n$ alfabesi üzerinde tanımlı

$$L_1||L_2||\ldots||L_n = \{s \in \sum^* : s\uparrow_i = s_i \in L_i,\ i=1, \ldots, n\} \quad (2.8)$$

Burada; $s\uparrow_i$'ye *s*'nin $\sum_i$ üzerindeki projeksiyonu denir ve bu işlem $\sum_i$ ait olmayan tüm sembollerden *s*'nin silinmesi olarak tanımlanmaktadır.

Bir öz döngülü dil (self-loop language); $\sum$ üzerinde tanımlı bir dil L ve $\sum_e$ farklı bir dil olmak üzere $L' = L||\sum_e$ olarak tanımlanmaktadır.

2.3.1.2 Ramadge-Wonham yöntemi ile ayrık olay gözeticisi tasarımı

Ramadge-Wonham yöntemi ile ayrık olay sistemlerinin denetimi üç aşamadan oluşmaktadır. Bu aşamalar; sistemin modellenmesi, spesifikasyonların otomata olarak ifade edilmesi ve gözeticinin sentezlenmesidir.

2.3.1.2.1 Sistem modeli oluşturulması

Bir AOS'nin modeli, alt sistemler (subsystems) veya alt işlemler (subprocesses) halinde modellenebilir. Her bir alt sistem, alfabesi farklı dil jeneratörleri (G_1, G_2, ..., G_n) olarak modellenir. Bu jeneratörlerin ürettiği dil, sistemde meydana gelen olayları simgeleyen olay etiketleridir.

Elde edilen G_i jeneratörlerinin her birinin bloklanmasız olması gerekmektedir. Yani G_i'nin tüm durumlarından bir işaretli duruma (marker state) ulaşılabilir olması gerekir. Sisteme ait modelin elde edilebilmesi için alt sistemlerin tamamının senkron kompozisyonu oluşturulur. P sistem modelini veren dil olmak üzere; $P = G_1||G_2||\ldots||G_n$

yazılabilir. Çizge terminolojisi olarak bu durum, senkron kompozisyon yapılan alt sistemlere ait bağımsız durum kümelerinin çarpımı ($Q= Q_1 \times Q_2 \times \ ... \ \times Q_n$) olarak ifade edilmektedir.

Sistem jeneratörünün kontrolü, sistemde bulunan kontrol edilebilir olayların ($\sum_c \subset \sum$) tanımlanarak gözetici tarafından yetkilendirilmesi veya yasaklanması işlemleri ile sağlanmaktadır. Sistemde bulunan diğer olaylar ($\sum \setminus \sum_c$) ise her zaman yetkilendirilmiş durumda olmalıdır.

2.3.1.2.2 Spesifikasyonların modellenmesi

R&W yönteminde, spesifikasyonlar durum makinesi olarak modellenmektedir. Spesifikasyonlar, Sp ile simgelenir ve aşağıda görülen adımlar takip edilerek elde edilebilir:

1. Her bir denetim özelliği için bir H_i jeneratörü ($\sum_{Hi} \subset \sum_1 \cup \sum_2 \cup \ldots \cup \sum_n$) oluşturulmalıdır. Bu jeneratörlerin mutlaka bloklanmasız olması gerekir.
2. Tüm H_i jeneratörleri $\sum_e \equiv i\sum \setminus \sum_{Hi}$ dilindeki sembollerle öz döngülendirilir (self-looped). Bu işlem durum makinesi terminolojisinde, her durumun H_i tarafından kısıtlanmayan ve P sisteminde mümkün olan tüm olay etiketleri ile öz döngülendirilmesi anlamına gelmektedir. Öz döngülendirilmiş H_i jeneratörleri yalnızca kontrol edilemeyen olayları içermektedir.
3. Son olarak elde edilen tüm spesifikasyonlara ait H_i jeneratörlerinin kesişimi hesaplanarak toplam denetim özelikleri elde edilir.

Yukarıdaki adımlar takip edilerek elde edilen toplam Sp spesifikasyon kullanılarak sistemi arzulanan spesifikasyonları yerine getirmeye zorlayacak gözetici tasarlanabilir. Bu adımlar takip edilerek elde edilen Sp ile belirlenen koşulları yerine getirecek geçerli dil (legal language) jeneratörü elde edilebilir. Bu dil $L_g = P \cap S_p$ olarak ifade edilir.

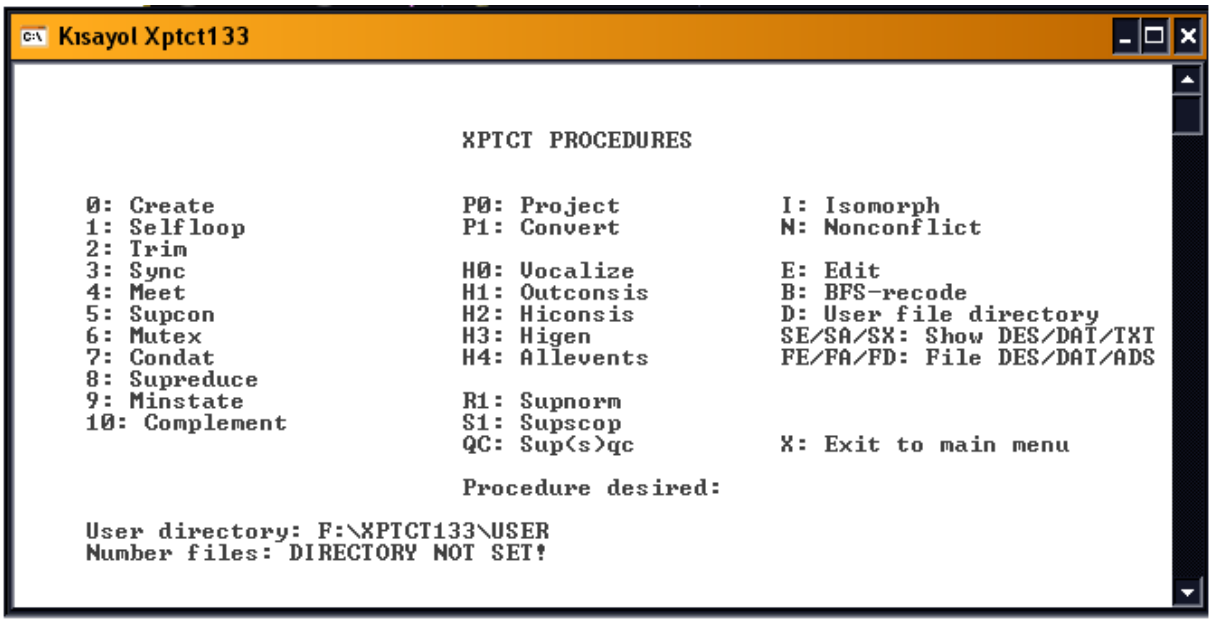
2.3.1.2.3 Gözeticinin sentezlenmesi

SCT, AOS'lerde denetleyici sentezlenmesi için önerilmiş genel bir yöntemdir. SCT yardımıyla, bir AOS'nin kontrol edilmemiş davranışını tanımlayan *plant* (*P*) (makine ya da kısaca sistem) ve kontrol edilmiş davranışını ifade eden *spesifikasyon*un (*Sp*) verilmesiyle, plant'ı istenilen spesifikasyona uygun bir şekilde kontrol etmek için bir *supervisor* (*S*) (kontrolör - gözetici) otomatik olarak sentezlenebilir. Plant, belirli bir harf seti kapsamında rastgele bir şekilde bir takım olaylar üreten bir üretici olarak düşünülür. Bu olaylar kontrol edilebilir ya da kontrol edilemez olaylar olabilir. Supervisor, plant tarafından üretilen bu olaylar dizisini gözlemler ve plant'ı kontrol edilebilir bir grup olayları üretmekten men edebilir. Bununla beraber supervisor, plant'ı yeni bir olay üretmeye zorlayamaz. Buna göre plant ve supervisor *P||S* kompozisyonu spesifikasyon *Sp*'yi sağlamak zorundadır.

SCT ile sentezlenen gözetici $C=(S, \gamma)$ ikilisinden oluşmaktadır. Burada S istenilen spesifikasyonları kontrol eden bir tanıyıcı (recognizer) ve γ ise $\gamma:\Sigma \rightarrow \{0, 1\}$ şeklinde tanımlanan kontrol çiftleridir. Kontrol edilebilir olaylar kümesine ait bir σ olayı, γ tarafından yasaklanabilir veya yetkilendirilebilir. Eğer $\gamma(\sigma)=0$ ise σ yasaklanır, $\gamma(\sigma)=1$ ise σ yetkilendirilir. Kontrol edilemeyen $\sigma \in (\sum \setminus \sum_c)$ olayları için her zaman $\gamma(\sigma)=1$ dir. Yani sistemde bulunan kontrol edilemeyen olaylar her zaman yetkilendirilmiş durumdadır. Gözetici modelinde, aynı olayların yasaklanmış olduğu durumlar birleştirilerek daha az sayıda duruma sahip bir model elde edilebilmektedir. Elde edilen bu model indirgenmiş gözetici modeli olarak adlandırılmaktadır.

2.3.1.3 TCT yazılımı

AOS'ler için gözetici hesabında gerekli olan işlemleri yapmak üzere Prof. Dr. W. M. Wonham tarafından http://www.control.utoronto.ca/DES/ adlı internet sitesinde TCT isimli bir bilgisayar yazılımı araştırmacılara ücretsiz olarak sunulmaktadır [68]. Bu yazılımın ekran görüntüsü Şekil 2.5'te verilmiştir. Bu kısımda ilgili yazılım kullanılarak yapılabilen temel bazı işlemler kısaca tanıtılacaktır.



Şekil 2.5 Ramadge-Wonham yöntemi (Supervisory Control Theory - SCT) için geliştirilmiş olan TCT yazılımının görünümü

TCT yazılımında olayları ve durumları simgeleyen etiketler sayılardan oluşmaktadır. Bu yazılımın kullanımı için ilk olarak ilgili sistemler jeneratör veya tanıyıcı olarak sisteme aşağıdaki beşli ifade gibi girilmelidir:

[Size, Init, Mark, Voc, Tran]

Burada; *Size* sistemdeki durum sayısını, *Init* başlangıç durumunu (başlangıç durumu her zaman 0 etiketli durumdur), *Mark* işaretli durumları, *Voc* çıkış verebilen durumları, *Trans* ise durumlar arası geçiş bilgisini göstermektedir. TCT yazılımı içerisinde formal diller üzerinde tanımlı tüm işlemler yapılabilmektedir. Bu işlemler ve TCT yazılımı kullanılarak bir AOS gözeticisinin tasarımında kullanılan temel komutlar aşağıda görülmektedir:

*DES=**Create**(DES):* Yeni bir tanıyıcı (dil üretici) oluşturmak için kullanılmaktadır. Bu komutu kullanmak için Şekil 2.5'te görülen ekrandayken 0 tuşuna basılır ve yazılımın yönlendirdiği sırayla AOS'ye verilecek isim (DES), başlangıç durumu (init) 0 numaralı durum olan bu sistemdeki toplam durum sayısı (size), sistemin amaca ulaştığını belirten işaretli durumlar (mark) (bu son durumlar kümesi toplam durumlar kümesinin alt kümesidir), çıkış üreten durumlar (Voc) ve son olarak sistemde gerçekleşebilecek bütün geçişler (tran); mevcut durum, geçişe neden olacak olay ve geçişin yapılacağı durum belirtilerek girilir.

*DES2=**Selfloop**(DES1, olay listesi):* Olay listesinde verilen olay etiketleri ile DES1 isimli sistemin (dil jeneratörü) tüm durumları öz-döngülendirilir. Bir olayı öz-döngülendirmek, bu olay oluşumuyla ilgili durumdan yine aynı duruma geçişin gerçekleşmesi anlamına gelmektedir. Bu komutun TCT yazılımında kullanımı Şekil 2.5'teki ekranda 1 tuşuna basılarak yapılır. Yazılım öz-döngü işleminin yapılacağı sistemin ismini (DES1) sorar ve öz-döngülendirilmiş sisteme verilecek bir isim (DES2) girilmesini ister. Bundan sonraki adımda sistemin bütün durumlarının öz-döngülendirileceği olay etiketleri (olay listesi) girilir.

*DES3=**Sync**(DES1,DES2):* DES1 ve DES2 olarak tanımlanan sistemlerin senkron kompozisyonu işlemini yapar. TCT yazılımında 3 tuşuna basılarak işlem başlatılır. Senkron kompozisyon yapılacak sistemlerin isimleri sırayla (DES1 ve DES2) girilir ve elde edilecek sisteme bir isim (DES3) verilir. İşlem sonunda istenilen isimde iki sistemin senkron kompozisyonunu veren sistem elde edilir. Bu komutla sadece iki sistemin senkron kompozisyonu yapılabilir. Daha çok sistem birleştirilecekse ikişerli olarak sıra gözetmeksizin yapılmalıdır. Sync komutuyla elde edilen birleşmiş sistemin (DES3) tanımlı olduğu alfabe ($\sum_{DES3}$) işlemde kullanılan sistemlerin (DES1 ve DES2) alfabeleri ($\sum_{DES1}$ ve $\sum_{DES2}$) birleşimidir ($\sum_{DES3} = \sum_{DES1} \cup \sum_{DES2}$).

*DES3=**Meet**(DES1,DES2):* DES1 ve DES2 olarak isimlendirilen sistemlerin birleştirilmesi işlemini yapar. TCT yazılımında 4 tuşuna basılarak işlem başlatılır. Birleştirilecek sistemlerin isimleri sırayla (DES1 ve DES2) girilir ve elde edilecek sisteme bir isim (DES3) verilir. İşlem sonunda istenilen isimde iki sistemin birleşimini veren sistem elde edilir. Bu komutla sadece iki sistemin birleşimi yapılabilir. Daha çok sistem birleştirilecekse ikişerli olarak sıra gözetmeksizin yapılmalıdır. Meet komutuyla elde edilen birleşmiş sistemin (DES3) tanımlı olduğu alfabe ($\sum_{DES3}$) işlemde kullanılan (DES1 ve DES2) alfabeleri ($\sum_{DES1}$ ve $\sum_{DES2}$) kesişimidir ($\sum_{DES3} = \sum_{DES1} \cap \sum_{DES2}$).

*DES3=**Supcon**(DES1,DES2):* DES1 olarak tanımlanmış AOS ve bu sistem için DES2 olarak tanımlanan spesifikasyonlar dikkate alınarak DES3 olarak tanımlanan bir gözetici hesaplanır. TCT yazılımında Şekil 2.5'teki menüdeyken 5 tuşuna basılarak *Supcon(.)* komutu çalıştırılır. Açılan pencerede kontrol edilmemiş sistem modeli ve sistemin çalışması istenilen kontrol özelliklerini ifade eden spesifikasyon modeli sırayla

girilir. Elde edilecek gözeticiye isim verildikten sonra yazılım otomatik olarak gözetici modelini oluşturacaktır.

*DAT=**Condat**(DES1,DES2):* DES1 ile tanımlanan AOS ve bu sistem için *DES2=Supcon(.)* ile hesaplanan gözetici kullanılarak DAT isminde kontrol verisi oluşturulur. Bu DAT verisi gözetici içerisindeki durumların hangilerinde, sistemin hangi olaylarının yasaklanacağını göstermektedir. TCT yazılımının ana menüsünde 7 tuşuna basılarak açılan pencerede AOS modeli (DES1) ve supcon komutuyla elde edilen sistemin kontrol edilmiş modeli (DES2) girilir. Bu işlemlerin devamında oluşturulacak kontrol bilgisine DAT ismi verilir.

*DES3=**Supreduce**(DES1,DES2,DAT):* Bu işlem yardımıyla *Supcon(.)* komutuyla elde edilen gözeticinin durum sayısı azaltılmaktadır. Bu küçültülmüş modele indirgenmiş gözetici denilmektedir. Bu komut, Şekil 2.5'teki menüdeyken 8 tuşuna basılarak açılan pencerede sırasıyla AOS'yi ifade eden modelin (DES1), *DES2=Supcon(.)* komutuyla elde edilen kontrol edilmiş modelin (DES2) ve *DAT=Condat(.)* komutuyla elde edilen kontrol verisinin (DAT) girilmesiyle indirgenmiş gözetici modelini (DES3) oluşturur.

2.3.1.4 TCT yazılımı kullanılarak gözetici tasarımı

Bu çalışmada, Ramadge - Wonham yöntemi yardımıyla ayrık olay sistemlerinin kontrolünü gerçekleştirecek gözetici modelinin elde edilmesi için TCT yazılımı kullanılmaktadır. Gözetici otomata modelinin elde edilmesi için öncelikle sistemi ifade eden kontrol edilmemiş modelin (PLANT) oluşturulması gerekmektedir. Daha sonra sistemin çalışmasında istenilen özellikler (spesifikasyonlar - SPEC) otomata modeline dönüştürülmelidir. Gözetici modeli ve indirgenmiş gözetici modeli oluşturulması için gerçekleştirilmesi gereken işlem sırası aşağıda ifade edilmiştir:

1. AOS'lerin Gözetimli Kontrol Teorisi yardımıyla kontrolünde yapılacak olan ilk işlem sistemi oluşturan tüm alt bileşenlerin (G1, G2, ..., GN) otomata olarak oluşturulması ve bu otomata modellerinin TCT yazılımında *Create(.)* komutu kullanılarak gerçekleştirilmesi işlemidir:

   G1=Create(G1)
   G2=Create(G2)

.
.
GN=Create(GN)

2. Elde edilen alt bileşenlerin senkron kompozisyonu ile sistemin kontrol edilmemiş modeli oluşturulur (PLANT=G1||G2||…||GN). Bu işlem TCT yazılımında *Sync(.)* komutu kullanılarak gerçekleştirilir:

   PLANT=Sync(G1,G2)
   PLANT=Sync(PLANT,G3)
   .
   .
   PLANT=Sync(PLANT,GN)

3. Gözetici tasarımında yapılacak diğer işlemde, sistemin çalışmasında istenilen özelliklerin her biri *Create(.)* komutuyla ayrı ayrı elde edilir (SPEC1, SPEC2, ..., SPECM):

   SPEC1=Create(SPEC1)
   SPEC2=Create(SPEC2)
   .
   .
   SPECM=Create(SPECM)

   Daha sonra elde edilen her bir spesifikasyon modeline kontrol edilmemiş sistem modeli PLANT'ta bulunan fakat ilgili spesifikasyon tarafından kısıtlanmayan olaylar ilgili spesifikasyon modelinin tüm durumlarına *Selfloop(.)* komutu kullanılarak özdöngü olarak eklenir:

   SPEC1=Selfloop(SPEC1, PLANT'ta bulunan diğer olaylar)
   SPEC2=Selfloop(SPEC2, PLANT'ta bulunan diğer olaylar)
   .
   .
   SPECM=Selfloop(SPECM, PLANT'ta bulunan diğer olaylar)

   Son olarak elde edilen tüm spesifikasyon modelleri (SPEC1, SPEC2, ..., SPECM) TCT yazılımında bulunan *Meet(.)* komutu kullanılarak sistemin çalışma şeklini ifade eden tek bir spesifikasyon (SPEC) modeli olarak elde edilir:

   SPEC=Meet(SPEC1, SPEC2)
   SPEC=Meet(SPEC, SPEC3)
   .
   .
   SPEC=Meet(SPEC, SPECM)

4. *SUP=(PLANT, SPEC)* komutu kullanılarak sistemin kontrol edilmiş şeklini ifade eden gözetici modeli SUP elde edilir.

5. Bir önceki adımda elde edilen SUP gözetici modeline göre daha az duruma sahip olan indirgenmiş gözetici modelinin TCT yazılımıyla elde edilmesi için *DAT=Condat(PLANT, SUP)* komutu kullanılarak DAT kontrol bilgisi elde edilir. Daha sonra *SIMSUP=Supreduce(PLANT,SUP,DAT)* komutu yardımıyla indirgenmiş gözetici modeli SIMSUP oluşturulur.

2.4 Uygulama

Bu kısımda Ramadge ve Wonham'ın önermiş olduğu yöntem kullanılarak örnek bir ayrık olay sistemi için gözeticinin nasıl oluşturulduğu anlatılmaktadır. Bu çalışmada kullanılan örnek ayrık olay sistemi literatürde sıkça kullanılan iki makine ve bir tampondan oluşan basit bir imalat sistemidir [37, 70, 71]. Bu sistemde Makine 1 ve Makine 2, kapasitesi 1(bir) olan tampon ile Şekil 2.6'da görüldüğü gibi birbirine bağlanmıştır. Sistemdeki her bir makinenin bir işlem yapmadığı durum *BOŞTA*, ilgili parçayı işlediği durum *ÇALIŞIYOR* ve arıza oluştuğu durum *ARIZALI* olmak üzere üç durumu mevcuttur. Tamponun ise dolu olduğu durumu belirten *DOLU* ve boş olduğu durumu belirten *BOŞ* olmak üzere iki durumu mevcuttur. Başlangıçta tamponun içerisinde bir parça olmadığı ve makinelerin BOŞTA olduğu kabul edilmektedir.

Şekil 2.6 Basit imalat sistemi

Bu imalat sisteminde bir ürünün üretimi şu şekilde gerçekleştirilmektedir. İşlenecek parça Makine 1 tarafından girişten alınmaktadır. Bu örnekte girişte sonsuz kapasiteli bir tampon olduğu ve giriş parçalarının buradan alındığı kabul edilmektedir. Makine 1 girişten aldığı parça üzerindeki işini (örneğin parçayı delmesi, kesmesi, vb) bitirmesi ile parça tampona geçmektedir. Makine 1 ÇALIŞIYOR durumundayken arıza meydana gelirse ARIZALI durumuna geçer. ARIZALI durumundayken tamir edilmesi halinde ise Makine 1 BOŞTA durumuna getirilmektedir. Tamponda bulunan parça Makine 2 tarafından alınarak üzerinde ilgili işlemler gerçekleştirilir ve çıktı ürün oluşturulur.

Makine 1'e benzer olarak Makine 2 ÇALIŞIYOR durumundayken arıza meydana gelirse ARIZALI durumuna geçer. ARIZALI durumundayken tamir edilmesi halinde ise Makine 2 BOŞTA durumuna getirilmektedir.

2.4.1 Makinelerin otomata modellerinin oluşturulması

R&W yöntemi ile gözetici tasarımı için ilk olarak sistemde bulunan makinelerin otomata modellerinin oluşturulması gerekmektedir. Bu amaçla ilk olarak bir makine çalışırken meydana gelebilecek olaylar belirlenmeli ve bu olaylar etiketlenmelidir. Makinenin başlangıçta boşta olduğu varsayılırsa girişten bir parça alarak işlemeye başlaması ilk olay olup *si*(i=1,2) olarak etiketlenebilir. Bu olay ile makine boşta durumundan çalışıyor durumuna geçmektedir. Makinenin aldığı parçayı işlemeyi bitirerek parçayı tampona aktarması olayı ise *fi* (i=1,2) ile etiketlenmiştir. Bu olayın oluşumu ile makine çalışıyor durumundan boşta durumuna geri dönmektedir. Makine çalışıyorken bir arıza oluşumu olayı *bi*(i=1,2) ile etiketlenmiştir. Bu olay ile makine çalışıyor durumundan arızalı durumuna geçer. Makine arızalı iken tamir edilmesi olayı ise *ri*(i=1,2) ile etiketlenir ve bu olayın oluşumu ile makine arızalı durumundan boşta durumuna geçer. Bir makinenin olayları ve anlamları Çizelge 2.1'de görülmektedir. Benzer şekilde bir makinedeki durumlar ve anlamları da Çizelge 2.2'de görülmektedir.

Çizelge 2.1 Bir makinedeki olaylar ve anlamları

Olay Etiketi	Açıklama
si	Makineyi çalıştır
fi	Makinenin çalışması bitti
bi	Makine arızalandı
ri	Makine tamir edildi

Çizelge 2.2 Bir makinedeki durumlar ve anlamları

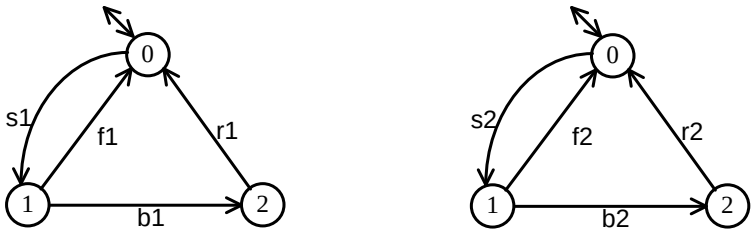
Durum	Açıklama
0	Makine çalışmıyor (boşta)
1	Makine çalışıyor
2	Makine arızalı

Bir makinede oluşabilecek olaylardan, *si* (çalışmaya başlama) ve *ri* (tamir edilme) kontrol edilebilir olaylardır. Bunlar dışındaki olaylar ise kontrol edilemeyen olaylardır. Ramadge-Wonham gözeticisi hesaplamak için kullanılan TCT yazılımında kullanılacak

olay etiketleri sayılardan oluşmaktadır. Bu etiketler verilirken dikkat edilmesi gereken nokta, kontrol edilebilir olaylara tek sayıların (1, 3, 5, 7,..) kontrol edilemeyen olaylara ise çift sayıların (2, 4, 6, 8, ..) verilmesidir. Bu özellik dikkate alınarak örnek imalat sistemi için TCT yazılımında kullanılan etiketler aşağıda görülmektedir.

Otomata etiketleri:	f1	s1	b1	r1	f2	s2	b2	r2
TCT karşılıkları:	10	11	12	13	20	21	22	23

Bu sisteme ait her bir makinenin otomata modeli Şekil 2.7'de görülmektedir. Otomata modelinde içi boş daireler sistemin durumlarını yönlenmiş oklar ise durumlar arasındaki geçişleri göstermektedir. İki yönlü ok ise hem başlangıç durumunu hem de "marker" durumunu göstermektedir.



a) b)

Şekil 2.7 a) Makine 1'in otomata modeli b) Makine 2'nin otomata modeli

Şekil 2.7.(a) ve Şekil 2.7.(b)'de sırası ile Makine 1 ve Makine 2'nin otomata modeli görülmektedir. Bu otomata modellerinde durumlara ve durumlar arası geçişlere sözel etiketler verilmiştir. Bu etiketler TCT yazılımı kullanılırken sayılara dönüştürülmektedir. Şekil 2.7'de kullanılan sözel etiketlerin TCT yazılımında kullanılan sayısal karşılıkları yukarıda verilmiştir. Bu sayısal karşılıklar dikkat alınarak Makine 1 ve Makine 2 otomatalarının TCT yazılımında oluşturulması için aşağıdaki işlemler yapılır.

MAK1 = Create(MAK1,[mark 0],[tran [0,11,1],[1,10,0],[1,12,2],[2,13,0]]) (3,4)
MAK2 = Create(MAK2,[mark 0],[tran [0,21,1],[1,20,0],[1,22,2],[2,23,0]]) (3,4)

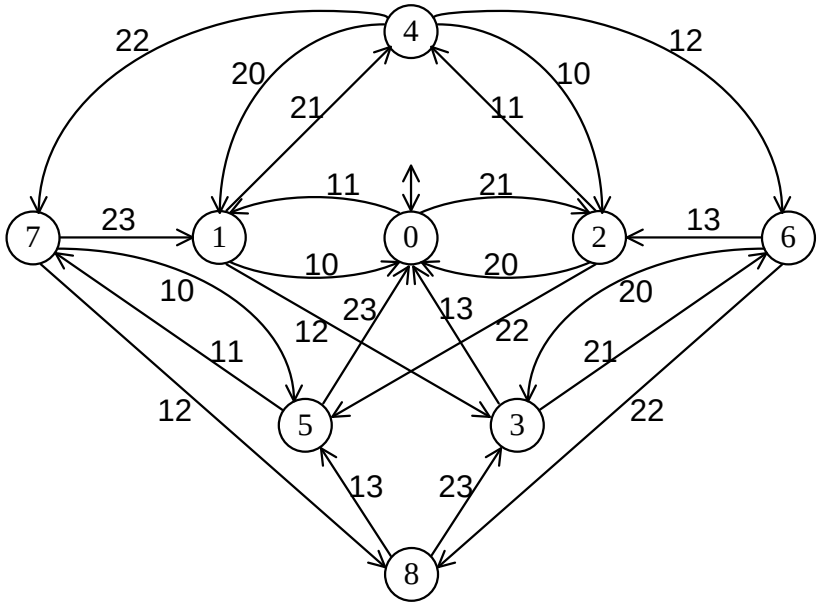
2.4.2 Sistemin kontrol edilmemiş davranışını ifade eden modelin (PLANT) elde edilmesi

Şekil 2.7'de görülen Makine 1 ve Makine 2 otomatalarının senkron kompozisyonu sistemin kontrol edilmemiş davranışını ifade eden otomata modelini verir. Bu işlem TCT yazılımında *Sync(.)* komutu kullanılarak aşağıdaki gibi gerçekleştirilir:

PLANT = Sync(MAK1,MAK2) (9,24) Blocked_events = None

PLANT # states: 9 state set: 0 ... 8 initial state: 0 marker states: 0
vocal states: none
 # transitions: 30

transitions:
[0, 11, 1] [0, 21, 2] [1, 10, 0] [1, 12, 3] [1, 14, 1] [1, 21, 4] [2, 11, 4] [2, 20, 0] [2, 22, 5] [2, 24, 2] [3, 13, 0] [3, 21, 6] [4, 10, 2] [4, 12, 6] [4, 14, 4] [4, 20, 1] [4, 22, 7] [4, 24, 4] [5, 11, 7] [5, 23, 0] [6, 13, 2] [6, 20, 3] [6, 22, 8] [6, 24, 6] [7, 0, 5] [7, 12, 8] [7, 14, 7] [7, 23, 1] [8, 13, 5] [8, 23, 3]



Şekil 2.8 Sistemin kontrol edilmemiş davranışını ifade eden PLANT otomata modeli

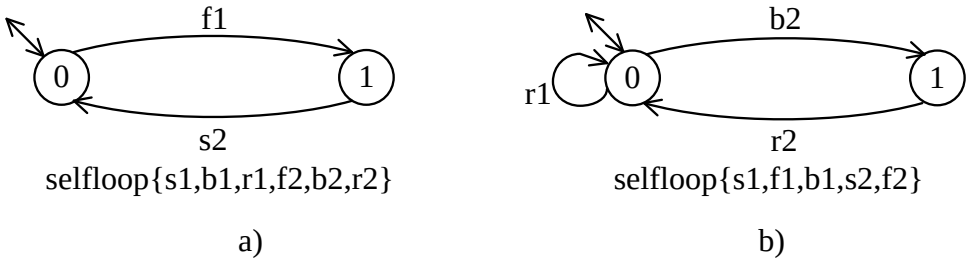
Sonuçta, sistemin kontrol edilmeden önceki halini ifade eden (Şekil 2.8'de görülen) 9 durum ve 24 geçişli olan PLANT adında bir otomata modeli elde edilir.

2.4.3 Spesifikasyonlar ve otomata modelleri

Bu üretim sisteminin çalışma şeklini ifade eden spesifikasyonlar aşağıdaki gibidir:

1. Sistemde bulunan tampon ne taşacak ne de tamamen boşalacak. Yani tampon dolu iken Makine 1, tampon boş iken de Makine 2 çalışmayacaktır.
2. Her iki makinenin bozulması durumunda ilk önce Makine 2 tamir edilecektir.

Bu spesifikasyonlardan ilkine ait otomata modeli Şekil 2.9.(a)'da görülmektedir. Bu modele göre her zaman *f1* (Makine1'in yapacağı işlemi bitirmesi) *s2*'den (Makine 2'nin tampondan parça almasından) önce oluşmaktadır. Ayrıca bu modele göre, Makine 1 çalışmasını bir kez bitirdikten (*f1*) sonra Makine 1'in çalışmasını bir kez daha bitirebilmesi için öncelikle Makine 2'nin çalışmaya başlaması (*s2*) gerekmektedir.



Şekil 2.9 a) Birinci spesifikasyona (BUFSPEC) ait otomata modeli
b) İkinci spesifikasyona (BRSPEC) ait otomata modeli

Şekil 2.9.(b)'de ikinci spesifikasyona ait otomata modeli görülmektedir. Bu modele göre Makine 1'in tamir edilebilmesi için (r1) Makine 2'nin arızalanmamış olması (b2) gerekir. Eğer, Makine 2 arızalanmışsa (b2), bu durumda Makine 1'in tamir edilebilmesi (r1) için öncelikli olarak Makine 2'in tamir edilmesi (r2) gerekir.

Spesifikasyonları gerçekleştirmek üzere tasarlanan her bir otomata modelindeki her duruma ilgili modelde bulunmayan diğer olaylar (etiketler) öz-döngü olarak eklenir. Bu işlem modellerin yanındaki Selfloop{.} ifadesi ile belirtilmiştir. Buna göre ilk spesifikasyona (BUFSPEC) ve ikinci spesifikasyona ait (BRSPEC) otomata modelleri TCT'de aşağıda görüldüğü gibi *Create(.)* ve *Selfloop(.)* komutları kullanılarak elde edilmiştir.

BUFSPEC = Create(BUFSPEC,[mark 0],[tran [0,10,1],[1,21,0]]) (2,2)

BUFSPEC = Selfloop(BUFSPEC,[11,12,13,20,22,23]) (2,14)

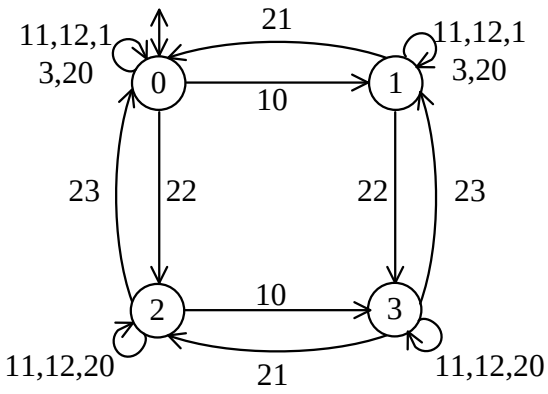
BRSPEC = Create(BRSPEC,[mark 0],[tran [0,13,0],[0,22,1],[1,23,0]]) (2,3)
BRSPEC = Selfloop(BRSPEC,[10,11,12,20,21]) (2,13)

Sisteme ait tek bir spesifikasyon elde etmek için *Meet(.)* komutu aşağıda görüldüğü gibi kullanılarak SPEC otomata modeli elde edilir. Elde edilen SPEC otomata modeli Şekil 2.10'da görülmektedir.

SPEC = Meet(BUFSPEC,BRSPEC) (4,22)

SPEC # states: 4 state set: 0 ... 3 initial state: 0 marker states: 0 vocal states: none
 # transitions: 30

transitions:
[0, 10, 1] [0, 11, 0] [0, 12, 0] [0, 13, 0] [0, 14, 0] [0, 20, 0] [0, 22, 2] [0, 24, 0] [1, 11, 1] [1, 12, 1] [1, 13, 1] [1, 14, 1] [1, 20, 1] [1, 21, 0] [1, 22, 3] [1, 24, 1] [2, 10, 3] [2, 11, 2] [2, 12, 2] [2, 14, 2] [2, 20, 2] [2, 23, 0] [2, 24, 2] [3, 11, 3][3, 12, 3] [3, 14, 3] [3, 20, 3] [3, 21, 2] [3, 23, 1] [3, 24, 3]



Şekil 2.10 SPEC otomata modeli

2.4.4 RW gözeticisi elde edilmesi

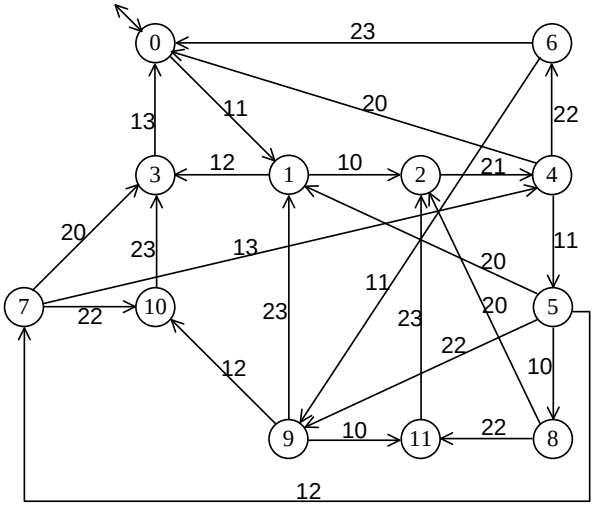
Gözeticinin TCT yazılımı ile tasarlanabilmesi için gerekli olan sistem modeli PLANT ve sepsifikasyonları ifade eden SPEC otomata modelleri önceki işlem sıralarında elde edilmiştir. Buna göre *Supcon(.)* komutu kullanılarak RW gözeticisi (SUP) aşağıdaki gibi elde edilir:

SUP = Supcon(PLANT,SPEC) (12,24)

Bu işlem sonucu TCT tarafından hesaplanan gözetici SUP aşağıda görülmektedir:

SUP # states: 12 state set: 0 ... 11 initial state: 0
marker states:0 vocal states: none # transitions: 24
transitions:
[0, 11, 1] [1, 10, 2] [1, 12, 3] [2, 21, 4] [3, 13, 0] [4, 11, 5] [4, 20, 0] [4, 22, 6]
[5, 10, 8] [5, 12, 7] [5, 20, 1] [5, 22, 9] [6, 11, 9] [6, 23, 0] [7, 13, 4] [7, 20, 3]
[7, 22, 10] [8, 20, 2] [8, 22, 11] [9, 10, 11] [9, 12, 10] [9, 23, 1] [10, 23, 3] [11, 23, 2]

Bu gözeticinin otomata modeli Şekil 2.11'de verilmiştir. Bu şekilde daireler içerisine yazılan sayılar TCT'nin ürettiği gözetici durumlarını, oklar üzerindeki sayılar ise gözetici olaylarını göstermektedir.



Şekil 2.11 Elde edilen RW gözeticisinin (SUP) otomata modeli

2.4.5 İndirgenmiş gözeticinin elde edilmesi

Gözetici modelinde aynı olayların yasaklandığı durumların birleştirilmesi sonucu daha az sayıda duruma sahip indirgenmiş gözetici modeli elde edilebilmektedir. Bu modelin elde edilmesi öncelikle yasaklanan olayların listesini veren kontrol verisini gerektirmektedir. Bunun için TCT yazılımında yapılması gereken işlem aşağıda görülmektedir.

DAT = Condat(PLANT,SUP) Controllable.

Gözetici modelinin ilgili durumlarında yasaklanan olaylar aşağıdaki gibi elde edilir.

DAT
Control data are displayed as a list of supervisor states where disabling occurs, together with the events that must be disabled there.
control data:
0: 21 1: 21 2: 11 3: 21
8: 11 10: 13 11: 11

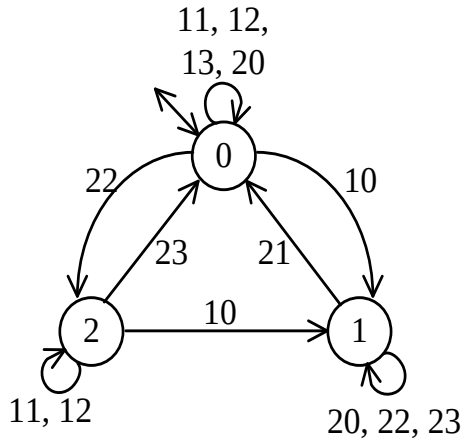
Elde edilen kontrol verisi, PLANT otomata modeli ve SUP gözetici otomata modeli TCT yazılımı kullanılarak aşağıdaki işleme tabi tutulur:

SIMSUP = Supreduce(PLANT,SUP,DAT) (3,14;slb=3)

Bu işlem sonucunda SIMSUP adında indirgenmiş bir gözetici elde edilir. TCT'de oluşturulan indirgenmiş gözetici modeli SIMSUP aşağıda görüldüğü gibidir.

SIMSUP # states: 3 state set: 0 ... 2 initial state: 0
marker states: 0 vocal states: none # transitions: 14
transitions:
[0, 10, 1] [0, 11, 0] [0, 12, 0] [0,13, 0] [0, 20, 0] [0, 22, 2] [1, 20, 1] [1, 21, 0] [1, 22, 1] [1, 23, 1][2, 10, 1] [2, 11, 2] [2, 12, 2] [2, 23, 0]

Modelin otomata olarak gösterimi Şekil 2.12'de verilmiştir.



Şekil 2.12 İndirgenmiş gözetici (SIMSUP) otomata modeli

2.5 Petri Ağları

Petri ağları (Petri Nets – PN), AOS'lerin kontrolü, tasarımı ve analizinde yaygın olarak kullanılan formal bir araçtır. PN'ler, otomata ile haberleşme çalışmalarında ağ benzeri matematiksel bir araç öneren Alman matematikçi ve bilgisayar bilimcisi Carl A. Petri'den sonra bu ismi almıştır [16]. PN temelleri için [72]'ye bakılabilir. Ağırlıkları

oklar ile ilişkilendirilen PN'lere ağırlıklı oka sahip PN denilmektedir. Ok haritalamalarında negatif olmayan tam sayılar kullanılır. Bu durumda, her ok bir w çarpanına sahiptir. Bu çarpan okun ağırlığı olarak adlandırılır. Sıradan PN'lerde ok ağırlıkları 1'dir. Standart PN'lerin aksine, yasaklama okuna sahip PN'lerde bir mevkide jeton bulunmama durumu test edilebilmektedir (sıfır test edilebilirliği). Yasaklama okları yaygın olarak bilinen bir standart PN genişletmesidir. PN modellerine eklenen yasaklama oku ile PN'lerin modelleme gücü Turing makineleri kadar güçlü hale gelmiştir [73]. Bir mevkide bulunan jeton sayısının, kesin bir eşik sayısından daha az olduğu ağırlıklı yasaklama oku kullanılarak test edilebilir [73]. Yasaklama okunu tamamlamak üzere, yetkileme oku Uzam ve Jones tarafından önerilmiştir [60-61]. Yetkileme oku ile PN modelindeki bir mevkinin içerisinde jeton bulunma durumu test edilebilmektedir (1 test edilebilirliği). Yetkileme okunun ağırlıklı olduğu durumda bir mevkide bulunan jeton sayısının belirli bir eşik değerden fazla olma durumu test edilebilir.

Ağırlıklı giriş, çıkış, yetkileme ve yasaklama okuna sahip bir PN aşağıdaki gibi tanımlanabilir.

$$PN=(P, T, Pre, Post, In, En, M_0) \quad (2.9)$$

Burada;

- $P = \{p_1, p_2, ..., p_n\}$ boş olmayan sonlu mevkiler kümesi;
- $T = \{t_1, t_2, ..., t_m\}$ boş olmayan sonlu geçişler kümesi, $P \cup T \neq \varnothing$ ve $P \cap T = \varnothing$;
- Pre: $(P \times T) \rightarrow N$ mevkilerden geçişlere yönlenmiş ağırlıklı okları tanımlayan giriş fonksiyonudur. N negatif olmayan tam sayılar kümesidir.
- Post: $(T \times P) \rightarrow N$ geçişlerden mevkilere yönlenmiş ağırlıklı okları tanımlayan çıkış fonksiyonudur.
- In: $(P \times T) \rightarrow N$ mevkilerden geçişlere yönlenmiş ağırlıklı yasaklama oklarını tanımlayan yasaklama fonksiyonudur.
- En: $(P \times T) \rightarrow N$ mevkilerden geçişlere yönlenmiş ağırlıklı yetkileme oklarını tanımlayan yetkileme fonksiyonudur.
- $M_0 : P \rightarrow N$ başlangıç işaretlemesidir.

Ağırlıklı giriş, çıkış, yasaklama ve yetkileme oku içeren PN'ler iki tip düğümden oluşur. Bunlardan ilki daire (○) ile gösterilen mevkiler diğeri ise çizgi (⟶) ile gösterilen geçişlerdir. Bu tür PN'lerde üç farklı ok yapısı kullanılmaktadır. Bu oklar, yönlenmiş ok (⟶) ile gösterilen sıradan (giriş ve çıkış) ok, sonu daire (⟶○) ile gösterilen yasaklama oku ve sonu içi boş ok (⟶▷) ile gösterilen yetkileme okudur. Ağırlıklı sıradan oklar mevkileri geçişlere veya geçişleri mevkilere bağlarken, yetkileme ve yasaklama okları yalnızca geçişleri mevkilere bağlarken kullanılırlar. Mevkiler içerisinde bulunan jetonların dağılımı sistemin güncel durumunu, geçişler ise olayları göstermektedir. Her bir geçiş, geçişin ön koşullarını (pre-conditions) ve son koşullarını (post-conditions) gösteren giriş ve çıkış mevkileri kümesine sahiptir. PN'nin işaretlemesi mevkilerde bulunan jeton dağılımları ile belirlenir. Jetonlar siyah noktalar (•) ile gösterilir. Jetonların mevkiler arasındaki dolaşımı "yetkileme kuralları" ve "tetikleme kuralları" ile belirlenir. Yetkileme kuralları, bir geçişin hangi koşullar altında tetiklenebileceğini tanımlar. Tetikleme kuralları ise bir geçişin tetiklenmesi ile jeton dağılımının nasıl olacağını tanımlar. Yetkileme ve tetikleme kurallarının her ikisi de oklar tarafından belirlenir. Yetkileme kuralları giriş, yetkileme ve yasaklama oklarının tamamı ile ilgiliyken tetikleme kuralları sadece giriş ve çıkış okları ile ilişkilidir. Giriş okları hem yetkileme koşullarında hem de tetikleme kurallarında etkilidir. PN'lerde jetonların ağdaki dolaşımı aşağıdaki yetkileme ve tetikleme kuralları ile yerine getirilir.

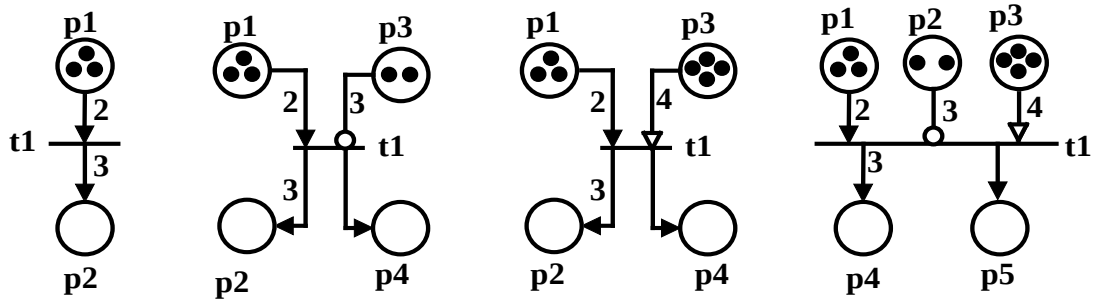
Yetkileme kuralları: Ağırlıklı giriş, çıkış, yetkileme ve yasaklama oku içeren PN modellerinde bir geçişin tetiklenmesine izin veren üç kural vardır. Bir geçişin tetiklenebilmesi için aşağıdaki ön koşulları sağlaması gerekir.

1. Eğer, bir *p* giriş mevkisi bir *t* geçişine ağırlıklı ok *Pre(p,t)* ile bağlanmış ise, *p* mevkisi, en az giriş oku ağırlığına eşit sayıda jeton içeriyorsa yani M(*p*) ≥ *Pre(p,t*) ise *t* geçişi yetkilendirilmiştir denir. Şekil 2.13.(a)'da görülen PN modelinde, p1 mevkisinde 2 veya daha fazla jeton bulunduğu durumda t1 geçişi yetkilenir. p1 mevkisinde 3 jeton bulunduğundan yani M(*p1*) = 3 ve Pre(p1,t1) = 2 olduğundan t1 geçişi yetkilendirilmiştir.
2. Eğer, bir *p* giriş mevkisi bir *t* geçişine ağırlıklı yasaklama oku In*(p,t)* ile bağlanmış ise, *p* mevkisi, yasaklama oku ağırlığından az sayıda jeton içeriyorsa yani M(*p*) < *In(p,t)* ise *t* geçişi yetkilendirilmiştir denir. Bu durum Şekil 2.13.(b)'de görülmektedir. Bu modelde görülen t1 geçişinin yetkilendirilebilmesi için p1 mevkisinde 2 veya daha fazla jeton, p3 mevkisinde

ise 3'ten az sayıda jeton bulunması gerekmektedir. Bu modelde, M(*p1*) = 3, Pre(p1,t1) = 2, ve M(*p3*) = 2, In(p3,t1) = 3 olduğundan t1 geçişi yetkilendirilmiştir.

3. Eğer, bir *p* giriş mevkisi bir *t* geçişine ağırlıklı yetkileme oku En*(p,t)* ile bağlanmış ise, *p* mevkisi, en az yetkileme oku ağırlığına eşit sayıda jeton içeriyorsa yani M(*p*) ≥ *En(p,t)* ise *t* geçişi yetkilendirilmiştir denir. Bu durum Şekil 1.(c)'de görülmektedir. Bu modelde görülen t1 geçişinin yetkilendirilebilmesi için p1 mevkisinde 2 veya daha fazla jeton, ve p3 mevkisinde en az 4 jeton bulunması gerekmektedir. Bu modelde, M(*p1*) = 3, Pre(p1,t1) = 2 ve M(*p3*) = 4, En(p3,t1) = 4 olduğundan t1 geçişi yetkilendirilmiştir.

Şekil 2.13.(d)'de görülen t1 geçişi, yukarıdaki tüm yekileme koşullarına aynı zamanda gereksinim duymaktadır. Bu modelde görülen t1 geçişinin yetkilendirilebilmesi için p1 mevkisinde 2 veya daha fazla jeton, p2 mevkisinde 3'ten az sayıda jeton ve p3 mevkisinde en az 4 jeton bulunması gerekmektedir. Bu modelde, M(*p1*) = 3; Pre(p1,t1) = 2, M(*p2*) = 2; In(p2,t1) = 3 ve M(*p3*) = 4; En(p3,t1) = 4 olduğundan t1 geçişi yetkilendirilmiştir.

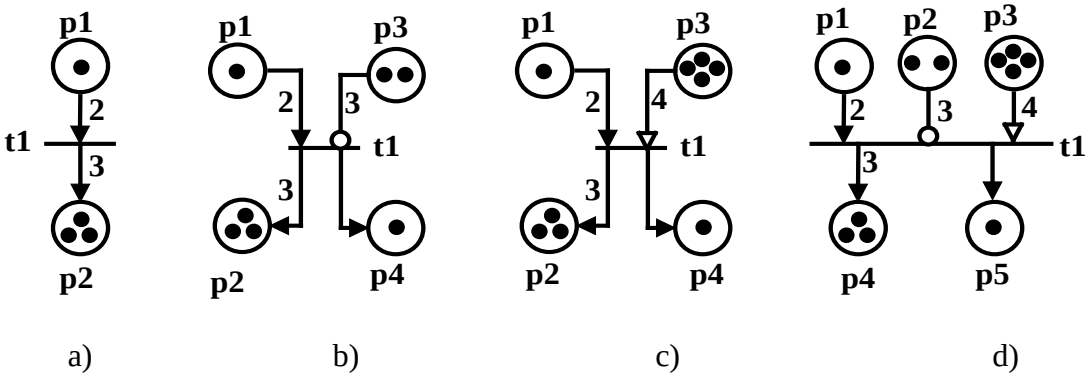


a) b) c) d)

Şekil 2.13 Yetkileme kurallarının gösterimi

Tetikleme kuralları: Ağırlıklı giriş, çıkış, yetkileme ve yasaklama oku içeren PN modellerinde yetkilendirilmiş bir *t* geçişinin tetiklenmesi ile, her bir giriş mevkisine bağlı *Pre*(p_i,*t)* giriş okunun ağırlığına eşit sayıda jeton p_i giriş mevkilerinden eksiltilerek, her bir *Post*(*t*,p_o) çıkış oku ağırlığına eşit sayıda jeton p_o çıkış mevkilerine aktarılır. Tetiklenme sonucunda ağırlıklı yetkileme ve ağırlıklı yasaklama oku bağlı olan

giriş mevkilerindeki jeton sayısı değiştirilmez. Tetikleme kuralları Şekil 2.13 ve Şekil 2.14'ten takip edilebilir. Tetikleme kuralları ışığında Şekil 2.13'te görülen modellerdeki geçişlerin tetiklenmesi ile oluşacak yeni jeton dağılımları Şekil 2.14'te görüldüğü gibi olmaktadır.



a) b) c) d)

Şekil 2.14 Tetikleme kurallarının gösterimi

PN modelinde bulunan yetkilendirilmiş bir geçişin tetiklenmesi ile ağdaki jeton dağılımı değişmektedir. M_0 başlangıç jeton işaretlemesi olmak üzere, eğer, M_0'ı M_i'ye ulaştıran bir tetikleme sırası mevcut ise M_i işaretlemesi erişilebilir (reachable) denir. Tetikleme sırası $\sigma = t_1, t_2, t_3, ..., t_m$ şeklinde gösterilmektedir. M_i işaretlemesinin M_0 işaretlemesinden erişilebilir olduğu $M_0 [\sigma > M_i$ şeklinde gösterilir. Eğer bir PN modeli ağırlıklı oklara sahip değil ise (okların ağırlıkları bire eşit ise) bu PN sıradandır (ordinary) denir. Bir PN modelinde bulunan mevkilerdeki jeton sayısı M_0 başlangıç işaretlemesinden ulaşılabilir tüm işaretlemeler için belli bir k sayısını geçmiyorsa bu PN k-sınırlıdır (k-bounded) veya sınırlı (bounded) denir. En fazla bir jeton bulundurabilen mevkilere güvenli (safe) mevki, tüm mevkileri güvenli (safe) olan PN modellerine ise güvenilir (safe) PN denir. Bir başka ifadeyle, eğer bir PN 1-sınırlı ise bu PN güvenilirdir denir. Bir PN'deki tüm jeton işaretlemelerinde, tüm mevkilerindeki jetonların toplamı sabit kalıyorsa, bu PN korunumlu (conservative) dur denir. PN'nin tüm işaretlemelerinden, ağı bir işaretlemeye götüren bir tetikleme sırası var ise ve bu tetikleme sıralamasında *t* geçişi yetkilendirilmiş ise bu *t* geçişi canlıdır (live) denir. Ağdaki tüm geçişler canlı ise PN canlıdır denir. PN modelinin canlı olması, modellenen sistemin çalışmasında kilitlenme (deadlock) olmadığını gösterir. Bir PN modelinde, eğer M_0 başlangıç jeton işaretlemesi tüm işaretlemelerden erişilebiliyorsa bu PN tersinirdir (reversible) denir.

2.6 Otomasyon Petri Ağları

Ayrık olay denetleyicilerini Petri ağları kullanarak tasarlamak mümkündür. Sıradan Petri ağları sensörler ve eyleyicilerle (actuators) ilgili olmadığından, sensörler ve eyleyicilerin her ikisini de içeren genişletilmiş bir Petri ağı yapısı olan Otomasyon Petri Ağları (Automation Petri Nets - APN) tanımlanmıştır [60]. Bir APN aşağıdaki gibi tanımlanır:

$$APN=(P, T, Pre, Post, In, En, X, Q, M_0) \quad (2.10)$$

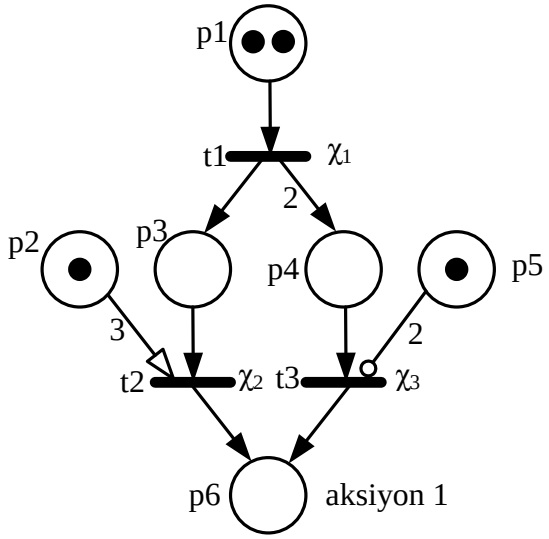
Burada;

- $P = \{p_1, p_2, \ldots, p_n\}$ sonlu sayıda boş olmayan mevkiler kümesidir.
- $T = \{t_1, t_2, \ldots, t_n\}$ sonlu sayıda boş olmayan geçişler kümesidir.
- Pre: $(P\times T) \rightarrow N$ mevkilerden geçişlere yönlenmiş sıradan ağırlıklı okları ifade eden bir giriş fonksiyonudur. N negatif olmayan tamsayılar kümesidir.
- Post: $(T\times P) \rightarrow N$, geçişlerden mevkilere yönlenmiş sıradan ağırlıklı okları ifade eden bir çıkış fonksiyonudur.
- In: $(P\times T) \rightarrow N$, mevkilerden geçişlere olan ağırlıklı yasaklama oklarını ifade eden bir yasaklama giriş fonksiyonudur.
- En: $(P\times T) \rightarrow N$, mevkilerden geçişlere olan ağırlıklı yetkileme oklarını ifade eden bir yetkileme giriş fonksiyonudur.
- $\chi = \{\chi_1, \chi_2, ..., \chi_m\}$ Geçişlere atanmış, sonlu sayıda boş olmayan tetikleme koşulları kümesidir.
- $Q=\{q_1, q_2, \ldots, q_n\}$, mevkilere atanabilir, sonlu sayıda aksiyonlar kümesidir.
- $M_0 : P \rightarrow N$ ilk işaretlemedir.

Şekil 2.15'te bir APN modeli görülmektedir. APN'de bulunan mevki, geçiş ve ok gösterimleri normal PN'deki gibidir. APN'de sensör bilgileri, geçişlerde tetikleme koşulları olarak kullanılır. Sensör bilgilerinin varlığı veya yokluğu APN'de, geçiş tetikleme ön koşullarıyla birlikte kullanılabilir. Bir *t* geçişiyle ilişkilendirilmiş olan bir χ, tetikleme koşulu, bir Boole değişkenidir, ilgili *t* geçişinin tetiklenmesine izin verilmediği bir durumda "0" olur veya ilgili *t* geçişi eğer açıksa tetiklenmesine izin verildiği durumda "l" olur. APN'in işareti, her mevkideki jeton sayısıyla; jetonlar

ise siyah noktalarla)(gösterilmektedir. Jetonların mevkiler arasındaki hareketi APN'in gelişimini gösterir ve bu, açık geçişlerin tetiklenmesiyle yapılır. Jetonların ağdaki akışı yetkileme ve tetikleme kuralları ile belirlenir. APN'deki yetkileme kuralları normal PN'ler için açıklanan kurallar ile aynıdır. Tetikleme kuralları ise aşağıda görülmektedir.

Tetikleme kuralları*:* APN'de yetkilendirilmiş bir *t* geçişi *t*'nin dış tetikleme koşulunun sağlanması ile tetiklenebilir. $\chi = 1$ olduğu özel durumda, *t* geçişi yetkilendirilmiştir ve her zaman tetiklenmesine izin verilmektedir. APN'de bulunan yetkilendirilmiş bir *t* geçişi tetiklendiğinde, normal PN'ler için tanımlanan tetikleme kurallarında olduğu gibi mevkiye bağlı ilgili giriş, çıkış, yetkileme ve yasaklama okunun ağırlığı ve bağlantısına göre jeton eksiltilmekte veya eklenmektedir.



Şekil 2.15 Örnek bir APN modeli

2.7 Kontrollü Petri Ağları

Kontrollü Petri ağları (Controlled Petri Nets-CtlPN), normal PN'lere harici yetkilendirme koşulları eklenmesi ile oluşturulmaktadır. Bu koşullar, ağdaki jetonların dolaşımını kontrol etmek için *kontrol mevkileri* olarak adlandırılan mevkilerin, sisteme eklenmesi ile sağlanmaktadır. CtlPN'ler ilk olarak Krogh ile Ichikawa ve Hiraishi tarafından önerilmiştir [74-75].

Bir kontrollü Petri ağı (CtlPN) bir üçlü ile tanımlanmaktadır.

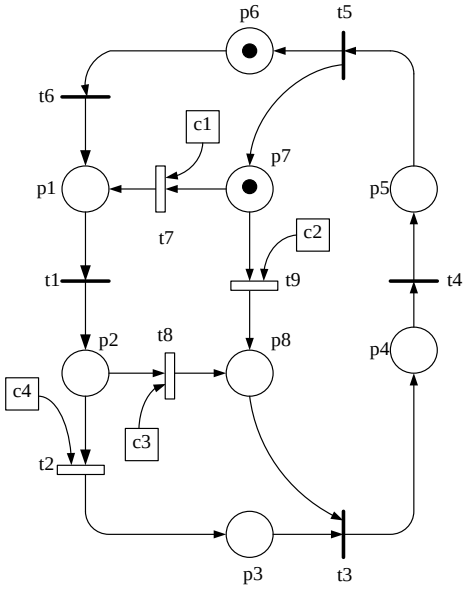
$$CtlPN=(N, C, B, m_0) \quad (2.4)$$

Burada; $N=(P, T, \varepsilon)$ basit bir PN'yi simgelemektedir. P ve T sırasıyla sonlu mevkiler ve geçişler kümesidir. $\varepsilon \subseteq (P \times T) \cup (T \times P)$ ise yönlenmiş oklar kümesini göstermektedir. C kontrol mevkilerini simgeleyen kümedir. $B \subseteq (c \times T)$ kontrol mevkilerinden geçişlere yönlenmiş olan okları, m_0 ise başlangıç jeton dağılımını göstermektedir. Şekil 2.16'da bir CtlPN'in grafiksel modeli görülmektedir. Bu modelde kareler ile gösterilen mevkiler kontrol mevkileridir. CtlPN'de geçişler kontrol edilebilir ve kontrol edilemeyen olmak üzere ikiye ayrılmaktadır. Kontrol mevkilerinin ilişkilendirildiği geçişler kontrol edilebilir $T_C = \{t \in T | \exists c, (c,t) \in B\}$, geriye kalanlar ise kontrol edilemeyen geçişlerdir $T_U = T \setminus T_C$.

CtlPN'nin Petri ağının durumu ağ içerisinde o anki jeton işaretlemesidir. Bu işaretleme $m: P \rightarrow Z$ şeklinde gösterilir. Muhtemel tüm işaretlemelerin kümesi M ile gösterilir. Bir u kontrolü, $C \rightarrow \{0,1\}$ kontrol mevkilerine ikili (binary) jeton sayısı ataması yapan bir işlemdir. Tüm kontrolleri içeren küme U ile simgelenmektedir. CtlPN'de durum değişimleri ağdaki yetkilendirilmiş geçişlerin tetiklenmesi ile sağlanmaktadır.

Şekil 2.16'dan görüleceği üzere kontrol edilebilir geçişler içi boş çubukla gösterilmektedir. Her kontrol edilebilir geçişe kare ile gösterilen bir kontrol mevkisi bağlanmaktadır. Kontrol edilebilir bir geçiş bu mevkide jeton olup olmaması ile kontrol edilmektedir. Normal geçiş ve mevkiler bilinen sıradan PN'lerde olduğu gibi çalışmaktadır.

CtlPN'de iki tür yetkilenme mevcuttur. Bunlar, durum yetkilemesi (State enable) ve kontrol yetkilemesidir (Control enable). Eğer bir geçişin girişinde bulunan normal mevkideki jeton sayısı $m(p) \geq 1$ ise o geçiş durum yetkilendirilmiştir. Benzer olarak, girişinde bir kontrol mevkisi bağlı geçiş için tanımlanmış kontrol u(c)=1 ise o geçiş kontrol yetkilendirilmiştir. Eğer u(c)=0 ise o geçiş yetkilendirilmemiştir. Kontrol edilemeyen geçişler için kontrol yetkilendirmesi söz konusu değildir.



Şekil 2.16 Bir kontrollü Petri ağı (CtlPN) modeli

[75]'de önerilen CtlPN yapısında, kontrol edilebilir bir geçişin tetiklenmesi sonucunda kontrol mevkisinde bulunan jeton alınmaktadır. Diğer modellerde [30, 31, 76] ise kontrol mevkisinde bulunan jeton, kontrol edilebilir geçişi yetkilendirmek için kullanılmaktadır. Bu bağlamda, kontrol mevkisinden kontrol edilebilir geçişe yönlenmiş ok, yetkileme oku olarak ta düşünülebilir. Eğer kontrol mevkisinde, jeton bulunuyorsa ilgili kontrol edilebilir geçiş yetkilendirilmiş demektir.

2.8 Petri Ağlarının PLC ile Gerçekleştirilmesi

Modern üretim sistemlerinin PN temelli kontrolü günümüz araştırmacıları tarafından çalışılan güncel bir konudur. Endüstrideki fabrikalarda bulunan üretim sistemlerinin kontrolünde, programlanabilir lojik denetleyiciler (Programmable Logic Controllers – PLCs) yaygın olarak kullanılmaktadır. PLC'lerin programlamasında endüstride yaygın olarak merdiven diyagramı kodu kullanılmaktadır. Bu nedenlerle, PN olarak elde edilmiş olan denetleyicinin PLC koduna dönüştürülmesi gerekmektedir. Bu gereksinimi karşılamak üzere jeton aktarım lojiği (Token Passing Logic-TPL) adıyla bir dönüşüm yöntemi önerilmiştir [37, 58-61]. AOS'lerin PLC merdiven diyagramı kodu ve PN temelli kontrolü konusunda [55]'e bakılabilir. [55]'ten açıkça görülebileceği üzere TPL yöntemi kontrol konusunda çalışan gruplar tarafından kabul görmüş bir yöntemdir. Bu

yöntemin en büyük özelliği PN'den merdiven diyagramı koduna doğrudan dönüşümü mümkün kılmasıdır.

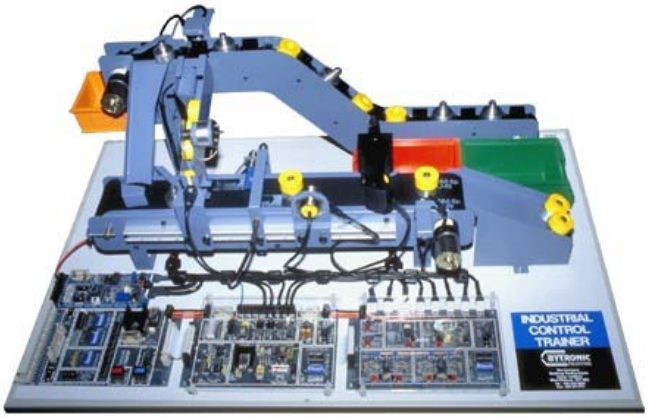
Sıradan ve güvenilir PN yapısından, PLC merdiven diyagramı koduna dönüşüm için ilk olarak, PN modelinde bulunan her bir mevki için ayrı bir Boole değişkeni (hafıza biti) atanır. Bu hafıza bitinin set (1) olması temsil ettiği mevkide jeton olması, reset (0) olması ise temsil ettiği mevkide jeton bulunmaması durumlarını ifade etmektedir. Petri ağı modelinde bulunan geçişler, çoğu PLC'nin sahip olduğu set [--(S)--] ve reset [--(R)--] bobinleri kullanılarak her biri ayrı bir basamakta gerçekleştirilir. Petri ağlarında jetonların dolaşımı geçişlerin tetiklenmesi ile olmaktadır. TPL yönteminde, mevkileri temsil eden hafıza bitlerinin, geçişlerin gerçekleştirildiği basamaklarda set ve resetlenmesi ile jeton dolaşımı karakterize edilir. Kapasitesi birden fazla olan mevkilere sahip PN modellerinin de TPL kullanılarak PLC merdiven diyagramı koduna dönüşümü mümkündür. Bu dönüşümde mevkileri karakterize etmek için hafıza bitleri yerine çoğu PLC'de hazır bulunan sayıcı (Counter) fonksiyonları kullanılmaktadır. Sayıcının sayma değeri ilgili mevkide bulunan jeton sayısını simgelemektedir. Ağdaki jeton dolaşımı sayıcının sayma değerinin arttırılması veya azaltılması ile sağlanmaktadır.

Tez kapsamında önerilen Petri ağı gerçekleştirmelerinde, TPL metodu kullanılmaktadır. Gerçekleştirme işlemleri tezin ileriki kısımlarında detaylı olarak örnekler üzerinde açıklanmıştır.

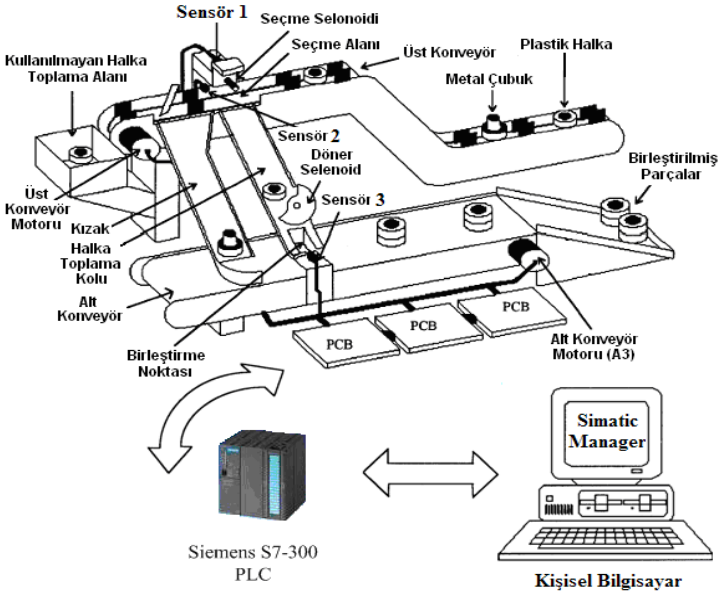
2.9 Deneysel Endüstriyel Üretim Sistemi

Endüstriyel Kontrol Deney Seti (ICT-Industrial Control Trainer), Bytronic firması tarafından geliştirilmiş ve eğitim-ve-araştırma amaçlı kullanılmak üzere hazırlanmış bir deneysel endüstriyel üretim sistemi modelidir. Bu sistem üzerinde herhangi bir denetleyici yapısı bulunmamaktadır. Deney seti içerisindeki motor ve sensörlerin bağlantı uçları konnektörlere bağlanmıştır. Denetleyiciler, deney seti üzerinde bulunan bu konnektörler yardımıyla sisteme bağlanmaktadırlar. Şekil 2.17'de görülen endüstriyel kontrol deney setinin üzerinde bulunan sensör, selenoid ve motorların yerleri Şekil 2.18'deki şemada detaylı olarak görülmektedir. Şekil 2.18'den de görüleceği gibi bir Siemens S7-300 (CPU 319) PLC, bu sisteme kontrolör olarak

bağlanmıştır. Simatic Manager isimli bilgisayar programı kullanılarak PLC programlanabilmektedir.

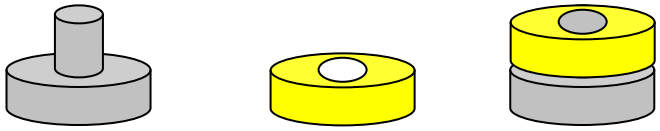


Şekil 2.17 Endüstriyel kontrol deney setinin (Industrial Control Trainer - ICT) resmi



Şekil 2.18 Endüstriyel üretim sistemi deney düzeneği

Bu deney seti; parça tanıma, birleştirme ve reddetme işlemlerinin yapılmasında kullanılmak üzere tasarlanmıştır. Parça olarak kullanılan metal çubuk ve plastik halkalar Şekil 2.19'da görülmektedir.



a) b) c)

Şekil 2.19 Parçaların şekilleri,
a) Metal çubuk, b) Plastik halka, c) İki parçanın birleştirilmiş şekli

Deney setinde bulunan üst konveyör ve alt konveyör sırasıyla üst konveyör motoru (A0) ve alt konveyör motoru(A1) ile çalıştırılmaktadır. Metal çubuklar ve plastik halkalar rastgele bir şekilde üst konveyör tarafından seçme işleminin yapılacağı seçme bölgesine taşınırlar. Metal çubuk ve plastik halkayı tanımlayıp birbirinden ayırt etmek için iki sensör (Sensör 1: proximity sensör, Sensör 2: infra-red reflective sensör) kullanılmaktadır. Parça seçme selenoidi (itici selenoid - A2) ile plastik halkalar, halka toplama kolu üzerine aktarılırlar. Halka toplama kolu üzerine maksimum beş adet halka yerleştirilebilmektedir. Metalik çubuklar ise üst konveyörün sonuna doğru yerleştirilmiş olan bir engele çarparak, üst konveyörden alt konveyöre kızak üzerinden aktarılırlar. Kullanılmayan ve üst konveyörün sonuna kadar giden plastik halkalar bu engele çarpmadan üst konveyör sonundaki kullanılmayan halka toplama alanına düşerler. Sensör 3 (infra-red emitter/detector) birleştirme noktasının boş olup olmadığını tespit etmek için kullanılır. Birleştirme noktasının boş olduğu durumda birleştirme selenoidi (döner selenoid - A3) halka besleme kolundan birleştirme noktasına bir plastik halka aktarmak için kullanılır. Birleştirme noktası alt konveyörün biraz üzerine yerleştirilmiştir ve bir metalik çubuk alt konveyör üzerinde bu noktadan geçtiği anda burada bulunan plastik halka ile birleştirilir. Böylece metalik çubuk ile plastik halka birleştirilerek son ürün oluşturulur. Alt konveyör, birleştirilmiş parçaları toplama kabına iletir.

Deney seti üzerinde bulunan motor, selenoid ve sensörlerin görevleri ile bu elemanların Siemens S7-300 PLC'ye bağlantı noktaları Çizelge 2.3'te görülmektedir.

Çizelge 2.3 Deney seti üzerinde bulunan motor, selenoid ve sensörlerin görevleri

İsim	PLC bağlantı noktası	Açıklama
A0	Q0.0	Seçme Selenoidi (İtici Selenoid)
A1	Q0.1	Birleştirme Selenoidi (Döner Selenoid)
A3	Q0.3	Üst Konveyör Motoru
A4	Q0.4	Alt Konveyör Motoru
Sensör 1	I0.0	Seçme Alanı (Yakınlık Sensörü)
Sensör 2	I0.1	Seçme Alanı (IR Sensör)
Sensör 3	I0.2	Birleştirme Noktası

BÖLÜM III

MELEZ YÖNTEMİN GERÇEK SİSTEMLERİN KONTROLÜNE UYGULANMASI

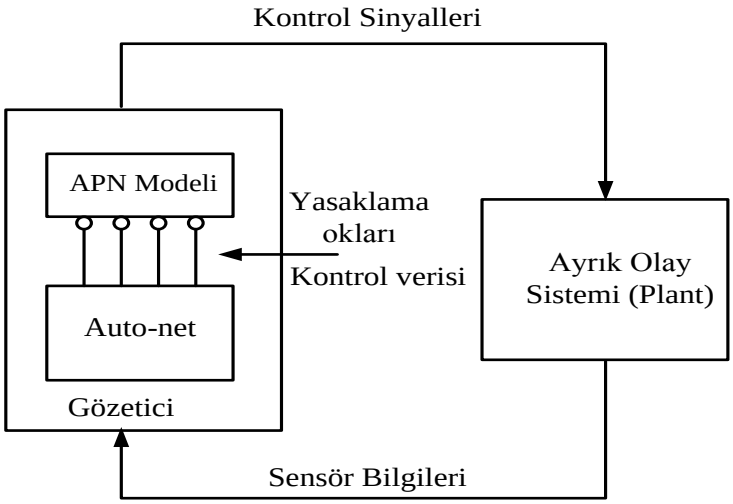
AOS'lerin kontrolünde yaygın olarak kullanılan iki yapının (Petri ağları ve Gözetimli kontrol teorisi) sunduğu avantajlardan aynı anda yararlanan melez bir yöntem Uzam ve Wonham tarafından [52]'de önerilmiştir. Önerilen yöntem ile Petri ağlarının sunduğu yüksek modelleme kabiliyeti ile Gözetimli kontrol teorisinin sunduğu genellik, optimal yani maksimum müsaade edici ve bloklanmasız denetleyicilerin elde edilebilirliği gibi avantajlardan aynı anda yararlanılmaktadır. [52]'de etkin bir denetleyici hesaplama yöntemi önerilmesine karşın, bu yöntemin gerçek bir sistemin kontrolünde uygulanabilirliği gösterilmemiştir. [52]'de önerilen yöntem genel PN yapıları için önerilmiştir.

Bu bölümde [52]'de önerilmiş olan melez yöntemin gerçek ayrık olay sistemlerinin kontrolüne uygulanabilirliği iki farklı endüstriyel sistem kullanılarak gösterilmiştir. Bunlardan ilki, Niğde Üniversitesi Elektrik-Elektronik Mühendisliği Bölümü Laboratuarında bulunan uzaktan kumandalı otomatik kapı modelinin kontrolüdür. Bu sistem adı geçen yöntem kullanılarak başarılı bir biçimde kontrol edilmiştir [66]. İkinci uygulama olarak, [52]'de önerilen melez yöntemin TÜBİTAK 107E125 kodlu bilimsel araştırma projesi kapsamında oluşturulan deney seti kullanılarak gerçek sistemlerin kontrolünde uygulanabilirliği gösterilmiştir [67]. [52]'de önerilmiş olan yöntem ve bu iki uygulama aşağıdaki kısımlarda detaylı olarak anlatılmaktadır.

3.1 AOS'lerin PN ve SCT Temelli Melez Kontrolü

AOS'lerin PN ve Gözetimli Kontrol Teorisi kullanılarak melez kontrolü [52]'de önerilmiştir. AOS'de gözetici, sistemden sensörler yardımıyla gelen geri besleme bilgisini kullanarak, sistemdeki bazı olayları yasaklamak suretiyle sistemi denetlemektedir. Şekil 3.1'de görüldüğü gibi gözetici iki kısımdan oluşmaktadır. Plant olarak adlandırılan AOS ve gözetici eş zamanlı olarak çalışmaktadır. AOS'de meydana gelen bir olay oluşumu, sensör bilgisi olarak gözeticiye ulaşmakta ve gözeticinin durum değişimi bu olay oluşumuna göre olmaktadır. Gözetici, durum geri besleme kontrolü

yapmaktadır. Hesaplanan gözetici derlenmiş tip gözetici olup, AOS'nin ürettiği dili, kontrol edilmiş davranışı modelleyen dil olmaya zorlamaktadır. Bu yöntem kullanılarak AOS denetleyicisi tasarımı için ilk olarak, Kontrol edilecek sistemin (Plant) kontrol edilmemiş APN modelinin oluşturulması gerekir. Bu işlemin ardından; Petri ağı modeli tampon modelleri kullanılarak otomata modellerine çevrilmeli ve spesifikasyonlar otomata olarak modellenmelidir. Bu işlemlerin ardından TCT yazılımı kullanılarak gözetici tasarlanır. Otomata olarak elde edilen gözleticinin auto-net olarak karşılığı bulunarak TCT yazılımın hesapladığı kontrol verileri auto-netin durumlarından Petri ağı modelinin geçişlerine bağlanan yasaklama okları ile gerçekleştirilir. Kontrol edilmiş model, APN modeli ile auto-net modelinin birleştirilmesi ile elde edilir.



Şekil 3.1 AOS'lerin melez kontrolü

Auto-net (Automaton Net), SCT ile hesaplanan Ramadge-Wonham gözeticisinin PN benzeri grafiksel gösterimidir. Bu nette yalnızca bir adet jeton bulunmakta ve mevkilerinin kapasitesi bir (1) dir. Gözeticiye ait otomata modelinde bulunan her bir durumun PN mevkisi, her bir geçişin sıradan PN oku şeklinde gösterilmesi ile gözeticinin auto-net modeli elde edilebilir. Auto-net modelinde başlangıçta, gözeticinin başlangıç durumunu simgeleyen mevkiye bir adet jeton konulur. Auto-net yapısındaki geçişler kontrol edilebilir veya kontrol edilemeyen olabilir. Auto-net AOS'nin kontrol edilmiş davranışını temsil eder ve rolü kontrolü (olayları yasaklamayı) sağlamaktadır. Auto-netin durum değişimi plant'taki olay oluşumuna senkron gerçekleşir. APN modelindeki ve auto-net'teki olaylar aynı etikete sahip olarak senkron çalışmaktadır. Kullanılan yasaklama okları ile kontrol edilebilir olay oluşumlarına müdahale

edilebilmektedir. [52]'de önerilen yöntem kullanılarak gözetici tasarım aşamaları aşağıdaki gibidir:

1. Sistemin kontrol edilmemiş davranışı APN olarak modellenir. Yasaklanmış durum spesifikasyonları belirlenir.
2. APN modelinin sınırlı (bounded) olduğu ve mevkilerinin jeton kapasiteleri belirlenir.
3. Eğer mümkünse Petri ağı sadeleştirme kuralları ile APN sadeleştirilir.
4. APN modelindeki mevkiler [52]'de önerilen yöntem kullanılarak eşdeğer tampon (buffer) modellerine dönüştürülür. Spesifikasyonlar otomata olarak modellenir.
5. Gözetimli Kontrol Teorisi ile gözetici (SUPER) ve kontrol verisi (CDAT) hesaplanır.
6. SUPER, auto-nete (AUTONET) dönüştürülür.
7. AUTONET, PLANT'a yasaklama okları ile bağlanarak kapalı çevrim melez model elde edilir.
8. Elde edilen kapalı çevrim melez kontrol sistemi bir PLC ile gerçekleştirilir.

İkinci adımdaki sınırlılık hesaplaması standart PN yazılımları kullanılarak ya da gözlemle yapılabilir. Üçüncü adımdaki, sadeleştirme kuralları literatürde mevcut olup, PN modelinde canlılık, sınırlılık ve tersinirlik gibi özellikleri değiştirmemektedir.

Dördüncü adımda gerekli olan dönüşüm işlemleri için temel bir yaklaşım [52]'de mevcuttur. Bu yaklaşıma göre, eğer bir mevkinin jeton kapasitesi *b* ise, bu mevki, kapasitesi *b+1* olan bir tampon modeline dönüştürülmektedir. Bu temel düşünce ışığında, PN modellerinde karşılaşılabilecek çeşitli durum ve modeller için dönüşümler konusunda tez kapsamında araştırmalar yapılmıştır. Yapılan çalışmalar, tezin beşinci bölümünde ve [90]'da sunulmuştur.

Beşinci adımda yapılması gereken hesaplamalar TCT yazılımı [68] kullanılarak yapılabilir. Tezin ikinci bölümünde TCT yazılımının kullanımı detaylı olarak açıklanmıştır. APN modelinde bulunan mevkiler (p1, p2, p3, ... pn) bu mevkilerin tampon karşılıklarının otomata modelleri P1, P2, P3, ..., PN olmak üzere beşinci adımda yapılacak işlemler aşağıdaki gibi özetlenebilir:

5.1 AOS'lerin Gözetimli Kontrol Teorisi yardımıyla kontrolünde yapılacak olan ilk işlem sistemi oluşturan tüm alt bileşenlerin (p1, p2, …, pN) otomata olarak

oluşturulması ve bu otomata modellerinin TCT yazılımında *Create(.)* komutu kullanılarak gerçekleştirilmesi işlemidir:

P1=Create(P1)
P2=Create(P2)
.
.
Pn=Create(Pn)

5.2 Elde edilen alt bileşenlerin senkron kompozisyonu ile sistemin kontrol edilmemiş modeli oluşturulur (PLANT=P1||P2||…||PN). Bu işlem TCT yazılımında *Sync(.)* komutu kullanılarak gerçekleştirilir:

PLANT=Sync(P1,P2)
PLANT=Sync(PLANT,P3)
.
.
PLANT=Sync(PLANT,PN)

5.3 Gözetici tasarımında yapılacak diğer işlemde, sistemin çalışmasında istenilen özelliklerin her biri *Create(.)* komutuyla ayrı ayrı elde edilir (SPEC1, SPEC2, ..., SPECM):

SPEC1=Create(SPEC1)
SPEC2=Create(SPEC2)
.
.
SPECM=Create(SPECM)

Daha sonra elde edilen her bir spesifikasyon modeline kontrol edilmemiş sistem modeli PLANT'ta bulunan fakat ilgili spesifikasyon tarafından kısıtlanmayan olaylar ilgili spesifikasyon modelinin tüm durumlarına *Selfloop(.)* komutu kullanılarak özdöngü olarak eklenir:

SPEC1=Selfloop(SPEC1, PLANT'ta bulunan diğer olaylar)
SPEC2=Selfloop(SPEC2, PLANT'ta bulunan diğer olaylar)
.
.
SPECM=Selfloop(SPECM, PLANT'ta bulunan diğer olaylar)

Son olarak elde edilen tüm spesifikasyon modelleri (SPEC1, SPEC2, .., SPECM) TCT yazılımında bulunan *Meet(.)* komutu kullanılarak sistemin çalışma şeklini ifade eden tek bir spesifikasyon (SPEC) modeli olarak elde edilir:

SPEC=Meet(SPEC1, SPEC2)
SPEC=Meet(SPEC, SPEC3)

.
.
SPEC=Meet(SPEC, SPECM)

5.4 *SUP=(PLANT, SPEC)* komutu kullanılarak sistemin kontrol edilmiş şeklini ifade eden gözetici modeli SUP elde edilir.

5.5 Bir önceki adımda elde edilen SUP gözetici modeline göre daha az duruma sahip olan indirgenmiş gözetici modelinin TCT yazılımıyla elde edilmesi için *DAT=Condat(PLANT, SUP)* komutu kullanılarak DAT kontrol bilgisi elde edilir.

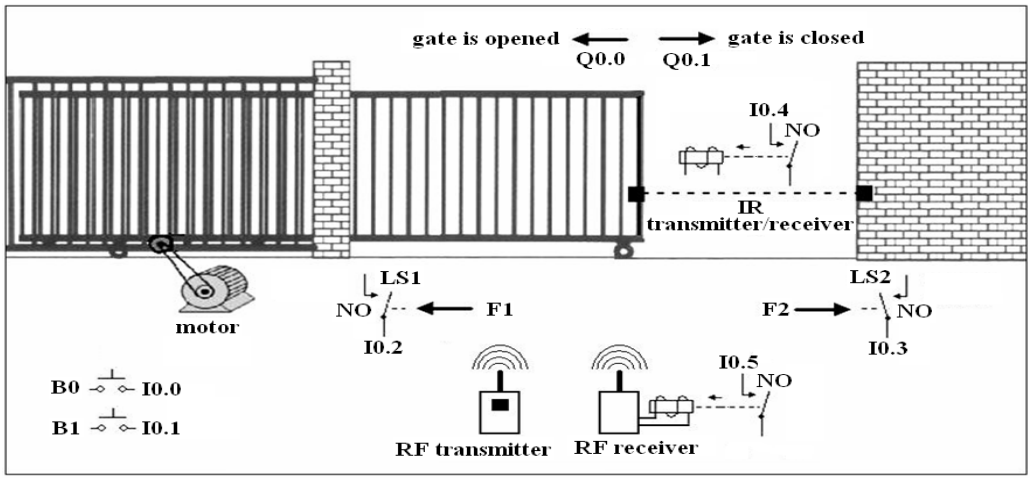
Altıncı adımda, TCT ile elde edilen otomata formundaki gözetici, PN'lere benzer bir yapı olan auto-net formuna dönüştürülür. Bu işlemde, gözetici modelinde bulunan özdöngüler, durumlarda değişikliğe sebep olmadığından ihmal edilir.

Yedinci adımda, auto-net gözetici PLANT'a kontrol verisinde hesaplanan yasaklama okları ile bağlanarak kapalı çevrim melez model elde edilir.

Son adımda, kapalı çevrim melez kontrol sistemi PLC ile gerçekleştirilir.

3.2 Uygulama 1: Uzaktan Kumandalı Otomatik Kapı Kontrolü

İlk uygulamada ayrık olay sistemi olarak uzaktan kumandalı bir otomatik kapı modeli kumanda edilecektir. Uzaktan kumandalı otomatik kapı modelinin blok şeması Şekil 3.2'de görülmektedir. Kapı bir motor yardımıyla açılıp kapanmaktadır. Q0.0 ve Q0.1 sırasıyla Aç ve Kapat eylemlerini gerçekleştiren motor kontrol sinyalleridir. I0.0, Aç eylemini başlatacak olan butonu simgelemektedir. I0.0 butonuna basıldığında Q0.0 çıkışı aktif olarak motor geri yönde çalışmakta ve kapı açılmaktadır. I0.2 ve I0.3 kapının tam olarak açılıp kapandığını algılayan sınır anahtarlarıdır. I0.4 kapı aralığında herhangi bir cismin olup olmadığını algılayan kızılötesi (infra-red) sensördür. I0.5 ise RF ile çalışan uzaktan kumanda kontağını simgelemektedir.



Şekil 3.2 Uzaktan kumandalı kapı modeli blok şeması

Bu sistem için Petri ağları ve SCT temelli melez kontrol sistemi aşağıdaki adımlar takip edilerek elde edilir:

1. Adım: Uzaktan kumandalı otomatik kapı sistemine ait APN modeli Şekil 3.3'te görülmektedir. APN modelinde bulunan geçişlerin harici tetikleme koşulları Çizelge 3.1'de verilmiştir. Bu sistemde, istenilen çalışma özellikleri (Spesifikasyonlar) aşağıda sıralanmıştır.

1) Kapı kapalı iken aç butonuna (I0.0) veya uzaktan kumanda butonuna (I0.5) basıldığında kapı açılmaya başlayacaktır.
2) Kapı açılıp 1. sınır anahtarı (I0.2) konum değiştirince, kapının açıldığı anlaşılıp motor duracak ve kapı açık konumda 5 saniye bekletilecektir. Süre sonunda kapı kapanmaya başlayacaktır.
3) Kapı kapanırken kapı aralığına herhangi bir cisim girerse yani $\overline{I0.4}$ sinyalinin oluşmasıyla motor geri yönde çalışmaya başlatılacak ve kapı açılacaktır.
4) Kapı tam olarak kapandığında, kapının kapandığını gösteren 2. sınır anahtarı (I0.3) konum değiştirince motor duracaktır.

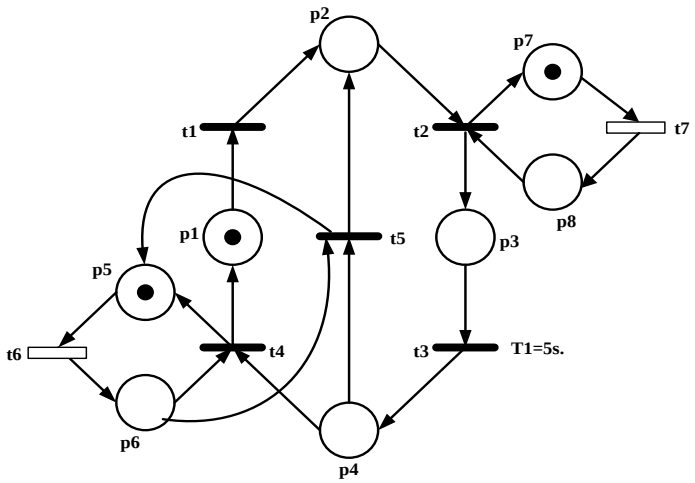
2. Adım: Şekil 3.3'te görülen APN modeli sınırlıdır. Bu modelde bulunan mevkilerin tümünün jeton kapasitesi bir(1)'dir.

3. Adım: Model basit bir model olup, sadeleştirmeye gerek duyulmamaktadır.

4. Adım: APN modelinde bulunan her bir mevki için, ilgili ok bağlantıları dikkate alınarak Şekil 3.4'te görüldüğü gibi, tampon eşdeğer modelleri elde edilmiştir. APN modelinde bulunan geçişlerden t6 ve t7 geçişi kontrol edilebilir diğer geçişler ise kontrol edilemeyen geçişlerdir. TCT yazılımında bu geçişler için kullanılan olay etiketleri aşağıda görülmektedir. Bu sistem için yukarıda belirtilen spesifikasyonlara ait otomata modelleri ise Şekil 3.5'te görülmektedir. İlk spesifikasyon modeli (SP1), 10 (Kapıyı aç) veya 50 (Kapı arasında cisim var) etiketli olayların gerçekleşmesinden sonra 20 (kapı açıldı) etiketli olay gerçekleşene kadar motorun geri yönde çalışarak (71 etiketli olay) kapının açılacağını ifade etmektedir. İkinci spesifikasyon modeli (SP2) ise, kapı açıldıktan sonraki bekleme süresinin bitiminden (30 etiketli olay) sonra 40 (kapı kapandı) veya 50 (Kapı arasında cisim var) olaylarından biri gerçekleşene kadar motorun ileri yönde çalışarak (61 etiketli olay) kapın kapanacağını ifade etmektedir.

TCT olay etiketleri:

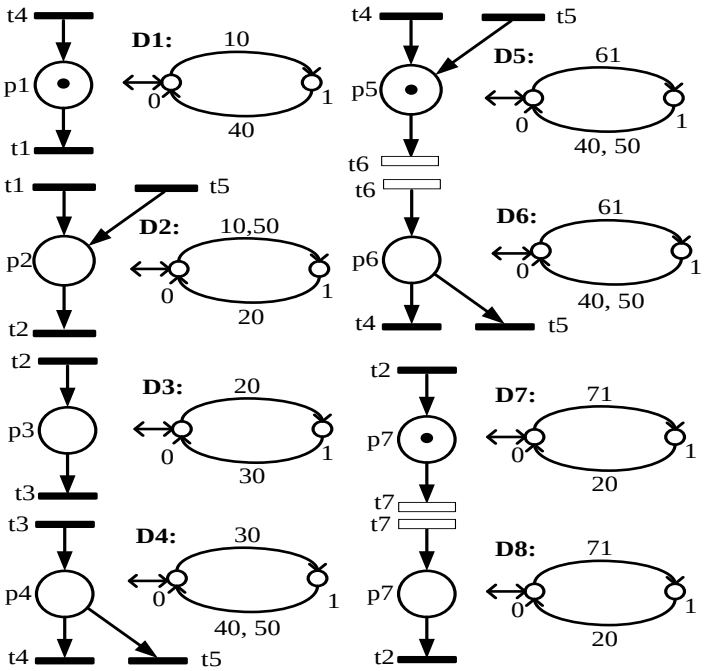
APN:	t1	t2	t3	t4	t5	t6	t7
TCT:	10	20	30	40	50	61	71



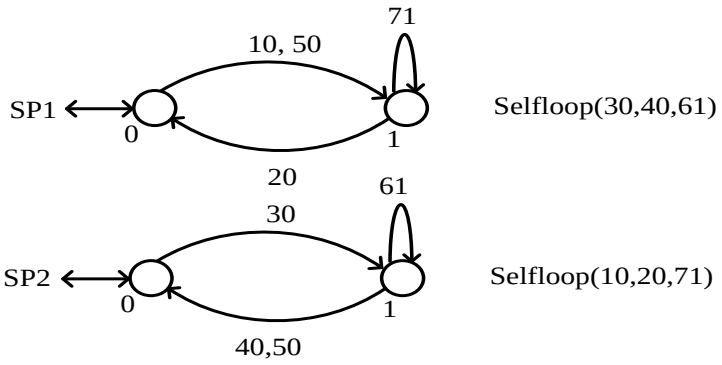
Şekil 3.3 Kapının kontrol edilmemiş APN modeli

Çizelge 3.1 Kapının APN modelinde bulunan geçişler ve harici tetikleme koşulları

Geçiş	Açıklama ve Harici Tetikleme Koşulu
t1	Kapıyı aç (I0.0 veya I0.5)
t2	Kapı açıldı (I0.2)
t3	Bekleme süresi bitti (T1=5s.)
t4	Kapı kapandı (I0.3)
t5	Kapı arasında cisim var ($\overline{\text{I0.4}}$)
t6	Motoru ileri yönde çalıştır
t7	Motoru geri yönde çalıştır



Şekil 3.4 Petri ağı modelindeki mevkilerin otomata karşılığı



Şekil 3.5 Kontrol spesifikasyonlarının otomata modelleri

5. Adım: Bu sistem için TCT yazılımı kullanılarak RW gözeticisi aşağıdaki gibi elde edilmiştir. İlk olarak, *Create(.)* komutu ile tampon modellerine ve spesifikasyonlara ait otomata modelleri aşağıdaki gibi tanımlanır:

D1 = Create(D1,[mark 0],[tran [0,10,1],[1,40,0]]) (2,2)
D2 = Create(D2,[mark 0],[tran [0,10,1],[0,50,1],[1,20,0]]) (2,3)
D3 = Create(D3,[mark 0],[tran [0,20,1],[1,30,0]]) (2,2)
D4 = Create(D4,[mark 0],[tran [0,30,1],[1,40,0],[1,50,0]]) (2,3)
D5 = Create(D5,[mark 0],[tran [0,61,1],[1,40,0],[1,50,0]]) (2,3)
D6 = Create(D6,[mark 0],[tran [0,61,1],[1,40,0],[1,50,0]]) (2,3)
D7 = Create(D7,[mark 0],[tran [0,71,1],[1,20,0]]) (2,2)
D8 = Create(D8,[mark 0],[tran [0,71,1],[1,20,0]]) (2,2)

PLANT, PLANT=D1||D2||D3||D4||D5||D6||D7||D8 senkron kompozisyonu ile aşağıdaki gibi elde edilmiştir. PLANT modelinde 16 adet durum ve 30 adet geçiş bulunmaktadır.

PLANT = Sync(D1,D2) (4,7) Blocked_events = None
PLANT = Sync(PLANT,D3) (8,16) Blocked_events = None
PLANT = Sync(PLANT,D4) (4,5) Blocked_events = None
PLANT = Sync(PLANT,D5) (8,12) Blocked_events = None
PLANT = Sync(PLANT,D6) (8,12) Blocked_events = None
PLANT = Sync(PLANT,D7) (16,30) Blocked_events = None
PLANT = Sync(PLANT,D8) (16,30) Blocked_events = None

Spesifikasyonlar aşağıda görüldüğü gibi tanımlanmış ve *Meet(.)* komutu ile birleştirilerek SPEC elde edilmiştir.

S1 = Create(S1,[mark 0],[tran [0,10,1],[0,50,1],[1,20,0],[1,71,1]]) (2,4)
S2 = Create(S2,[mark 0],[tran [0,30,1],[1,50,0],[1,40,0],[1,61,1]]) (2,4)
S1 = Selfloop(S1,[30,40,61]) (2,10)
S2 = Selfloop(S2,[10,20,71]) (2,10)
SPEC = Meet(S1,S2) (4,13) Blocked_events = None

Bu işlemin ardından TCT yazılımındaki *Supcon(.)* komutu kullanılarak, sistemi spesifikasyonları yerine getirmeye zorlayacak gözetici ve yasaklanacak olay listesi hesaplanır. Bu hesaplama için yapılan TCT işlemleri aşağıda görülmektedir.

SUPER = Supcon(PLANT,SPEC) (6, 7)
CDAT = Condat(PLANT,SUPER) (Controllable).

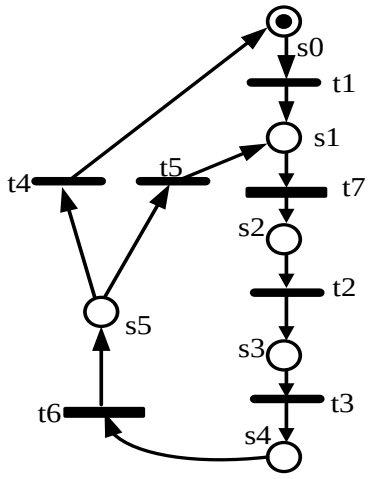
Bu işlemlerin ardından elde edilen gözetici ve yasaklama listesi aşağıdaki gibidir.

SUPER: #states: 6 states set:0…5 initial state: 0 #trans..7
Transitions:
[0 10 1], [1 71 2],[2 20 3],[3 30 4],[4 61 5],[5 40 0],[5 50 1]

CDAT:
0: 61 71 1: 61 2:61 3:61 71 4: 71 5: 71

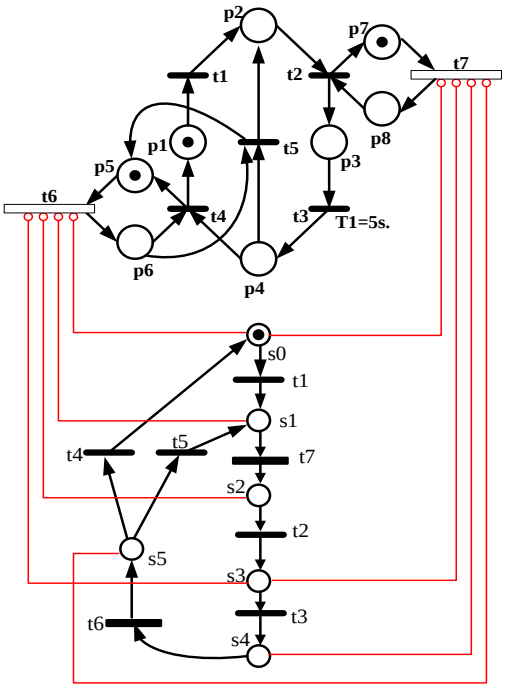
Burada, [0 10 1] gösterimi, sistem ‘0’ etiketli durumdayken ‘10’ etiketli olayın gerçekleşmesi ile ‘1’ etiketli duruma geçeceğini ifade etmektedir. Ayrıca, yasaklanacak olay listesindeki 0: 61 71 gösterimi ise gözeticinin ‘0’ etiketli durumunda ‘61’ ve ‘71’ etiketli olayların yasaklanacağını ifade etmektedir.

6.Adım: TCT yazılımı kullanılarak elde edilen otomata modeli Şekil 3.6'da görüldüğü gibi auto-nete çevrilmiştir. Buradaki yapıda tüm ağda sadece bir adet jeton bulunmakta ve mevkilerin kapasitesi bir jeton ile sınırlıdır.



Şekil 3.6 Gözeticinin auto-net modeli

7. Adım: Bu işlemlerin ardından elde edilen auto-net yapısındaki gözetici ve Petri ağı yapısındaki sistem için TCT ile elde edilen yasaklanacak olay listesindeki olaylar yasaklama oku kullanılarak gerçekleştirilir. Gözeticinin auto-net modelinin yasaklama okları ile sistemin APN modeline bağlanması ile elde edilen melez yapı Şekil 3.7'de görülmektedir.

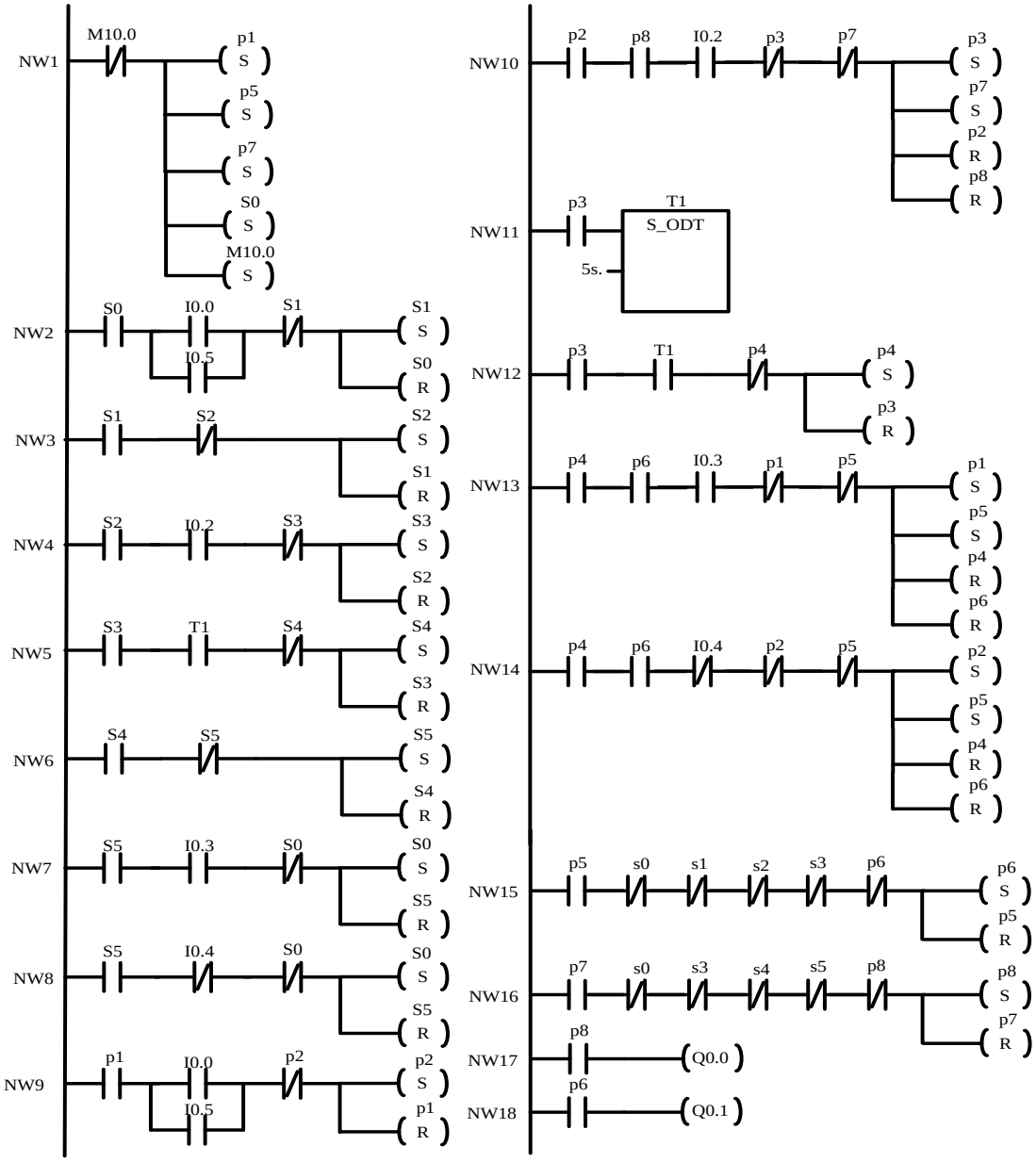


Şekil 3.7 Gözeticinin auto-net modeli ve sistemin APN modelini içeren melez yapı

8. Adım: Petri ağlarını merdiven diyagramı kullanarak PLC ile gerçekleştirmek için öncelikle mevkilerdeki jetonları temsil etmek için her bir mevkiye, bir bayrak (flag, marker veya dahili röle) atanır. Zamanlı geçişleri gerçekleştirmek için ise Düz zaman rölesi (On Delay Timer) kullanılır. Basamak diyagramında üst basamaklara yazılan kod alt basamaktaki koddan daha önce tarandığından kodda ilk olarak başlangıç jeton dağılımı, ardından gözetici ve son olarak ta sistemin Petri ağı modeli gerçekleştirilmektedir. Kod tarama önceliğinden dolayı ilk olarak gözeticinin durum değişikliğine bakılmakta daha sonra sistem gözetici tarafından arzulanılan çalışmaya zorlanmaktadır. Uzaktan kumandalı kapı modelinin melez yöntem ile kontrolü için elde edilen denetleyici yapısının PLC ile gerçekleştirilmesi için yazılan kod Şekil 3.8'de görülmektedir. Şekildeki kodun daha anlaşılır olabilmesi için kullanılan marker bitleri yerine APN modeli ve hesaplanan gözeticide kullanılan mevki ve durum etiketleri kullanılmıştır. Başlangıç işlemleri kodun ilk basamağında (NW1) gerçekleştirilmiştir. Kodun ilk taranması esnasında M10.0 marker biti '0' olduğundan başlangıçta içerisinde jeton bulunan mevkiler için atanan merker bitlerini set etmektedir. Kodun birinci basamağının sonunda M10.0 biti set edilmektedir. Böylelikle ikinci taramada bu basamak bir daha çalıştırılmamaktadır.

İkinci ve sekizinci basamaklar (NW2-NW8) arasındaki kod ile gözetici gerçekleştirilmiştir. Gözetici modelindeki geçişlere ait harici tetikleme şartları sistem modelindeki aynı etiketli geçişin şartına eşittir. Dokuzuncu ve on altıncı basamaklar (NW9-NW16) arasındaki kod ile sistemin APN modeli gerçekleştirilmiştir. Son olarak NW17 ve NW18'de motorun ileri ve geri çalışmasını sağlayan çıkış atamaları yapılmıştır. PLC merdiven diyagramı kodunda görülen PLC komutlarının açıklamaları ve kodda kullanılan düz zaman rölesi (S_ODT) ile ilgili bilgiler EK-A'da verilmiştir.

Şekil 3.8'de görülen PLC merdiven diyagramı kodu Siemens S7-300 PLC'ye yüklenerek, sistemin spesifikasyonlarda belirtilen çalışmayı başarılı bir şekilde gösterdiği gözlemlenmiştir.



Şekil 3.8 Melez kontrolör yapısının PLC kodu karşılığı

3.3 Uygulama 2: Deneysel Endüstriyel Üretim Sisteminin Kontrolü

İkinci uygulama olarak, detayları ikinci bölümde verilen endüstriyel üretim sistemi içeren deney düzeneği kullanılmıştır. Endüstriyel üretim sistemi için Petri ağları ve SCT temelli melez kontrol sistemi aşağıdaki adımlar takip edilerek elde edilmiştir.

1. Adım: Endüstriyel üretim sisteminin kontrol edilmemiş APN modeli Şekil 3.9'da görülmektedir Bu modelde on adet mevki P = {p1, p2, ..., p10} ve $\chi = \{\chi_1, \chi_2, ..., \chi_7\}$ harici tetikleme koşulları ile ilişkilendirilmiş yedi adet geçiş T = {t1, t2, ..., t7}

bulunmaktadır. harici tetikleme koşulları, sırası ile $\chi_1 = \overline{\text{I0.0}}$, $\chi_2 = \text{I0.0} \,\&\, \overline{\text{I0.1}}$, $\chi_3 = \overline{\text{I0.0}}$, $\chi_4 = \text{I0.2}$, $\chi_5 = \overline{\text{I0.2}}$, $\chi_6 = 1$, $\chi_7 = 1$. Bu modelde, t3, t4 ve t5 geçişleri sırası ile 1 saniye, 0.7 saniye ve 1.5 saniye zaman gecikmelerine sahip zamanlı geçişlerdir. p7 ve p8 mevkileri sırası ile seçme selenoidinin *kapalı* ve *açık* durumunu göstermektedir. Benzer olarak p9 ve p10 mevkileri de birleştirme selenoidinin *kapalı* ve *açık* durumlarını simgelemektedir. p1, p3 ve p5 mevkilerinde bulunan jetonlar sırası ile seçme alanının, halka biriktirme kolunun ve birleştirme noktasının boş olduğunu simgelemektedir. p2, p4 ve p6 mevkilerinde jeton bulunması ise sırası ile seçme alanında, halka biriktirme kolunda ve birleştirme noktasında plastik halka olduğunu göstermektedir.

Başlangıçta, her iki selenoid te çalışmamakta yani kapalı durumdadırlar ve üretim sisteminin herhangi bir yerinde plastik halka yoktur. Seçme alanında plastik halka yoksa (p1 mevkisinde jeton varsa) ve seçme alanında plastik halka olduğu algılanmışsa ($\chi_2 = \text{I0.0} \,\&\, \overline{\text{I0.1}}$), t2 geçişi tetiklenir ve p1 mevkisindeki jeton alınarak p2 mevkisine jeton depolanır. Bu seçme alanında bir plastik halka olduğu anlamına gelmektedir. Seçme alanındaki halka ya seçme selenoidi kullanılarak halka toplama koluna gidebilir, ya da kullanılmayan halka toplama alanına gitmek üzere seçme alnından çıkabilir. Eğer halka seçme alanında bir halka varsa, selenoid çalışıyorsa, halka biriktirme kolu boş ise ve halka, seçme alanından ayrılmışsa ($\chi_3 = \overline{\text{I0.0}}$) zamanlı geçiş t3 bir saniye gecikme ile tetiklenmektedir. Bu geçişin tetiklenmesi ile p_2 mevkisindeki jeton p4 mevkisine aktarılmaktadır. Bu, seçme alanındaki halkanın halka toplama koluna aktarıldığını ve bu işlemin bir saniye sürdüğü anlamına gelmektedir. Eğer p2 ve p7 mevkilerinde birer jeton bulunuyorsa ve $\chi_1 = \overline{\text{I0.0}}$ tetikleme koşulu gerçekleşmişse t1 geçişi tetiklenir ve p2 mevkisindeki jeton p1 mevkisine aktarılır. Bu, seçme alanındaki halkanın, kullanılmayan halka toplama alanına gitmek üzere seçme alanından çıktığı anlamına gelir.

Eğer halka toplama kolunda halka varsa (p4 mevkisinde jeton varsa), birleştirme alanında halka yoksa (p5 mevkisinde jeton varsa), birleştirme selenoidi açıksa ve halkanın yokluğu algılanmışsa ($\chi_4 = \text{I0.2}$) t4 zamanlı geçişi 0.7 saniye zaman gecikmesi ile tetiklenmektedir. Bu, halka toplama kolundaki halkanın, döner selenoid

kullanılarak, birleştirme noktasına aktarıldığı ve bu işlemin 0.7 saniye sürdüğü anlamına gelmektedir.

Eğer birleştirme noktasında bir halka varsa (p6 mevkisinde bir jeton varsa), metal çubuk ile halka birleşerek birleştirme noktasından halka ayrılmış ise ($\chi_5 = \overline{I0.2}$), t5 zamanlı geçişi 1.5 saniye gecikme ile tetiklenmektedir. Bu geçişin tetiklenmesi ile p6 mevkisindeki jeton p5 mevkisine aktarılmaktadır. Bu işlem, sistemde bir halka ile bir metal çubuğun birleştiğini ve sistemi terk ettiğini göstermektedir. Şekil 3.9'dan da görüldüğü üzere Q0.0 ve Q0.1 aksiyonları p8 ve p10 mevkilerine atanmıştır. p8 mevkisinde jeton bulunması ile Q0.0 aksiyonu gerçekleşmekte ve itici selonid çalışmaktadır. Benzer olarak p10 mevkisinde bir adet jeton bulunması ile Q0.1 aksiyonu gerçekleşmekte ve döner selenoid çalıştırılmaktadır.

Endüstriyel üretim sisteminin istenilen çalışma özellikleri (Spesifikasyonlar) aşağıda sıralanmıştır.

1. Seçme selenodi sadece seçme alanında bir halka olduğunda ve halka toplama kolunda boş yer olduğunda çalışacaktır (SPEC1).
2. Birleştirme selenodi sadece halka toplama kolunda bir halka olduğunda ve birleştirme noktası boş olduğunda çalışacaktır (SPEC2).

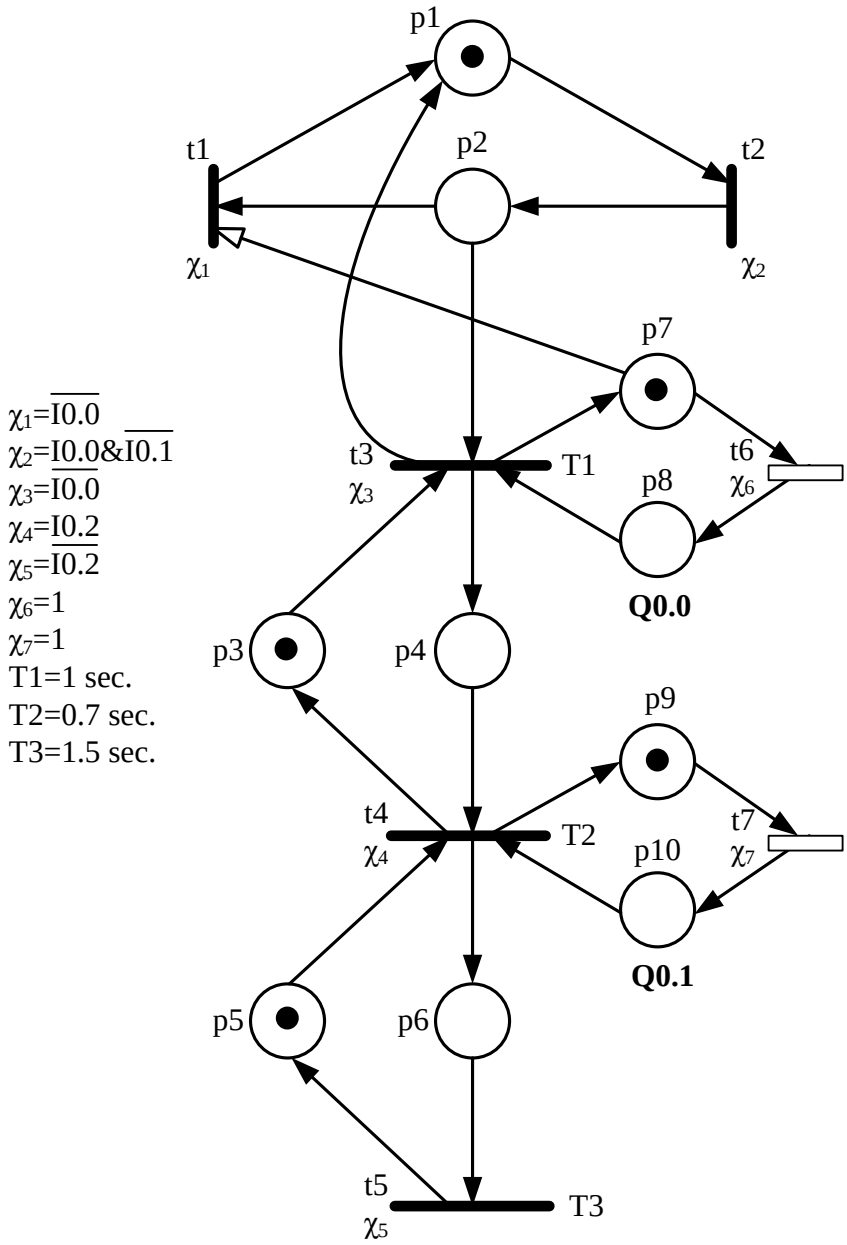
2. Adım: Şekil 3.9'da görülen APN modeli sınırlıdır. Bu modelde bulunan mevkilerin tümünün jeton kapasitesi bir(1)'dir.

3. Adım: Model basit bir model olup, sadeleştirmeye gerek duyulmamaktadır.

4. Adım: Şekil 3.9'da görülen Petri ağı modelindeki her bir mevkinin aşağıda verilen TCT olay kodlaması dikkate alınarak elde edilen tampon modelleri Şekil 3.10'da görülmektedir.

TCT olay etiketleri:

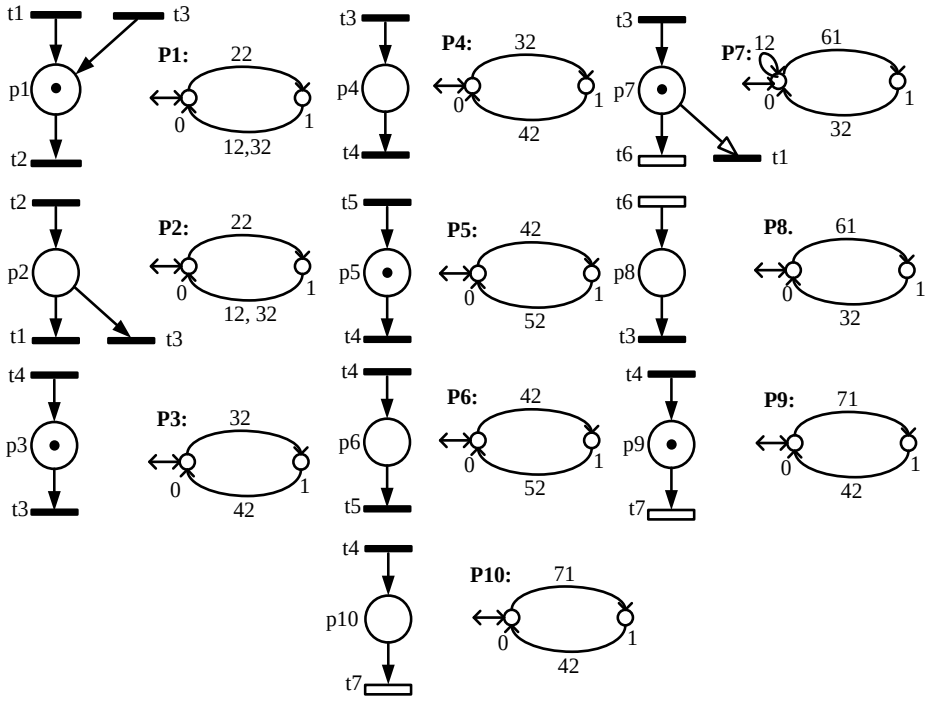
APN:	t1	t2	t3	t4	t5	t6	t7
TCT:	12	22	32	42	52	61	71



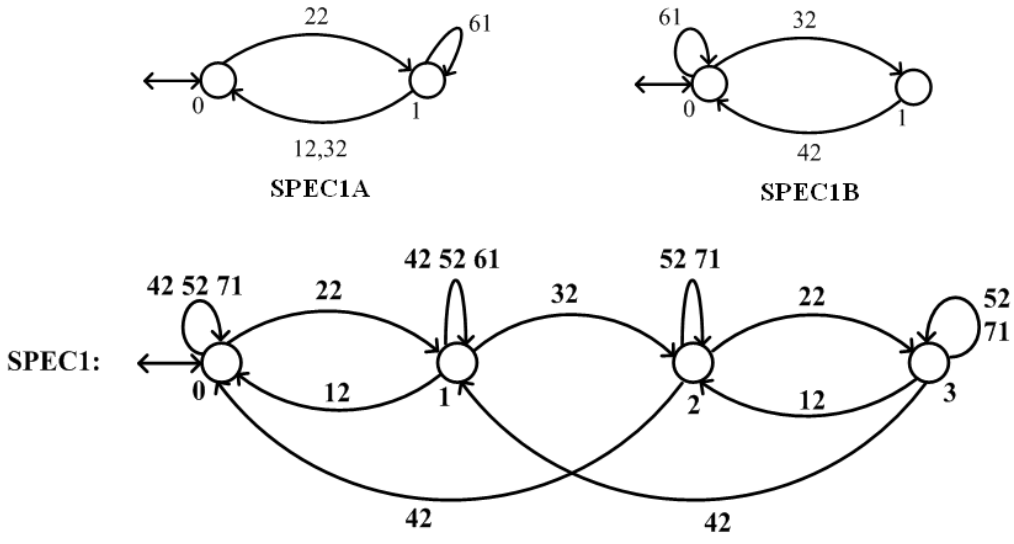
Şekil 3.9 Deneysel endüstriyel üretim sisteminin kontrol edilmemiş APN modeli

Deneysel endüstriyel üretim sisteminde iki adet spesifikasyon mevcuttur. Birinci spesifikasyonu iki kısma ayırmak mümkündür. Bunlar "Seçme selenodini seçme alanında bir halka olduğunda çalıştır" ve "Seçme selenodini halka toplama kolunda boş yer olduğunda çalıştır". Bu kısımlar, sırası ile SPEC1A ve SPEC1B isimli otomatalarla modellenir. Bu iki otomatanın senkron kompozisyon işlemiyle birleştirilmesi ile birinci özelliğe ait otomata modeli elde edilir. Bu birleştirme işleminin ardından, sistem modelinde bulunan ancak SPEC1 tarafından kısıtlanmayan olayların (52 ve 71 etiketli olaylar) SPEC1'in tüm durumlarına özdöngü (Selfloop) olarak eklenmesi

gerekmektedir. Birinci spesifikasyona ait otomata modelleri Şekil 3.11'de görülmektedir.



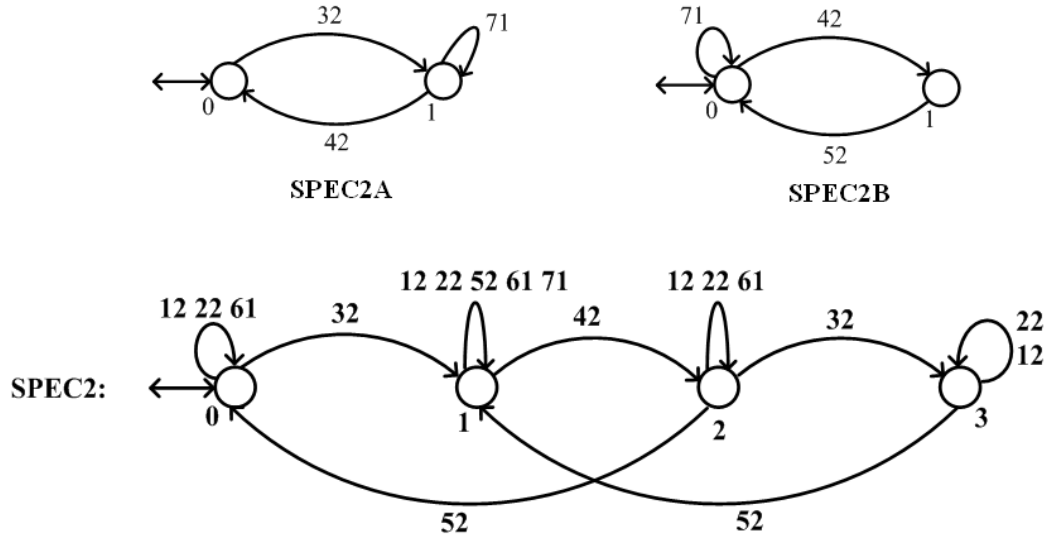
Şekil 3.10 APN modelindeki mevkilerin tampon karşılıkları



Şekil 3.11 Birinci spesifikasyona ait otomata modelleri

İkinci spesifikasyonda benzer olarak iki kısma ayrılabilir. Bunlar, "Birleştirme selenodini halka toplama kolunda bir halka olduğunda çalıştır" ve "Birleştirme

selenodini birleştirme noktası boş olduğunda çalıştır". Birinci spesifikasyonda olduğu gibi bu iki parçaya ait otomata modelleri birleştirilir. Bu işlemin ardından sistem modelinde bulunan ancak SPEC2 tarafından kısıtlanmayan olaylar (12, 22, 61 etiketli olaylar) SPEC2'nin tüm mevkilerine özdöngü olarak eklenir. İkinci spesifikasyona ait otomata modelleri ise Şekil 3.12'de görülmektedir.



Şekil 3.12 İkinci spesifikasyona ait otomata modelleri

5. Adım: Bu sistem için TCT yazılımı kullanılarak RW gözeticisi aşağıdaki gibi elde edilmiştir. İlk olarak, *Create(.)* komutu ile tampon modelleri ve spesifikasyonlara ait otomata modelleri aşağıdaki gibi tanımlanır:

P1 = Create(M1,[mark 0],[tran [0,22,1],[1,12,0],[1,32,0]]) (2,3)
P2 = Create(M2,[mark 0],[tran [0,22,1],[1,12,0],[1,32,0]]) (2,3)
P3 = Create(M3,[mark 0],[tran [0,32,1],[1,42,0]]) (2,2)
P4 = Create(M4,[mark 0],[tran [0,32,1],[1,42,0]]) (2,2)
P5 = Create(M5,[mark 0],[tran [0,42,1],[1,52,0]]) (2,2)
P6 = Create(M6,[mark 0],[tran [0,42,1],[1,52,0]]) (2,2)
P7 = Create(M7,[mark 0],[tran [0,12,0],[0,61,1],[1,32,0]]) (2,3)
P8 = Create(M8,[mark 0],[tran [0,61,1],[1,32,0]]) (2,2)
P9 = Create(M9,[mark 0],[tran [0,71,1],[1,42,0]]) (2,2)
P10 = Create(M10,[mark 0],[tran [0,71,1],[1,42,0]]) (2,2)

PLANT, PLANT=P1||P2||P3||P4||P5||P6||P7||P8||P9||P10 senkron kompozisyonu ile aşağıdaki gibi elde edilmiştir. PLANT modelinde 32 adet durum ve 80 adet geçiş bulunmaktadır

PLANT= Sync(P1,P2) (2,3)
PLANT = Sync(PLANT,P3) (4,7)

PLANT = Sync(PLANT,P4) (4,7)
PLANT = Sync(PLANT,P5) (8,16)
PLANT = Sync(PLANT,P6) (8,16)
PLANT = Sync(PLANT,P7) (16,34)
PLANT = Sync(PLANT,P8) (16,34)
PLANT = Sync(PLANT,P9) (32,80)
PLANT = Sync(PLANT,P10) (32,80)

PLANT # states: 32 state set: 0 ... 31 initial state: 0 marker states: 0
vocal states: none # transitions: 80

transitions:

[0, 22, 1] [0, 61, 2] [0, 71, 3] [1, 12, 0] [1, 61, 4] [1, 71, 5] [2, 22, 4] [2, 71, 6] [3, 22, 5]
[3, 61, 6] [4, 32, 7] [4, 71, 8] [5, 12, 3] [5, 61, 8] [6, 22, 8] [7, 22, 9] [7, 61, 10] [7, 71, 11]
[8, 32, 11] [9, 12, 7] [9, 61, 12] [9, 71, 13] [10, 22, 12] [10, 71, 14] [11, 22, 13]
[11, 42, 15] [11, 61, 14] [12, 71, 16] [13, 12, 11] [13, 42, 17] [13, 61, 16] [14, 22, 16]
[14, 42, 18] [15, 22, 17] [15, 52, 0] [15, 61, 18] [15, 71, 19] [16, 42, 20] [17, 12, 15]
[17, 52, 1] [17, 61, 20] [17, 71, 21] [18, 22, 20] [18, 52, 2] [18, 71, 22] [19, 22, 21]
[19, 52, 3] [19, 61, 22] [20, 32, 23] [20, 52, 4] [20, 71, 24] [21, 12, 19] [21, 52, 5]
[21, 61, 24] [22, 22, 24] [22, 52, 6] [23, 22, 25] [23, 52, 7] [23, 61, 26] [23, 71, 27]
[24, 32, 27] [24, 52, 8] [25, 12, 23] [25, 52, 9] [25, 61, 28] [25, 71, 29] [26, 22, 28]
[26, 52, 10] [26, 71, 30] [27, 22, 29] [27, 52, 11] [27, 61, 30] [28, 52, 12] [28, 71, 31]
[29, 12, 27] [29, 52, 13] [29, 61, 31] [30, 22, 31] [30, 52, 14] [31, 52, 16]

Spesifikasyonları elde etmek için yapılan TCT işlemleri aşağıda görülmektedir.

SPEC1A = Create(SPEC1A,[mark 0],[tran [0,22,1],[1,12,0],[1,32,0],[1,61,1]]) (2,4)
SPEC1B = Create(SPEC1B,[mark 0],[tran [0,32,1],[0,61,0],[1,42,0]]) (2,3)
SPEC1 = Sync(SPEC1A,SPEC1B) (4,8) Blocked_events = None
SPEC1 = Selfloop(SPEC1,[52,71]) (4,16)
SPEC2A = Create(SPEC2A,[mark 0],[tran [0,32,1],[1,42,0],[1,71,1]]) (2,3)
SPEC2B = Create(SPEC2B,[mark 0],[tran [0,42,1],[0,71,0],[1,52,0]]) (2,3)
SPEC2 = Sync(SPEC2A,SPEC2B) (4,6) Blocked_events = None
SPEC = Meet(SPEC1,SPEC2) (8,20)

TCT yazılımı kullanılarak gözeticinin sentezlenebilmesi için Supcon( ) ve Supdat( ) komutları kullanılmaktadır. Gözeticinin sentezlenmesi için aşağıdaki işlemler yapılmıştır.

SUPER = Supcon(PLANT, SPEC) (12,23)
CDAT = Condat(PLANT, SUPER) Controllable.

Bu işlemlerin ardından elde edilen gözetici ve kontrol verisi aşağıdaki gibidir.

SUPER # states:12 states set:0 … 11 initial satete: 0
Marker states: 0 vocal states: none # transitions: 23

transitions:
[0,22,1] [1,12,0] [1,61,2] [2,32,3] [3,22,4] [3,71,5] [4,12,3] [4,71,6] [5,22,6] [5,42,7] [6,12,5] [6,42,8] [7,22,8] [7,52,0] [8,12,7] [8,52,1] [8,61,9] [9,32,10] [9,52,2] [10,22,11] [10,52,3] [11,12,10] [11,52,4]

CDAT
Control data:
0: 61 71 1: 71 2: 71 3: 61 4: 61 5: 61 6: 61 7: 61 71
8: 71 9: 71 10: 61 71 11: 61 71

Kontrol verisi yukarıda görülen liste şeklindedir. Bu listede gözeticinin hangi durumunda hangi olayların yasaklanacağı görülmektedir.

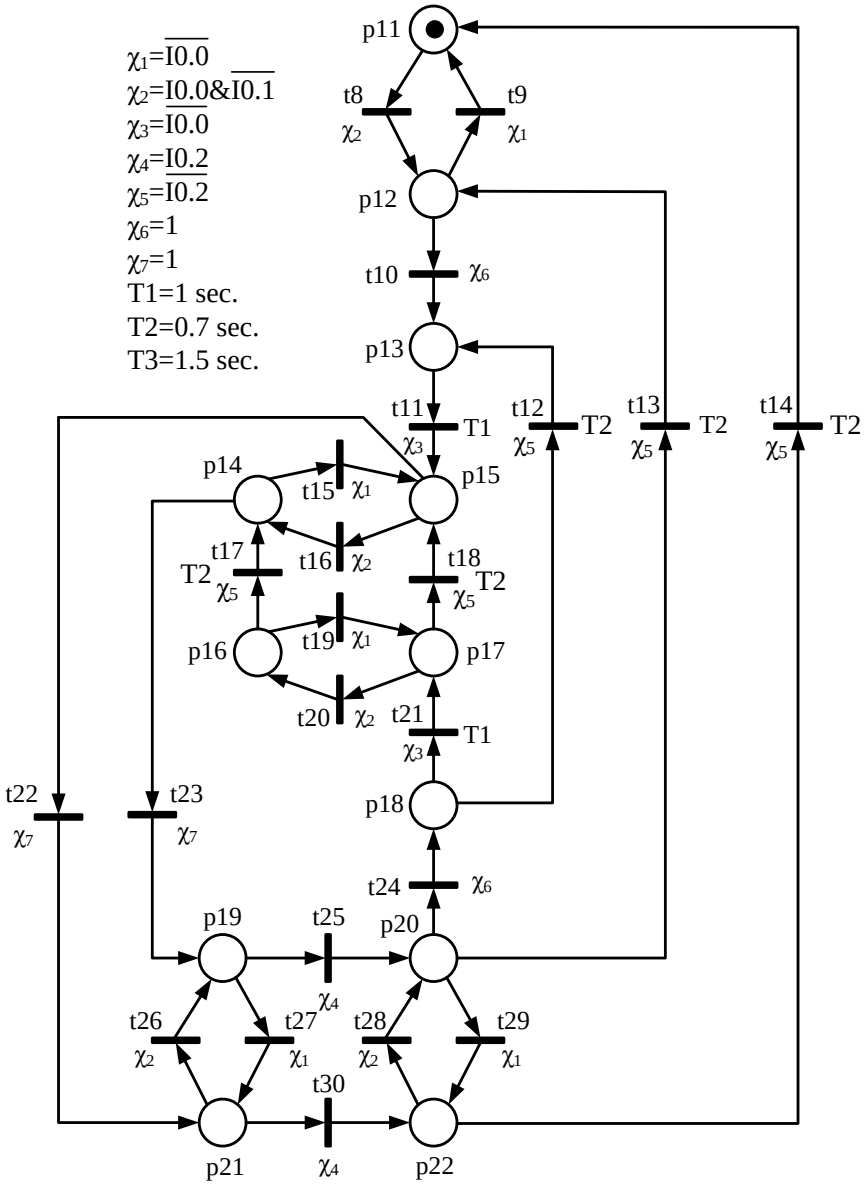
6. Adım: TCT yazılımı kullanılarak elde edilen otomata modeli Şekil 3.13'te görüldüğü gibi auto-nete çevrilmiştir.

7. Adım: Bu işlemlerin ardından elde edilen auto-net yapısındaki gözetici ve Petri ağı yapısındaki sistem için TCT ile elde edilen yasaklanacak olay listesindeki olaylar, yasaklama oku kullanılarak gerçekleştirilir. Yukarıda hesaplanan kontrol verisi, P = {p11, p12, ..., p22} mevkilerinden t6 ve t7 geçişlerine *In(p11,t6), In(p11,t7), In(p12,t7), In(p13,t7), In(p14,t6), In(p15,t6), In(p16,t6), In(p16,t7), In(p17,t6), In(p17, t7), In(p18,t7), In(p19,t6), In(p20,t7), In(p21,t6), In(p22,t6),* ve *In(p22,t7)* yasaklama okları kullanılarak gerçekleştirilmektedir. Sistemin APN modeli, auto-net olarak gözetici ve yasaklama oku olarak gerçekleştirilen kontrol poliçesini içeren kapalı çevrim melez kontrol sistemi modeli Şekil 3.14'te görülmektedir.

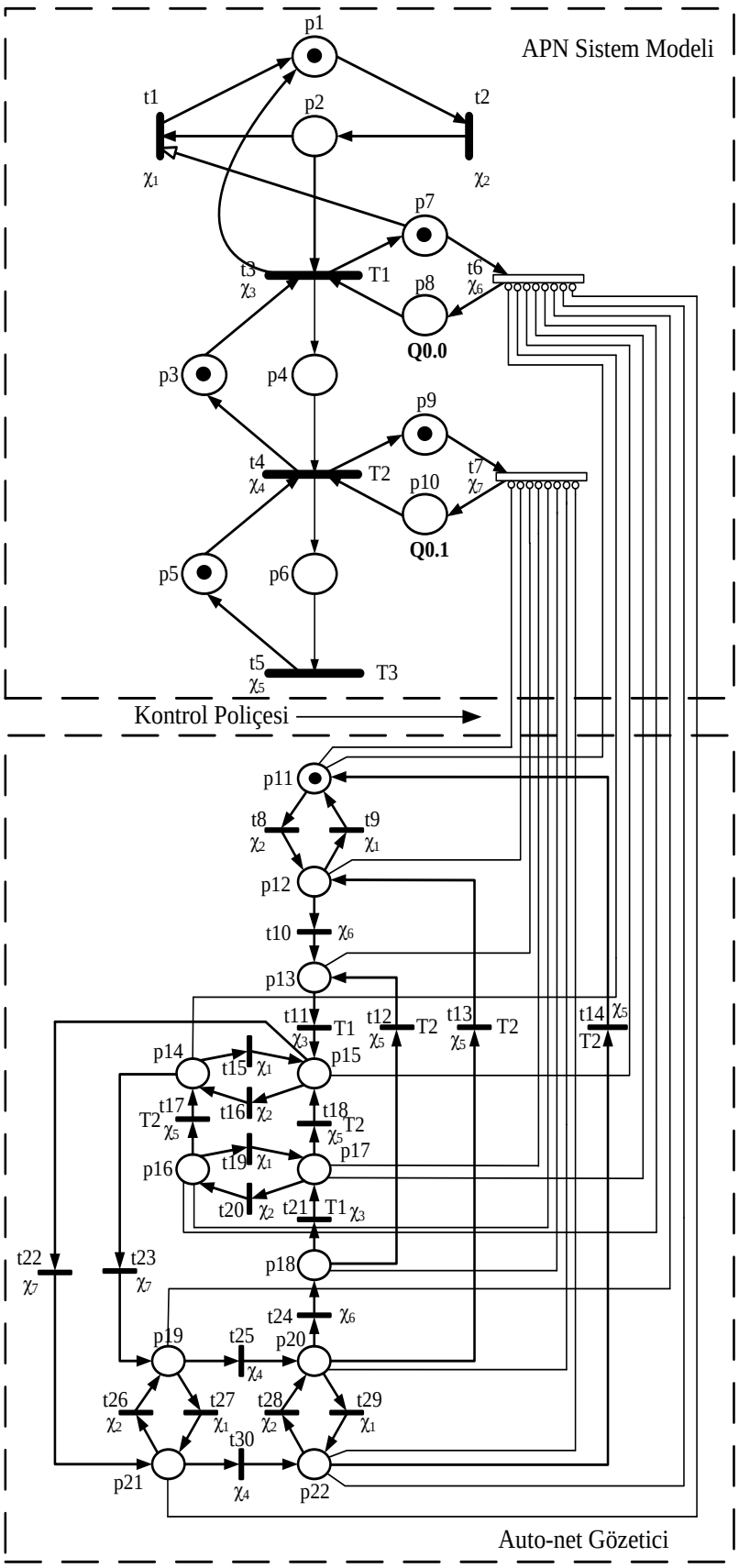
8. Adım: Deneysel endüstriyel üretim sistemi için elde edilen melez kontrol sistemine ait PLC merdiven diyagramı kodu Şekil 3.15'te görülmektedir. Birinci uygulamada PLC kodunun daha anlaşılır olması için kullanılması gereken hafıza birimleri için modelde bulunan mevki ve durum isimleri sembolik olarak kullanılmıştır. Bu uygulamada ise doğrudan PLC'nin kendi sembolleri verilmiştir. Her bir mevkiye bir hafıza biti (marker) atanmıştır. Zamanlı geçişler için ise On Delay Timer yani düz zaman rölesi kullanılmıştır. Hafıza hafıza bitleri M0.1, M0.2, M0.3, M0.4, M0.5, M0.6, M0.7, M1.0, M1.1 ve M1.2 APN modelindeki P=(p1, p2, ..., p10) mevkileri için atanmıştır. Benzer olarak hafıza bitleri M2.0, M2.1, M2.2, M2.3, M2.4, M2.5, M2.6, M2.7, M3.0, M3.1, M3.2, ve M3.3 auto-net olarak tasarlanan gözetici mevkileri P=

(p11, p12, …, p22) için kullanılmıştır. T1 (1 s. zaman gecikmeli), T2 (0.7 s. zaman gecikmeli) ve T3 (1.5 s. zaman gecikmeli) düz zaman röleleri sırası ile APN modelindeki t3, t4 ve t5 geçişlerine atanmıştır.

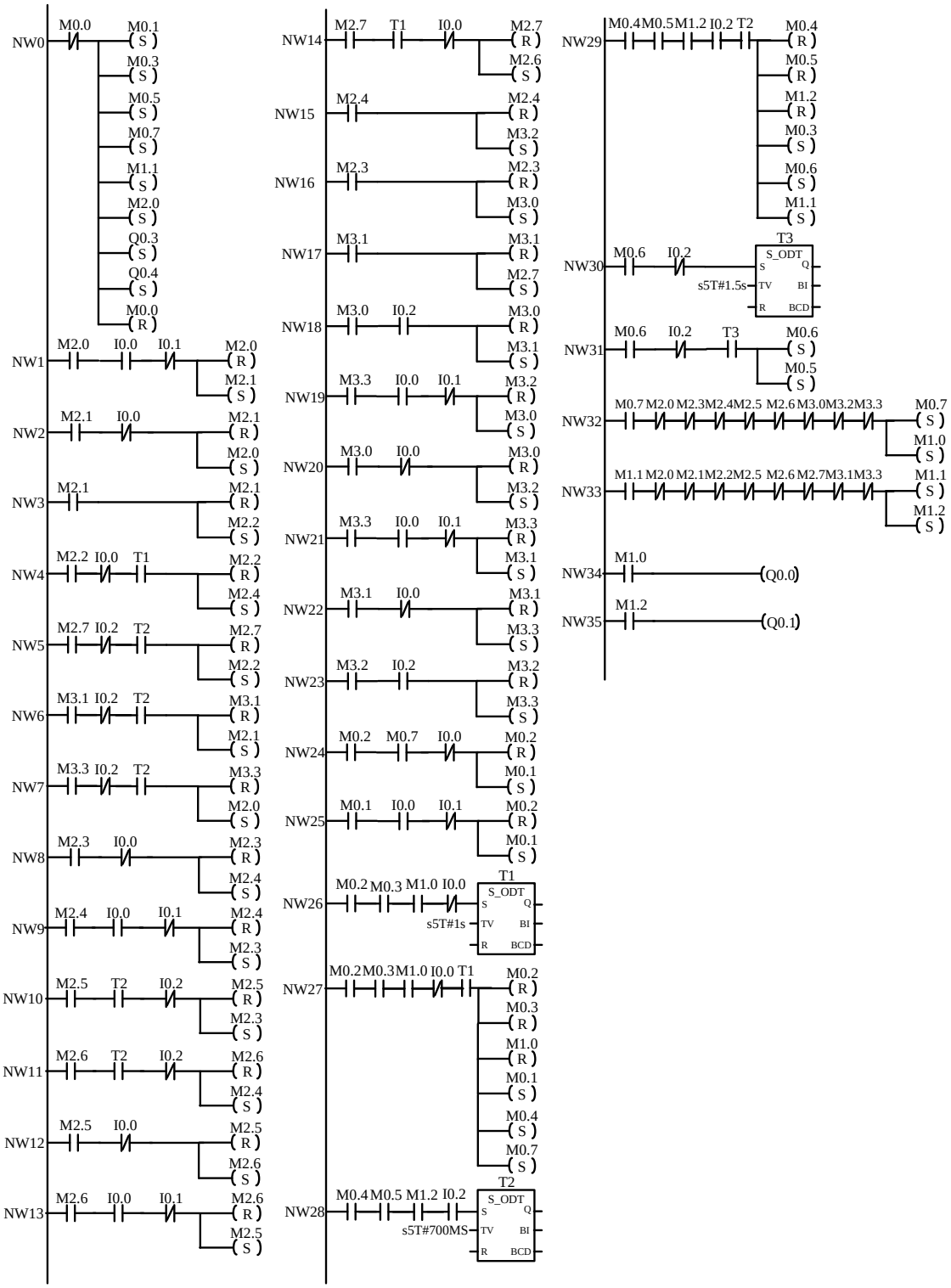
Şekil 3.15'te görülen kodun ilk basamağı (NW0) başlangıç atamaları için kullanılmıştır. Bu basamakta, başlangıçta içerisinde jeton bulunan mevkiler için atanan bitler set edilmektedir. Ayrıca bu basamakta alt ve üst konveyör motorları da çalıştırılmaktadır. NW1-NW23 arasında auto-net formundaki gözetici gerçekleştirilmiştir. NW24-NW33 arasında sistemin APN modeli gerçekleştirilmiştir. Son olarak NW34 ve NW35'te p8 ve p10 mevkilerine atanmış olan çıkış aksiyonları gerçekleştirilmiştir.



Şekil 3.13 Gözeticinin auto-net modeli



Şekil 3.14 Kapalı çevrim melez kontrol sistemi modeli



Şekil 3.15 Deneysel endüstriyel üretim sistemini kontrol etmek için elde edilen PLC merdiven diyagramı kodu

Şekil 3.15'te görülen PLC merdiven diyagramı kodu Siemens S7-300 PLC'ye yüklenerek, sistemin spesifikasyonlarda belirtilen çalışmayı başarılı bir şekilde gösterdiği gözlemlenmiştir.

3. 4 Sonuç

Ayrık olay sistemlerinin melez kontrolü konusunda [52]'de önerilen yöntemin gerçek sistemlerin uygulanmasında kullanılabilirliği iki örnek ile gösterilmiştir. Bunlardan ilki, Niğde Üniversitesi Elektrik -Elektronik Mühendisliği Bölümü Laboratuarında bulunan uzaktan kumandalı otomatik kapı modelinin kontrolüdür. Bu sistem adı geçen yöntem kullanılarak başarılı bir biçimde kontrol edilmiştir. Bu çalışmadan elde edilen sonuçlar, [66]'da yayınlanmıştır.

İkinci uygulama olarak, [52]'de önerilen melez yöntemin deneysel endüstriyel üretim sistemi kullanılarak gerçek sistemlerin kontrolünde uygulanabilirliği gösterilmiştir. Bu çalışmadan elde edilen sonuçlar ise [67]'de yayınlanmıştır.

Bu uygulamalarda, Petri ağlarının genişletilmiş biçimi olan otomasyon Petri ağları (APN) [60] kullanılmıştır. AOS'lerin melez kontrolü konusunda [52]'de önerilen yöntemde elde edilen gözeticilerin indirgenmesiyle hesaplanan, indirgenmiş gözeticinin auto-net olarak kullanılması öngörülmektedir. Ancak, indirgenmiş gözeticilerin PLC ile gerçekleştirilmeleri esnasında çığ etkisi probleminden dolayı, elde edilen gözeticiyi gerçekleştirmek için yazılan PLC merdiven diyagramı kodu doğru çalışmamaktadır. Çığ etkisi probleminin çözümüne yönelik gerçekleştirilen çalışmalar tezin bir sonraki bölümünde incelenmektedir.

BÖLÜM IV

ÇIĞ ETKİSİ PROBLEMİNİN ETKİN ÇÖZÜMÜ VE İNDİRGENMİŞ MELEZ DENETLEYİCİLERİN PLC İLE GERÇEKLEŞTİRİLMESİ

4.1 Çığ Etkisi Probleminin Çözümü İçin Etkin Bir Yöntem (AEEM)

Sonlu durum makinelerine (Finite State Machines - FSM) ve formal dillere dayalı Gözetimli Kontrol Teorisi (Supervisory Control Theory - SCT), ayrık olay sistemlerinin (AOS) kontrolü için önerilmiş genel bir yöntemdir [3-9]. SCT'de amaç, kontrol edilmemiş sistem modeli ve kontrol spesifikasyonlarını kullanarak, sistemi spesifikasyonların belirlediği gibi çalışmaya zorlayacak bir gözetici sentezlemektir. Programlanabilir Lojik Denetleyiciler (Programmable Lojik Controller – PLC), kontrol görevlerini yerine getirmek için modern endüstriyel sistemlerde yaygın olarak kullanılmaktadır. Otomata biçimindeki gözeticilerin PLC ile gerçekleştirilmesi pek çok araştırmacı tarafından incelenmiştir [11, 57, 64, 70, 78-86]. SCT'nin esnek üretim hücrelerine uygulanması ve PLC ile gerçekleştirilmesi [12, 64, 79-81] çalışmalarda rapor edilmiştir. De Querioz ve Curry ile Vieira ve diğerleri tarafından yapılan çalışmalarda lokal modüler gözeticiler ve bunların PLC ile gerçekleştirilmesi incelenmiştir [82, 87]. Merdiven diyagramı kodu (Ladder Logic Diagram - LLD) ve ardışık fonksiyon grafikleri (Sequantial Function Charts SFC) kullanılarak gözeticilerin gerçekleştirilmesi sırası ile [64] ve [87] nolu çalışmalarda gösterilmiştir. Bu çalışmaların temel özelliği, kontrol sisteminin gerçek giriş/çıkış sinyalleri ile teorik gözetici arasında ara yüz olarak kullanılmasıdır. Sonlu durum makinelerinin PLC merdiven diyagramı koduna dönüşümü Liu ve Darabi ile Moniruzzaman ve Gohari tarafından yapılan çalışmalarda önerilmiştir [84, 85]. Sonlu durum makinelerinin PLC ile gerçekleştirilmesi Manesis ve diğerleri tarafından da incelenmiştir [86]. Son çalışmaların birinde gözeticilerin merdiven diyagramı olarak gerçekleştirilmesi için yeni bir metot önerilmiştir [70, 88]. Bu metot, temel olarak aksiyonların (çıkış sinyallerinin) gözeticinin ilgili durumlarına atanmasını ifade etmektedir.

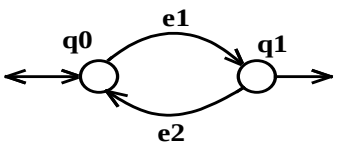
Otomata biçimindeki gözeticilerin PLC ile gerçekleştirilmesinde karşılaşılan muhtemel problemler ve çözümleri [78]'de açıklanmıştır. Programın aynı PLC tarama döngüsü içerisinde rastgele sayıda duruma sıçraması olayı çığ etkisi (avalanche effect) olarak

adlandırılan önemli bir problemdir. [78]'de bu problemin basitçe çözümü olarak PLC kodundaki basamakların tersten yazılması önerilmektedir. Ancak, bu genel bir çözüm değildir. [89]'da bazı durumlarda bu yöntemin hataya sebep olduğu gösterilmiştir. Çığ etkisi probleminin çözümü için, otomataların lojik olarak ifade edilmesine dayanan bir metot Hasdemir ve diğerleri tarafından önerilmiştir. Bu çalışmada önerilen yöntemin dezavantajı, gözeticinin her bir durumunu gerçekleştirmek için iki bitlik hafıza elemanına ihtiyaç duyulmasıdır. Bu husus, çok sayıda duruma sahip gözeticilerin özellikle sınırlı sayıda hafıza alanı bulunduran PLC'ler ile gerçekleştirilmesinde büyük problem teşkil etmektedir. Bu problemin çözümü için, bu çalışmada, "çığ etkisi önleme metodu (Avalanche Effect Elimination Method - AEEM)" adında etkin bir yöntem önerilmiştir [77]. Önerilen metodun uygulanabilirliği örneklerle gösterilmiştir.

4.1.1 Çığ etkisi problemi

Gözetimli Kontrol Teorisi (Supervisory Control Theory – SCT) sürekli sistemler için önerilen kontrol yapılarının ayrık olay sistemlerine uyarlanmış biçimidir [3-9]. SCT'de sistem davranışını modellemek için otomata teorisi ve formal diller kullanılır. Sistem ve sistemin kontrol edilmiş davranışını belirleyen spesifikasyonlar otomata olarak modellenir. SCT kullanılarak, sistemi istenilen biçimde kontrol eden bir Ramadge-Wonham (RW) gözeticisi elde edilir. Ayrık olay sistemleri olayların anlık oluşumu ile çalışmaktadır. SCT ile hesaplanan gözetici otomata formundadır. Otomata ve SCT hakkında bazı bilgiler tezin 2. bölümünde bulunabilir.

Şekil 4.1'de iki duruma sahip basit bir otomata modeli görülmektedir. Bu modelde daireler ile gösterilen q0 ve q1 olarak adlandırılmış iki durum mevcuttur. Bu durumlardan q0 durumuna bağlı olan çift yönlü ok, bu durumun hem başlangıç hem de işaretli (marker) durum olduğunu göstermektedir. q1 durumuna bağlı çıkış oku ise bu durumun sadece işaretli (marker) durum olduğunu göstermektedir. Durumları birbirine bağlayan etiketli oklar ise durumlar arası geçişleri göstermektedir. Geçişlerdeki e1 ve e2 etiketleri olayları ifade etmektedir.

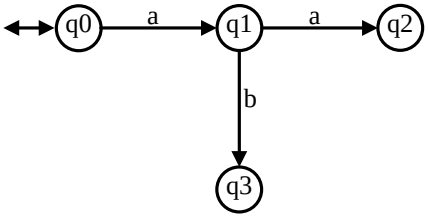


Şekil 4.1 Basit bir otomata modeli

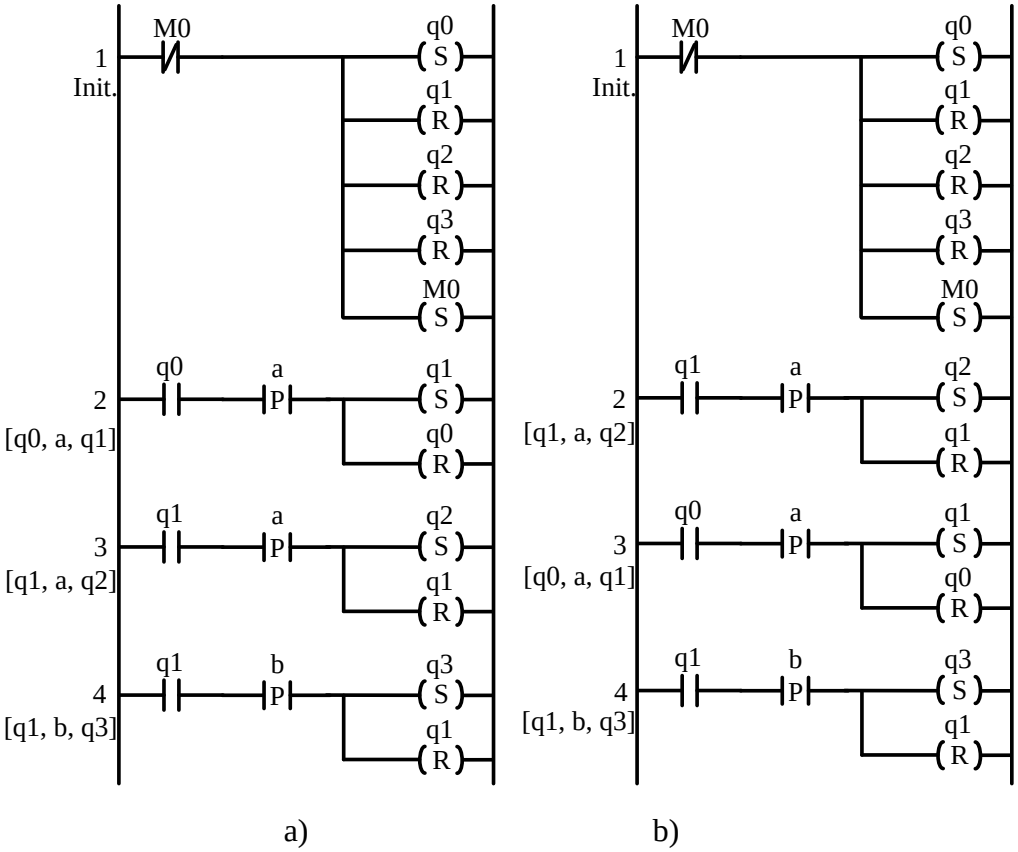
Otomatalar PLC ile gerçekleştirilirken, durumlar Boole değişkenleriyle, olaylar ise sistemden alınan sinyallerin yükselen ya da düşen kenarları ile ifade edilebilir. Olayların bu şekilde tanımlanması için yükselen kenar algılama kontağı --|P|-- (pozitif geçiş), ve düşen kenar algılama kontağı --|N|-- (negatif geçiş) kullanılmaktadır. PLC programlama dili olan merdiven diyagramı için bazı IEC 61131-3 LD (Ladder Diagram) standart sembolleri Ek-A'da görülmektedir.

Gözeticilerin PLC ile gerçekleştirilmesi için gereken merdiven diyagramı koduna dönüşüm basit bir işlemdir. Literatürde bu işlem için önerilmiş iyi bilinen bir metot mevcuttur. Aşağıda kısaca bu çalışmada kullanılacak olan yöntem açıklanmaktadır. Gözeticinin her bir durumuna karşılık gelen bir Boole değişkeni (bir hafıza biti) atanır. Başlangıçta, RW gözeticisinin başlangıç durumunu simgeleyen hafıza biti SET, diğer durumlarını simgeleyen hafıza bitleri ise RESET edilir. Sonra, her bir geçiş ayrı bir basamakta SET ve RESET komutları kullanılarak gerçekleştirilir. Aynı tarama döngüsünde, yükselen veya düşen kenarlarla ilişkilendirilmiş olayların dikkate alınmaması ile öngörülemez sayıda duruma atlama gerçekleşebilir. Bu problem "çığ etkisi" olarak adlandırılmaktadır ve Boole değişkenlerinin eş zamanlı olarak değerlendirilmesinin sonucudur [78]. Örneğin Şekil 4.2'de görülen otomata modelini inceleyelim. Otomata, "a" olayının gerçekleşmesi ile q0 durumundan q1 durumuna geçiş yapmaktadır. Sonra, q1 durumundan q2 durumuna geçiş ancak "a" olayının yeniden oluşumu ile mümkündür. Eğer yeni bir "a" olayı oluşmadan "b" olayı meydana gelirse otomata q1 durumundan q3 durumuna geçecektir. Buna göre, bu otomata "aa" ve "ab" dizilerini üretebilmektedir [78]. Bu otomatanın Şekil 4.3.(a)'da görülen merdiven diyagramı kodu, bu otomatayı çığ etkisinden dolayı tam olarak gerçekleştirememektedir. Bu kodda çığ etkisinden dolayı "a" sinyalinin yükselen kenarıyla q0'dan q2'ye doğrudan bir geçiş olmaktadır. Bu problemin giderilmesi için basit bir çözüm, basamakların yerlerinin değiştirilmesidir [78]. Bu örnek için, Şekil 4.3.(a)'da görülen kodun 2. ve 3. basamaklarının yerlerinin değiştirilmesi ile elde edilen ve Şekil 4.3.(b)'de görülen kod sorunu çözmektedir. Bu metot aşırı derecede problemin tipine bağlıdır ve bazı durumlarda çalışmaz. Örnek olarak Şekil 4.4'te görülen basit otomata modeli verilebilir [89]. Şekilden görülebildiği gibi "a" olayı çığ etkisine sebep olmaktadır. Eğer otomata q0 durumundayken "a" olayı gerçekleşirse otomata aynı tarama döngüsünde q0 durumundan q1 durumuna ve tekrar q1 durumundan q0

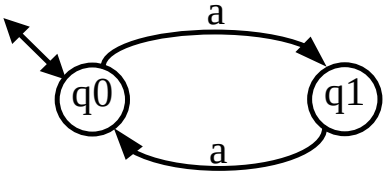
durumuna geçiş yapar. Yukarıda anlatılan metot bu otomatadaki çığ etkisi problemini çözememektedir [89].



Şekil 4.2 ‘aa’ ve ‘ab’ dizilerini üreten basit bir otomata modeli [78]



Şekil 4.3 Şekil 4.2’de görülen otomata modelinin merdiven diyagramı kodu gerçekleştirmesi a) Sadece bir ‘a’ oluşumunu sağlayan merdiven diyagramı b) ‘aa’ ve ‘ab’ dizilerin oluşumunu sağlayan merdiven diyagramı



Şekil 4.4 Çığ etkisi problemine sahip örnek bir otomata modeli [89]

4.1.2 Bir otomatadaki çığ etkisi probleminin belirlenmesi

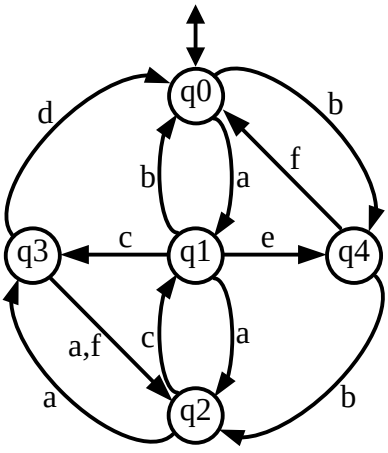
RW gözeticilerinin hesaplanması için pek çok yazılım mevcuttur. Tez kapsamında, RW gözeticileri TCT [68] yazılımı kullanılarak hesaplanmaktadır. TCT yazılımında bir otomata ".ADS" dosyası olarak tanımlanmaktadır. Pozitif tamsayılar TCT yazılımında durum (0, …, 2000000) ve olay (0, …, 999) etiketi olarak kullanılmaktadır. "0" etiketi her zaman başlangıç durumunu göstermektedir. TCT yazılımında, bir otomatanın geçişleri *"kaynak durumu olay etiketi hedef durum"* formatına girilmektedir. Örneğin 2 0 1 girişi, otomatada 2 ile etiketlenmiş durumdan 1 ile etiketlenmiş duruma 0 etiketine sahip bir geçiş olduğunu göstermektedir. Özel bir durum olarak 1 0 1 girişi 1 ile etiketlinmiş durumun 0 ile etiketlenmiş bir öz döngüsü (Selfloop) olduğu anlamına gelmektedir. Tüm geçişler dosyasının sonuna liste olarak girilmektedir. Bu geçişler bir matris gibi düşünülebilir. Bu matrisin satır sayısı tüm geçişlerin sayısına eşittir. Matrisin sütun sayısı ise 3 tür. İlk sütun kaynak durumları, ikinci sütun olay etiketlerini, son sütun ise hedef durumları içermektedir. Şekil 4.5'te örnek bir otomata modeli ve Şekil 4.6'da de bu otomatanın TCT girişi olan "TEST.ADS" dosyası görülmektedir. Bu otomata için kullanılan olay ve durum etiketlerinin kodlaması aşağıdaki gibidir.

Örnek otomata için olay kodlaması:

Otomata:	a	b	c	d	e	f
TCT:	1	2	3	4	5	6

Örnek otomata için durum kodlaması:

Otomata:	q0	q1	q2	q3	q4
TCT:	0	1	2	3	4



Şekil 4.5 'a', 'b', and 'c' olaylarının çığ etkisine sebep olduğu örnek bir otomata modeli

```
# CTCT ADS auto-generated

TEST

State size (State set will be (0,1....,size-1)):
# <-- Enter state size, in range 0 to 2000000, on line below.
5

Marker states:
# <-- Enter marker states, one per line.
# To mark all states, enter *.
# If no marker states, leave line blank.
# End marker list with blank line.
0

Vocal states:
# <-- Enter vocal output states, one per line.
# Format: State  Vocal_Output.  Vocal_Output in range 10 to 99.
# Example: 0 10
# If no vocal states, leave line blank.
# End vocal list with blank line.

Transitions:
# <-- Enter transition triple, one per line.
# Format: Exit_(Source)_State  Transition_Label  Entrance_(Target)_State.
# Transition_Label in range 0 to 999.
# Example: 2 0 1 (for transition labeled 0 from state 2 to state 1).
0 1 1
0 2 4
1 3 3
1 2 0
1 1 2
1 5 4
2 3 1
2 1 3
3 1 2
3 6 2
4 2 2
4 6 0
3 4 0
```

Şekil 4.6 Örnek bir .ADS soyası formatı: "TEST.ADS"

Çığ etkisi problemine sebep olan olayları belirlemek için aşağıda görülen tarama algoritması önerilmiştir [77].

***Algorithm Av_effect_detector**:*
***Girdi:** .ADS dosya formatında bir otomata. Otomata .ADS dosya formatında geçiş geçiş tanımlanır. Geçişler "kaynak durumu olay etiketi hedef durum" formatında girilmektedir. Bu formatın her bir satırı geçiş olarak adlandırılır. Geçişlerin tümü [n_trans x 3] boyutlu bir dizi olarak kabul edilebilir.*

1) *[Tanımlama] Crr_event: güncel olayı bildiren değişken, Crr_s_sts: güncel kaynak durumunu bildiren değişken. Crr_t_sts: güncel hedef durumu bildiren değişken All_Events:tüm olaylar dizisi, Av_effect_list: çığ etkisine sebep olan olaylar listesi, n_trans: geçişlerin sayısı, num_ev:olayların sayısı, S_state:kaynak durum dizisi, T_state:hedef durum dizisi. Events: olay dizisi.*
2) *[Başlangıç] Crr_event=0, i=0, j=0, k=0, index=0;*
3) *Crr_event=All_Events[i] olay listesinden bir olay alınır.*
4) *if (Crr_event==Events[j])*
   then Crr_s_sts=S_state[j] ve Crr_t_sts=T_state[j];

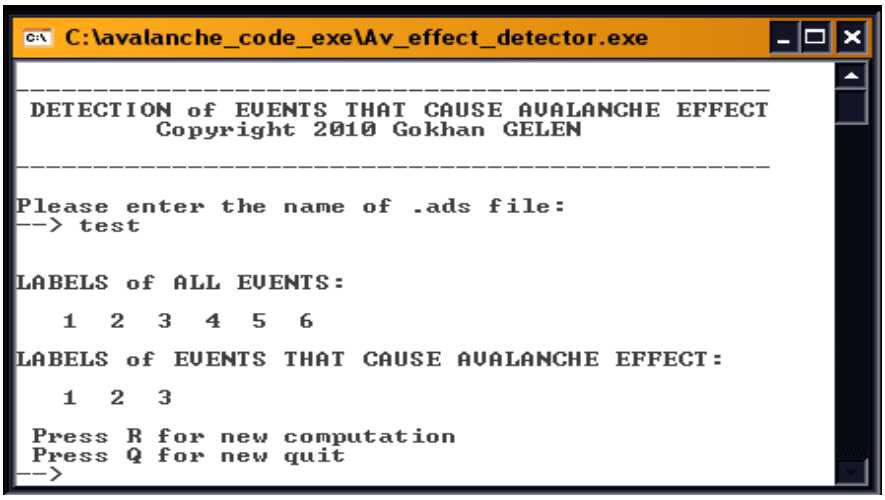
5) *(Eğer aynı olay için hedef durumuna eşit bir kaynak durumu veya kaynak durumuna eşit bir hedef durumu var ise bu olay çığ etkisine sebep olmaktadır.)*

```
if(((Crr_s_sts==T_state[k])||(Crr_t_sts==S_state[k]))
&&(Crr_event==Events[k]))
then
        Av_effect_list[index]=All_Events[i]
      index=index+1
        go to step 9
    end if.
```

6) *k=k+1;*
7) *if (k<n_trans) goto Step 5;*
8) *j=j+1 and if (j<n_trans) goto Step 4;*
9) *k=0, j=0, i=i+1, and if (i<num_ev) goto Step 3;*

***Çıktı**: Av_effect_list: çığ etkisine sebep olan olayların listesi*
end Algorithm Av_effect_detector.

Bu algoritmanın temel özelliği aynı olaya sahip iki geçiş bulmaktır. Bu algoritma C codu olarak gerçekleştirilmiş, derlenerek "Av_effect_detector.exe" isimli bir .exe dosyası elde edilmiştir. Algoritmaya ait C kaynak kodu Ek-B'de verilmiştir. Bu kodda .ADS giriş dosyası okunmakta ve matrise çevrilmektedir. Hedef ve kaynak durumu aynı olan geçişler yani özdöngüler (Selfloop) dikkate alınmamaktadır. Çünkü bu geçişler durum değişimine sebep olmadığından çığ etkisine sebep olmazlar. Eğer girdide çığ etkisine sebep olan olay bulunursa bu olaylar liste halinde ekrana ve "logfile.txt" isimli log dosyasına yazılmaktadır. "Av_effect_detector.exe" programı çalıştırılıp TEST.ADS isimli giriş dosyası çağrıldığında programın ürettiği çıktının ekran görüntüsü Şekil 4.7'de görülmektedir. Bu görüntüdeki listede "1, 2 ve 3" etiketli olayların yani Şekil 4.5'te görülen otomatanın "a, b ve c" etiketli olaylarının çığ etkisine sebep olduğu program tarafından bulunmaktadır.



Şekil 4.7 "Av_effect_detector.exe" programının "TEST.ADS" girişi için ekran görüntüsü

4.1.3 Çığ etkisi probleminin önlenmesi

Çığ etkisi problemi, programın aynı PLC tarama döngüsünde öngörülemez sayıda duruma atlamasıyla meydana gelir. Olayların oluşumu merdiven diyagramı kodunda, sinyallerin yükselen veya düşen kenarları ile elde edilmektedir. Bir sinyalin yükselen kenarı, ardı ardına gerçekleşen iki tarama döngüsündeki sinyal karşılaştırılarak bulunur. Eğer sinyal bir önceki taramada "OFF" ise ve şu anki taramada "ON" ise sinyalin yükselen kenarı algılanmış olur [78]. Bu çalışmada otomataların merdiven diyagramı kodu olarak gerçekleştirilmesinde karşılaşılan çığ etkisi probleminin çözümü için etkin bir yöntem önerilmiştir. Yöntem sadece çığ etkisine sebep olan olaylara uygulanmaktadır. Çığ etkisini önlemek için önerilen etkin metod (Avalanche Effect Elimination Method - AEEM) aşağıdaki adımları içerir [77]:

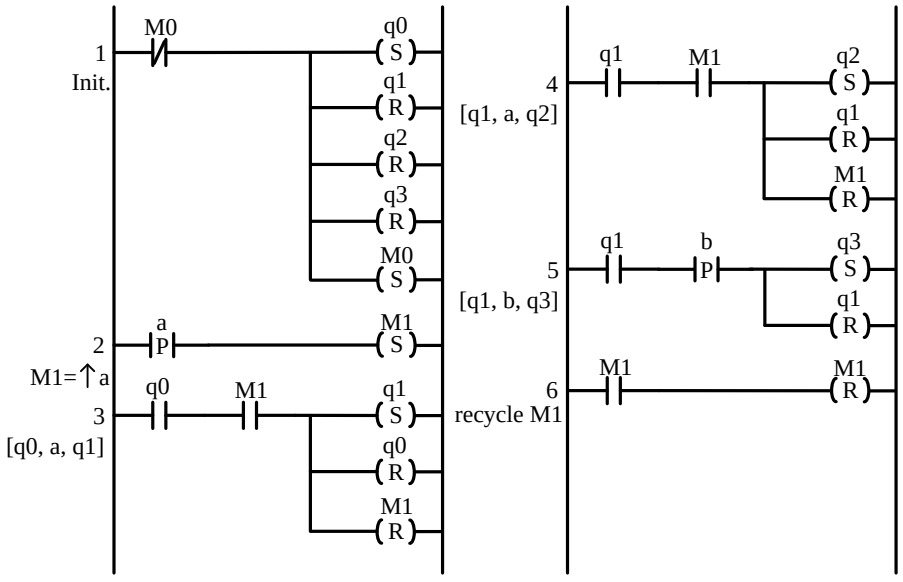
1. Merdiven diyagramı kodunun başlangıç basamağından sonra, çığ etkisine sebep olan her bir olayın oluşumuyla, ilgili bir *Mi* (i=1, 2, …, n; n: çığ etkisine sebep olan olay sayısı) Boole değişkeni SET edilir.
2. Çığ etkisine sebep olan olayın oluşumuyla gerçekleşebilecek bütün geçişlerin merdiven diyagramı kodunda, olayla ilişkilendirilen *Mi* hafıza biti RESET yapılır. (i=1, 2, …, n)
3. Çığ etkisine sebep olan olayın oluşup geçişin gerçekleşmediği durumlarda olay oluşumunun iptal edilmesi için merdiven diyagramı kodu sonunda *Mi* hafıza biti RESET yapılır. (i=1, 2, …, n)

Bu yöntemde, çığ etkisine sebep olan bir olayın oluşumu (sinyalin yükselen veya düşen kenarı) bir *Mi* hafıza bitine atanır. *Mi* set olduğunda bu biti çığ etkisinin görüldüğü geçişlerden biri kullanır. Bu bit her kullanımda ilgili basamakta resetlenir. Eğer bu bit kullanılmamış ise programın sonunda otomatik olarak resetlenir. Bu yöntem çığ etkisi probleminin çözümü için etkin ve kolay bir yöntemdir.

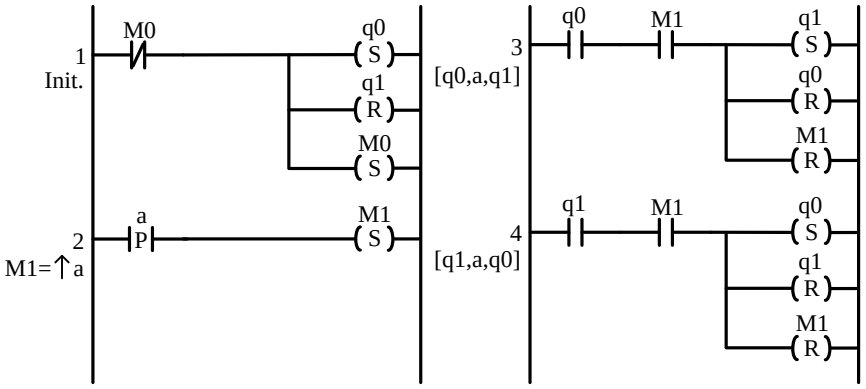
Bu yöntemin nasıl uygulanabildiğini incelemek için Şekil 4.2'de görülen otomata modelini inceleyelim. Bu modelde 'a' olayı çığ etkisine sebep olmaktadır. Bu nedenle AEEM yöntemi sadece bu olaya uygulanacaktır. Bu otomata için AEEM metodu uygulandıktan sonra elde edilen merdiven diyagramı kodu Şekil 4.8'de görülmektedir. Bu kodda, ilk basamakta q0 set edilip diğer durumlar (q1, q2 ve q3) resetlenerek

otomata başlangıç durumuna getirilmiştir. ‘a’ olayının oluşması ile M1 hafıza biti 2. basamakta set edilmektedir. 3, 4 ve 5. basamaklarda sırası ile [q0,a,q1], [q1,a,q2] ve [q1,b,q3] geçişleri gerçekleştirilmiştir. ‘a’ olayının sebep olduğu çığ etkisi probleminin giderilmesi için geçişin gerçekleşmesiyle M1 değişkeni 3. ve 4. basamaklarda resetlenmektedir. Çığ etkisine sebep olan ‘a’ olayı oluşumunun kaydedildiği M1 hafıza biti, eğer herhangi bir geçişin gerçekleşmesinde kullanılmamışsa merdiven diyagramının son basamağında (6. basamak) reset yapılır.

Başka bir örnek olarak Şekil 4,4’te görülen otomatada ‘a’ olayı yine çığ etkisine sebep olmaktadır. Bu otomata için AEEM metodu uygulandıktan sonra elde edilen merdiven diyagramı kodu Şekil 4.9’da görülmektedir. Bu kodun ilk satırı q0’ı set ve q1’i reset ederek otomatayı başlangıç konumuna getirir. ‘a’ olayının gerçekleşmesi ile M1 set edilir. 3. ve 4. basamaklarda sırası ile [q0,a,q1] ve [q1,a,q0] geçişleri gerçekleştirilmiştir. Modelde bulunan iki geçişte de ‘a’ olayı çığ etkisine sebep olduğundan M1 biti her iki geçişin sonunda resetlenmiştir. Bu nedenle bu özel örnek için M1 bitinin kodun sonunda bir daha resetlenmesine gerek yoktur. Çünkü otomatanın bütün geçişleri bu olay oluşumuyla gerçekleşecektir. ‘a’ olayının her oluşumunda otomatada mutlaka bir geçiş gerçekleşecektir ve bundan dolayı M1 hafıza biti resetlenecektir.



Şekil 4.8 Şekil 4.2’de görülen otomata modeli için
AEEM kullanılarak elde edilen merdiven diyagramı kodu



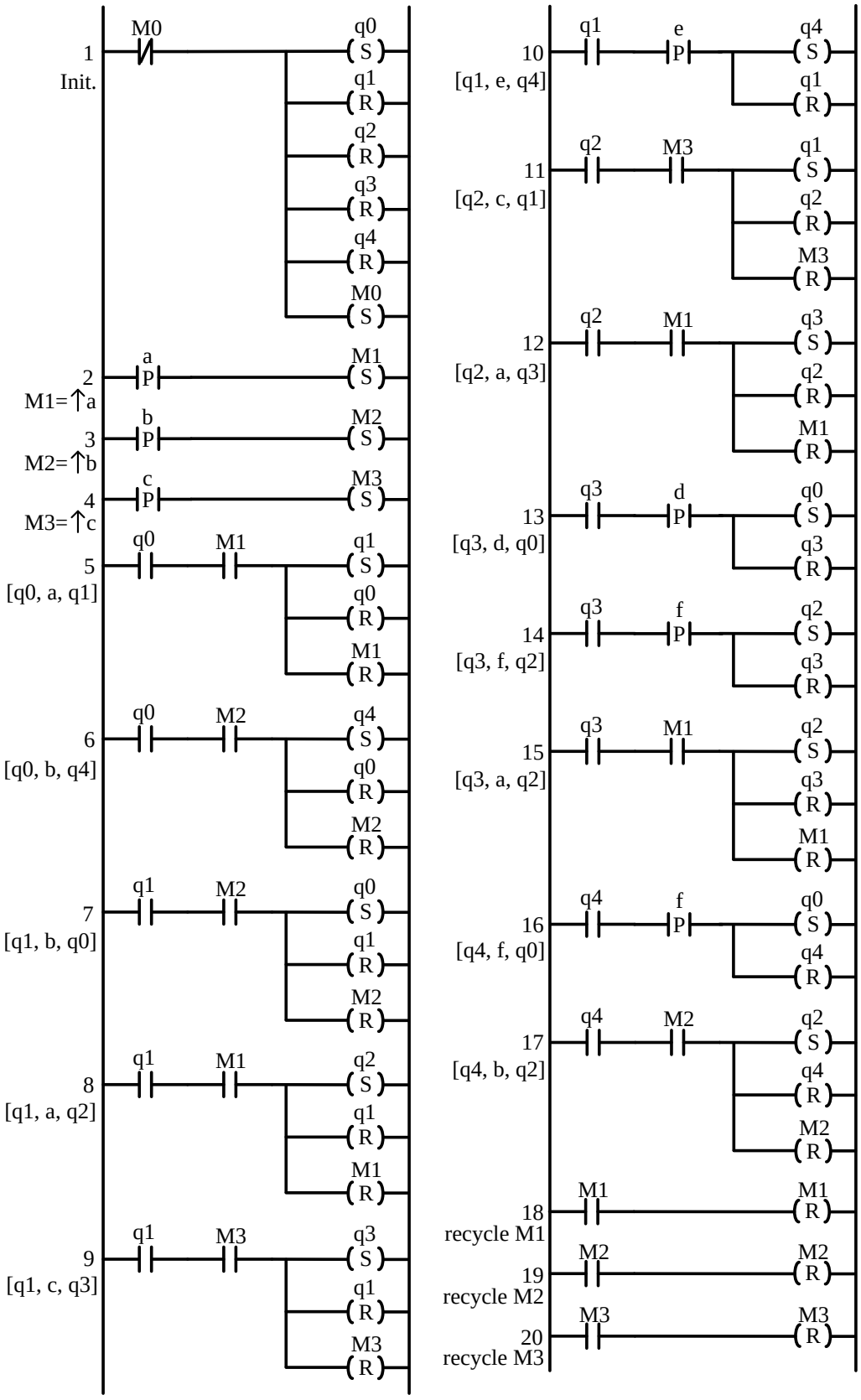
Şekil 4.9 Şekil 4.4'te görülen otomata modeli için
AEEM kullanılarak elde edilen merdiven diyagramı kodu.

4.1.4 Uygulama

Farklı bir uygulama örneği olarak Şekil 4,5'te görülen beş durum ve 13 geçişten oluşan otomata modeli dikkate alınmaktadır. Bu modelde etiketleri "a, b, c, d, e ve f" olan toplam altı adet olay bulunmaktadır. "Av_effect_detector" programı kullanılarak "a, b ve c" olaylarının çığ etkisine sebep olduğu bulunmuştur (Şekil 4.7). Bu yüzden, PLC merdiven diyagramı kodu ile gerçekleştirme için önerilen AEEM yalnızca bu olaylara uygulanacaktır.

Bu otomatanın PLC merdiven diyagramı kodu gerçekleştirmesi Şekil 4.10'da görülmektedir. İlk basamakta, q0 durumu set ve diğer durumlar (q1, q2, q3 ve q4) reset edilerek otomata başlangıç durumuna getirilmektedir. 'a' olayının gerçekleşmesi ile (benzer olarak 'b' ve 'c') M1 hafıza biti (benzer olarak M2 ve M3 hafıza bitleri) 2. (3. ve 4.) basamakta set edilmektedir. 5. ve 17. basamaklar arasında 13 geçişin tamamı gerçekleştirilmiştir. Çığ etkisine sebep olan 'a' olayı için M1 hafıza biti 5. 8. 12. ve 15. basamakların sonunda resetlenmektedir. Çığ etkisine sebep olan 'b' olayı için M2 hafıza biti 6., 7. ve 17. basamakların sonunda resetlenmektedir. Çığ etkisine sebep olan 'c' olayı için M3 hafıza biti 9. ve 11. basamakların sonunda resetlenmektedir. Eğer çığ etkisine sebep olan olayların oluşumu hiçbir geçişi gerçekleştirmezse, bu olayların tekrar algılanabilmesi için ilgili hafıza bitleri kodun sonunda resetlenir. Çığ etkisine sebep olan 'a' olayı için M1 hafıza biti 18. basamakta, 'b' olayı için M2 hafıza biti 19. basamakta ve 'c' olayı için M3 hafıza biti 20. basamakta resetlenmektedir. Bu çalışmada görülen tüm merdiven diyagramı kodları Siemens S7-300 PLC kullanılarak

gerçekleştirilmiş ve yapılan denemeler sonucunda istenildiği gibi doğru bir şekilde çalıştığı gözlemlenmiştir.



Şekil 4.10 Şekil 4.5'te görülen otomata modelinin
AEEM kullanılarak elde edilen merdiven diyagramı kodu

4.2 AOS'lerin İndirgenmiş Gözetici ile Melez Kontrolü

[52]'de ortaya konulan yöntem kullanılarak indirgenmiş gözetici içeren melez denetleyici tasarım aşamaları aşağıdaki gibi önerilmektedir:

1. Sistemin kontrol edilmemiş davranışı APN olarak modellenir. Yasaklanmış durum spesifikasyonları belirlenir.
2. APN modelinin sınırlı (bounded) olduğu ve mevkilerinin jeton kapasiteleri belirlenir.
3. Eğer gerekliyse Petri ağı sadeleştirme kuralları ile APN sadeleştirilir.
4. APN modelindeki mevkiler [52]'de önerilen yöntem kullanılarak eşdeğer tampon (buffer) modellerine dönüştürülür. Spesifikasyonlar otomata olarak modellenir.
5. Gözetimli Kontrol Teorisi ile gözetici (SUPER) ve kontrol verisi (CDAT) hesaplanır.
6. TCT yazılımında bulunan *Supreduce(.)* komutu ile indirgenmiş gözetici RED_SUP ve *Condat(.)* komutu ile RED_CDAT kontrol verisi hesaplanır.
7. RED_SUP, auto-nete (AUTONET) dönüştürülür.
8. AUTONET, PLANT'a yasaklama okları ile bağlanarak kapalı çevrim melez model elde edilir.
9. Elde edilen kapalı çevrim melez kontrol sistemi bir PLC ile gerçekleştirilir.

İkinci adımdaki sınırlılık hesaplaması standart PN yazılımları kullanılarak ya da basit durumlarda gözlemle yapılabilir. Üçüncü adımdaki, sadeleştirme kuralları literatürde mevcut olup, PN modelinde canlılık, sınırlılık ve tersinirlik gibi özellikleri değiştirmemektedir.

Dördüncü adımda gerekli olan dönüşüm işlemleri için temel bir yaklaşım [52]'de mevcuttur. Bu yaklaşıma göre, eğer bir mevkinin jeton kapasitesi b ise, bu mevki, kapasitesi $b+1$ olan bir tampon modeline dönüştürülmektedir. Bu temel düşünce ışığında, PN modellerinde karşılaşılabilecek çeşitli durum ve modeller için dönüşümler konusunda tez kapsamında araştırmalar yapılmıştır. Elde edilen sonuçlar bir sonraki bölümde ve [90]'da sunulmuştur.

Beşinci adımdaki hesaplamalar TCT yazılımı [68] kullanılarak yapılabilir. Tezin ikinci bölümünde TCT yazılımının kullanımı detaylı olarak açıklanmıştır. APN modelinde bulunan mevkiler (p1, p2, p3, ..., pN) bu mevkilerin tampon karşılıklarının otomata modelleri P1, P2, P3, ..., PN olmak üzere beşinci adımda yapılacak işlemler aşağıdaki gibi özetlenebilir:

5.1 AOS'lerin Gözetimli Kontrol Teorisi yardımıyla kontrolünde yapılacak olan ilk işlem sistemi oluşturan tüm alt bileşenlerin (P1, P2, …, PN) otomata olarak oluşturulması ve bu otomata modellerinin TCT yazılımında *Create(.)* komutu kullanılarak gerçekleştirilmesi işlemidir:

```
P1=Create(P1)
P2=Create(P2)
      .
      .
PN=Create(PN)
```

5.2 Elde edilen alt bileşenlerin senkron kompozisyonu ile sistemin kontrol edilmemiş modeli oluşturulur (PLANT=P1||P2||…||PN). Bu işlem TCT yazılımında *Sync(.)* komutu kullanılarak gerçekleştirilir:

```
PLANT=Sync(P1,P2)
PLANT=Sync(PLANT,P3)
      .
      .
PLANT=Sync(PLANT,PN)
```

5.3 Gözetici tasarımında yapılacak diğer işlemde, sistemin çalışmasında istenilen özelliklerin her biri yani spesifikasyonlar *Create(.)* komutuyla ayrı ayrı elde edilir (SPEC1, SPEC2, ..., SPECM):

```
SPEC1=Create(SPEC1)
SPEC2=Create(SPEC2)
      .
      .
SPECM=Create(SPECM)
```

Daha sonra elde edilen her bir spesifikasyon modeline kontrol edilmemiş sistem modeli PLANT'ta bulunan fakat ilgili spesifikasyon tarafından kısıtlanmayan olaylar ilgili spesifikasyon modelinin tüm durumlarına *Selfloop(.)* komutu kullanılarak özdöngü olarak eklenir:

```
SPEC1=Selfloop(SPEC1, PLANT'ta bulunan diğer olaylar)
```

SPEC2=Selfloop(SPEC2, PLANT'ta bulunan diğer olaylar)

.

.

SPECM=Selfloop(SPECM, PLANT'ta bulunan diğer olaylar)

Son olarak elde edilen tüm spesifikasyon modelleri (SPEC1, SPEC2, ..., SPECM) TCT yazılımında bulunan *Meet(.)* komutu kullanılarak sistemin çalışma şeklini ifade eden tek bir spesifikasyon (SPEC) modeli olarak elde edilir:

SPEC=Meet(SPEC1, SPEC2)
SPEC=Meet(SPEC, SPEC3)

.

.

SPEC=Meet(SPEC, SPECM)

5.4 *SUP=Supcon(PLANT,SPEC)* komutu kullanılarak sistemin kontrol edilmiş şeklini ifade eden gözetici modeli SUP elde edilir.

5.5 Bir önceki adımda elde edilen SUP gözetici modeline göre daha az duruma sahip olan indirgenmiş gözetici modelinin TCT yazılımıyla elde edilmesi için *DAT=Condat(PLANT, SUP)* komutu kullanılarak DAT kontrol bilgisi elde edilir.

Altıncı adımda, TCT yazılımında bulunan gözetici durum indirgeme komutu kullanılarak yekpare gözeticiden indirgenmiş gözetici elde edilir. Bu işlem aşağıdaki gibi yapılır.

RED_SUP=Supreduce(PLANT,CDAT)
RED_CDAT=Condat(PLANT,RED_SUP)

Yedinci adımda, bir önceki adımda elde edilen otomata formundaki indirgenmiş gözetici, PN'lere benzer bir yapı olan auto-net formuna dönüştürülür. Bu işlemde, gözetici modelinde bulunan özdöngüler, durumlarda değişikliğe sebep olmadığından ihmal edilir.

Sekizinci adımda, indirgenmiş gözeticiye ait auto-net modeli PLANT'a kontrol verisinde hesaplanan yasaklama okları ile bağlanarak kapalı çevrim melez model elde edilir

Son adımda, indirgenmiş gözeticili kapalı çevrim melez kontrol sistemi PLC ile gerçekleştirilir. Bu gerçekleştirme işleminde, tezin üçüncü bölümünde yer alan yekpare gözeticili sistemin gerçekleştirilmesinden farklı olarak, indirgenmiş gözeticide çığ etkisi problemi olup olmadığı önceki kısımda algoritması açıklanan "Av_effect_detector.exe" programı ile belirlenir. Eğer çığ etkisine sebep olan olay varsa bu olaylara sahip geçişler önceki kısımda önerilen AEEM yöntemi kullanılarak gerçekleştirilir. Gerçekleştirme işlemleri bir sonraki kısımda görülen örnek üzerinde detaylı olarak açıklanmıştır.

4.3 Deneysel Endüstriyel Üretim Sistemi İçin İndirgenmiş Gözeticili Melez Denetleyicilerin Hesaplanması ve PLC ile Gerçekleştirilmesi

Endüstriyel üretim sistemi için Petri ağları ve SCT temelli indirgenmiş gözeticili melez kontrol sistemi aşağıdaki adımlar izlenerek elde edilmiştir.

Yukarıda açıklanan ilk beş adımda yapılması gereken işlemler yekpare gözetici hesaplanırken yerine getirilen ilk beş adımın aynısıdır ve detaylı açıklamaları üçüncü bölümde yer almaktadır. Bu nedenle, ilk beş adım burada yeniden açıklanmayıp işlemlere altıncı adımdan başlanmıştır.

6. Adım: TCT yazılımı kullanılarak elde edilen yekpare gözetici SUPER ve kontrol verisi CDAT dikkate alınarak indirgenmiş gözetici (RED_SUP) ve yeni kontrol verisi (RED_CDAT) aşağıdaki gibi elde edilmiştir.

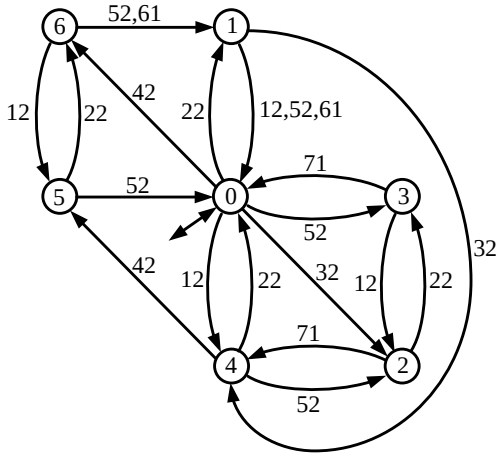
RED_SUP = Supreduce(PLANT,SUPER,SUPDAT) (7,21;slb=7)
RED_CDAT = Condat(PLANT,RED_SUP) Controllable.

Hesaplanan indirgenmiş gözetici (RED_SUP) ve yeni kontrol verisinin (RED_CDAT) detayları aşağıda görülmektedir. İndirgenmiş gözetici RED_SUP'a ait otomata modeli, Şekil 4.11'de görülmektedir.

RED_SUP # states: 7 state set: 0 ... 6 initial state: 0
marker states:0 vocal states: none transitions: 21
transitions:
[0,12, 4] [0,22,1] [0,32,2] [0,42,6] [0,52, 3] [1,12,0] [1,32,4] [1,52,0] [1,61, 0] [2,22,3] [2,71,4] [3,12,2] [3,71, 0] [4,22,0] [4,42,5] [4,52,2] [5,22,6] [5,52,0] [6,12,5] [6,52,1][6,61,1]

RED_CDAT
control data:

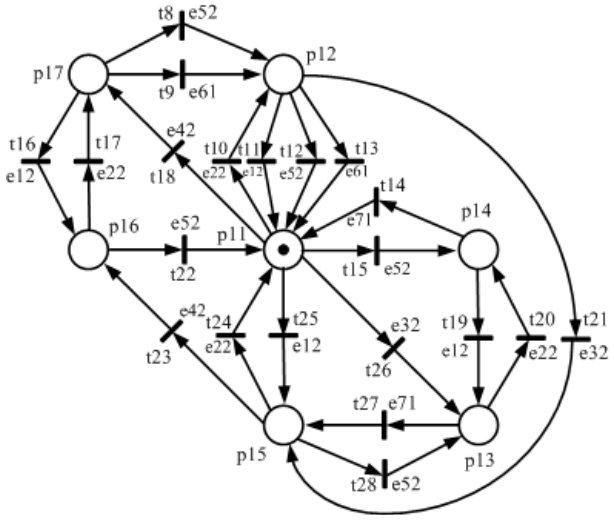
0: 61 71 *1: 71* *2: 61* *3: 61*
4: 61 71 *5: 61 71* *6: 71*



Şekil 4.11 Deneysel endüstriyel üretim sistemi için hesaplanan indirgenmiş gözetici RED_SUP'un otomata modeli

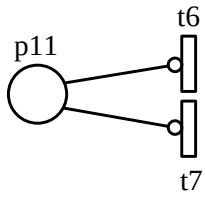
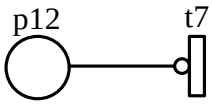
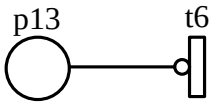
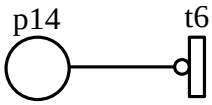
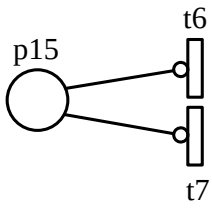
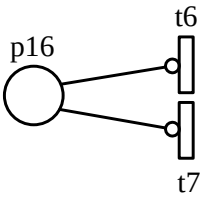
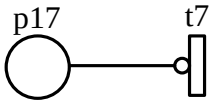
Şekil 4.11'den görülebileceği gibi indirgenmiş gözetici *7 durum ve 21 geçişe* sahiptir. Aynı kontrol spesifikasyonunu sağlayacak yekpare gözetici (SUPER) 12 durum ve 23 geçişe sahiptir.

7. Adım: Şekil 4.11'de görülen indirgenmiş gözeticiye ait otomata modeli Şekil 4.12'de görüldüğü gibi auto-net yapısına dönüştürülmüştür.



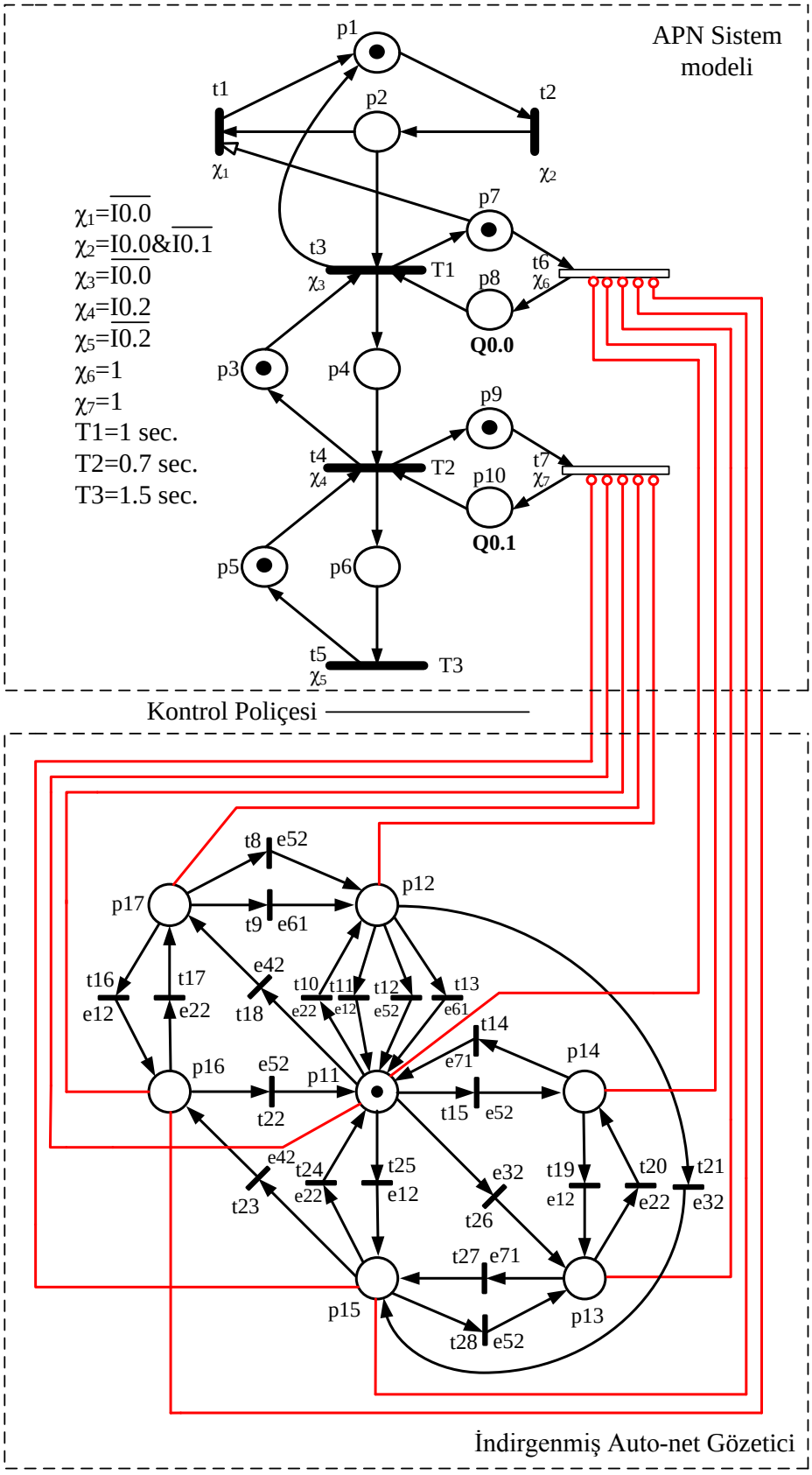
Şekil 4.12 Deneysel endüstriyel üretim sistemi için hesaplanan indirgenmiş gözeticinin auto-net modeli

8. Adım: Bu işlemlerin ardından elde edilen auto-net yapısındaki gözetici ve Petri ağı yapısındaki sistem için TCT ile elde edilen yasaklanacak olay listesindeki olaylar, yasaklama oku kullanılarak gerçekleştirilir. Yukarıda hesaplanan kontrol verisi, P = {p11, p12, ..., p17} mevkilerinden t_6 ve t_7 geçişlerine *In(p11, t6), In(p11, t7), In(p12, t7), In(p13, t6), In(p14, t6), In(p15, t6), In(p15, t7) In(p16, t6), In(p16, t7), In(p17, t7)* yasaklama okları kullanılarak gerçekleştirilmiştir. Kontrol verisinin yasaklama oku karşılıkları Şekil 4.13'te görülmektedir. Sistemin APN modeli, auto-net olarak indirgenmiş gözetici ve yasaklama oku olarak gerçekleştirilen kontrol poliçesini içeren kapalı çevrim melez kontrol sistemi modeli Şekil 4.14'te görülmektedir.

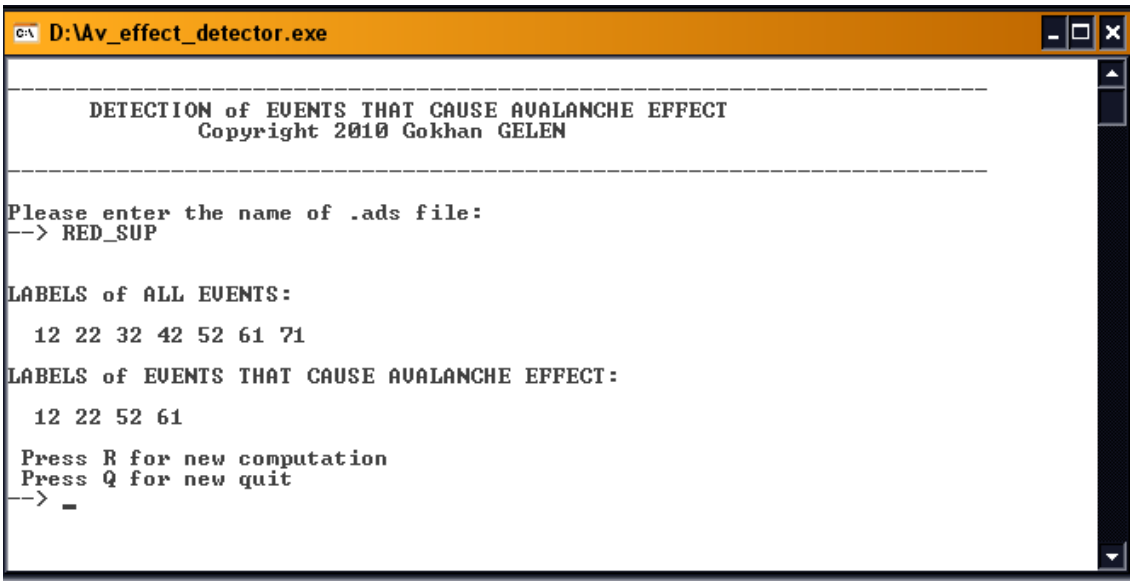
RED_CDAT	Yasaklama Oku	RED_CDAT	Yasaklama Oku
0: 61 71		1: 71	
2: 61		3: 61	
4: 61 71		5: 61 71	
6: 71			

Şekil 4.13 Kontrol verisi RED_CDAT'ın yasaklama oku karşılıkları

9. Adım: Deneysel endüstriyel üretim sistemi için elde edilen indirgenmiş gözeticinin melez kontrol sistemine ait PLC merdiven diyagramı kodu Şekil 4.16'da verilmiştir. Bu yapının PLC ile gerçekleştirilmesi için ilk olarak indirgenmiş gözetici modelinde çığ etkisine sebep olan olay bulunup bulunmadığı test edilmelidir. Bu test işlemi "Av_effect_detector.exe" programı ile yapılmıştır. Bu programa giriş olarak indirgenmiş gözeticinin .ADS dosyasının ismi girilmektedir. İndirgenmiş gözetici modelinde bulunan "12, 22, 52 ve 61" etiketli olayların çığ etkisine sebep olduğu "Av_effect_detector.exe" programı ile Şekil 4.15'te görüldüğü gibi belirlenmiştir.



Şekil 4.14 Deneysel endüstriyel üretim sistemi için hesaplanan indirgenmiş gözeticili kapalı çevrim melez kontrol sistemi



Şekil 4.15 İndirgenmiş gözetici için çığ etkisi probleminin bulunup bulunmadığını test etmek amacıyla çalıştırılan Av_effect_detector.exe programının ekran görüntüsü

İndirgenmiş gözeticili kapalı çevrim sistemini gerçekleştirmek için kullanılan PLC merdiven diyagramı kodu Şekil 4.16'da görülmektedir. PLC kodunun daha anlaşılır olması için kullanılması gereken hafıza birimleri için modelde bulunan mevki ve durum isimleri sembolik olarak gösterilmiştir. T1 (1 s. zaman gecikmeli), T2 (0.7 s. zaman gecikmeli) ve T3 (1.5 s. zaman gecikmeli) düz zaman röleleri sırası ile APN modelindeki t3, t4 ve t5 geçişlerine atanmıştır.

Şekil 4.16'da görülen kodun ilk basamağı (NW0) başlangıç atamaları için kullanılmıştır. Bu basamakta, başlangıçta içerisinde jeton bulunan mevkiler için atanan hafıza bitleri set edilmektedir. Ayrıca bu basamakta alt ve üst konveyör motorları da Q0.3 ve Q0.4 PLC çıkışları yardımıyla çalıştırılmaktadır. NW1-NW5 arasında APN modelinde bulunan kontrol edilemeyen geçişler, NW6-NW26 arasındaki basamaklarda auto-net gözeticinin geçişleri, NW27-NW28 arasında APN modelinde bulunan kontrol edilebilir geçişler ve son olarak NW29 ve NW30 da p8 ve p10 mevkilerine atanmış olan çıkış aksiyonları gerçekleştirilmiştir.

APN modelinde bulunan her bir geçiş gerçekleştirildikten sonra kullanılan [--( )] çıkış bobini ile olay oluşum sinyalleri elde edilmiştir. Çıkış bobini [--( )], ön şartının sağlanması ile bir PLC çevrimi süresince aktif olmakta, taramanın sonunda resetlenmektedir. Çığ etkisine sebep olmayan olaylar için, örneğin "32" etiketli olay için NW3 te t3 geçişi gerçekleştikten sonra [--(e32)] ataması ile e32 sinyali üretilmekte ve

auto-net gerçekleştirilirken bu sinyal kullanılmaktadır. Çığ etkisine sebep olan olaylar için ise SET komutu [--(S)] ile ilgili olay için kullanılacak sinyal set edilmekte ve hangi geçişte kullanılmakta ise o basamakta resetlenerek bir defa kullanılması sağlanmaktadır. Böylelikle çığ etkisi problemi çözülmüştür.

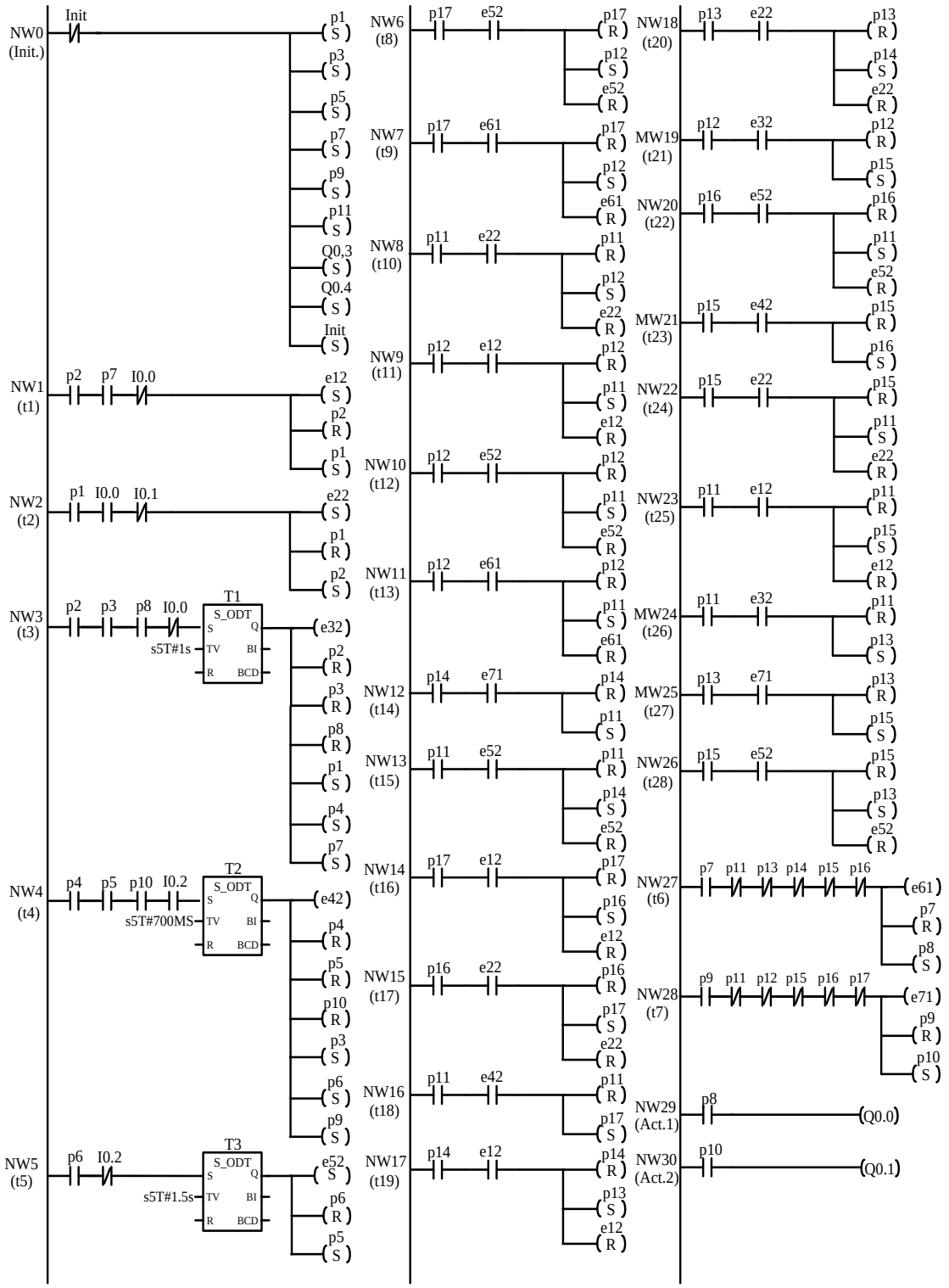
Şekil 4.16'da görülen PLC merdiven diyagramı kodu Siemens S7-300 PLC'ye yüklenerek, sistemin spesifikasyonlarda belirtilen çalışmayı başarılı bir şekilde yerine getirdiği gözlemlenmiştir.

4.4 Sonuç

Deneysel endüstriyel üretim sistemi için hesaplanan yekpare gözeticili ve indirgenmiş gözeticili melez kontrol sistemlerinin parametreleri Çizelge 4.1'de görülmektedir. Çizelgeden görülebileceği üzere indirgenmiş gözeticili kontrol sistemi daha az durum ve geçişe sahiptir. Dolayısıyla indirgenmiş sistem için yazılan PLC merdiven diyagramı kodu da hafızada daha az yer kaplamaktadır. Ancak, indirgenmiş gözeticili sistemde çığ etkisi problemi mevcuttur. Bu problemin çözümü için ekstra işlemler yapılmak zorunda kalınmıştır. Çığ etkisi problemine neden olan olayların fazlalaştığı problemler için, indirgenmiş gözeticili sistemi gerçekleştirmek için yazılacak PLC kodunun boyutu yekpare gözeticili sistemi gerçekleştirmek için yazılacak PLC kodunun boyutundan büyük olacaktır.

Çizelge 4.1 Deneysel endüstriyel üretim sistemi için hesaplanan yekpare ve indirgenmiş gözeticili sistemlerin karşılaştırma parametreleri

Gözetici	Toplam Geçiş Sayısı (Plant+Sup.)	Toplam Mevki Sayısı (Plant+Sup.)	A (Geçiş ve mevkilerin toplam sayısı)	Hafıza kullanımı (Byte)
Yekpare	30	22	52	422
İndirgenmiş	28	17	45	380



Şekil 4.16 Deneysel endüstriyel üretim sistemini indirgenmiş gözeticili kapalı çevrim melez yöntem ile kontrol etmek için elde edilen PLC merdiven diyagramı kodu

BÖLÜM V

GÖZETİMLİ KONTROL TEORİSİ GERÇEKLEŞTİRME ARACI TCT İLE PETRİ AĞI TEMELLİ DENETLEYİCİLERİN ANALİZİ

Ayrık Olay Sistemlerinin (AOS) kontrolü, son yirmi yıldır yaygın olarak çalışılmaktadır. Bu çalışmalarda, sonlu durum otomataları ve Petri ağları olmak üzere iki formal modelleme aracı kullanılmaktadır. Formal diller ve sonlu durum otomataları temelli Gözetimli Kontrol Teorisi (Supervisory Control Theory - SCT), sürekli sistemler için kullanılan kontrol teorisi yapılarını, ayrık olay sistemlerinin kontrolü için genişletmeyi önermiştir [3-9]. SCT ayrık olay sistemlerinin kontrolü çalışmaları için önerilmiş temel bir çalışmadır. Diğer yandan, kontrol tasarımı için Petri ağları (Petri Nets – PN) temelli yaklaşımlar da önerilmiştir [17, 29, 36, 91-93]. Bunun temel nedeni, vektör eklentili sistem olan Petri ağları ile elde edilen küçük boyutlu ağ yapısıyla, çok büyük boyutlu işaretlemeye sahip sistemler tanımlanabilmektedir. PN modelleri sadece gözetimli kontrolör tasarımına imkan vermezler, bunun yanında çok değişik nitel özelliklerin analizine ve nicel performans değerlendirmesine de imkan sunarlar. PN modellerinin dezavantajı, her problem için PN formunda optimum gözetici elde edilememesidir. Buna karşın, PN formunda optimum gözetici elde edilemeyen problemler için SCT ve otomata kullanılarak çözüm bulunabilir.

Ayrık olay sistemlerinin PN temelli kontrolünde, kontrol edilmemiş PN modeli, açık çevrim sistem modelini ifade eder. Herhangi bir kontrol olmaması durumunda, açık çevrim sistem beklenilen çalışmayı sağlayamaz. Sisteme ait kontrol edilmemiş PN modeli ve spesifikasyonların verildiği durumda, problemi çözmek için kontrol mevkileri (monitör) içeren PN temelli bir denetleyici sentezlenebilir. Kontrol edilmemiş PN modeli ile hesaplanan PN temelli denetleyicinin birleştirilmesi ile kapalı çevrim sistemi karakterize eden kontrol edilmiş model elde edilebilir. Spesifikasyonlar genellikle yasaklanmış durum türündendir. Yoğunlukla çalışılan bir diğer spesifikasyon türü ise, verilen PN modeli için PN temelli denetleyici hesaplayarak canlı (live) sistem davranışı (SCT terminolojisinde bloklanmasız sistem davranışı) elde edilmesidir. Maalesef, günümüzde hesaplanan PN temelli denetleyicilerin doğruluk analizini yapmak için genel bir teknik mevcut değildir. Bu analiz, verilen PN modelinin erişilebilirlik çizgesi (Reachability graph) dikkate alınarak yapılabilir [94]. Ancak, karmaşık sistemlere ait

erişilebilirlik çizgeleri dikkate alındığında bu yöntem pratik değildir. Ayrıca, kullanılabilecek PN analiz programları ile kontrol edilemeyen geçişlerin bulunduğu PN modellerinin analizi mümkün değildir. Bu nedenle, hesaplanan PN temelli denetleyicilerin doğruluk analizini yapmak için genel bir metodun elde edilmesi çok önemlidir. Bu kısımda, hesaplanan PN temelli denetleyicilerin doğruluk analizlerini yerine getirmek için TCT [68] yazılımı kullanılan genel bir metot önerilmiştir. TCT yazılımı, gözetimli kontrol teorisini öneren Prof. Dr. W. M. Wonham danışmanlığında SCT için gerekli hesaplamaları yapmak üzere geliştirilmektedir.

Bu kısımda önerilen doğruluk analizi kısaca şu şekilde yapılır: Başlangıçta, kontrol edilmemiş PN modelinin, spesifikasyonların ve PN temelli denetleyicinin (monitör veya kontrol mevkileri) verildiği kabul edilir. Doğruluk analizi aşağıdaki üç ana adım takip edilerek yerine getirilir.

1. Verilen problem PN domeninden otomata domenine dönüştürülerek SCT ve TCT yazılımı kullanılarak çözülür. Bu dönüşümler kontrol edilmemiş PN modeli ve spesifikasyonların her ikisini de içerir. Bu adımda elde edilen RWSUPER olarak adlandırılmış mükemmel (proper) gözetici, dikkate alınan problem için maksimum müsaade edici (maximally permissive) ve bloklanmasızdır.
2. PN temelli denetleyicinin otomata karşılığı ile kontrol edilmemiş sistem davranışına ait PN modelinin otomata karşılığının birleştirilmesi ile sistemin kontrol edilmiş modeli elde edilir. Bu birleştirme, senkron çarpım (synchronous product) işlemi ile gerçekleştirilir. Bu adımdan elde edilen gözetici PNSUPER olarak adlandırılır.
3. RWSUPER ve PNSUPER karşılaştırılır. Eğer iki model eşit ise "PN temelli denetleyici maksimum müsaade edici (maximally permissive) ve bloklanmasızdır" denir.

PN domeninden otomata domenine dönüşüm [52 ve 67]'de kullanılmıştır. Bu dönüşümler için daha genel PN sınıflarını içeren iyileştirmeler bu çalışmada önerilmiştir. PN'lerin erişilebilirlik çizgeleri kullanılarak otomatalara dönüştürülebileceği bilinmektedir [95]. PN'lerden otomatalara dönüşümle ilgili bazı güncel çalışmalar [45, 96-98]'de görülebilir. Otomatalardan PN'lere dönüşüm de

mümkündür [43]. Bu dönüşümler hali hazırda, ayrık olay sistemlerinin kontrolüne uygulanmıştır [42, 44-46].

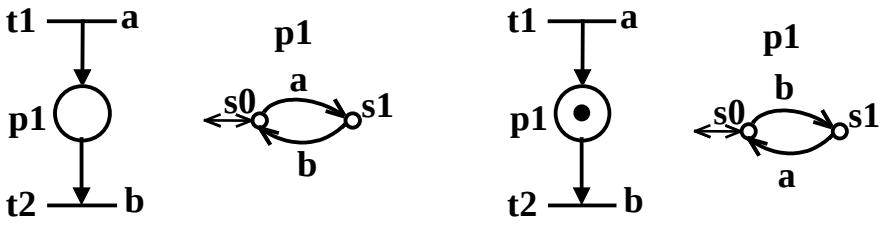
5.1 Sıradan Petri Ağlardan Sonlu Durum Otomatalarına Dönüşüm

Güvenilir ve sıradan PN'lerden (Safe Ordinary Petri Net - SOPN) sonlu durum otomatalarına dönüşüm için bu çalışmada önerilen temel strateji, bir PN mevkisinin iki durumlu bir otomata karşılığına dönüştürülmesine dayanmaktadır. Otomatanın durumları arasındaki geçişler ve özdöngüler (selfloops) SOPN modelinin davranışını simgelemektedir. PN yapısında bulunan her bir geçiş, otomata yapısında bir olay (event) ile karakterize edilecektir. Bu çalışmada gösterilen örneklerde her bir olaya karşılık gelen PN geçişine harfler etiket olarak atanmıştır. Ayrıca, bu çalışmada kullanılan PN modellerindeki geçişlerin tümü kontrol edilemeyen olarak gösterilmiştir. Ancak önerilen dönüşümler kontrol edilebilir geçişleri içeren PN modelleri için de kullanılabilir. Bu çalışmada kullanılan PN modellerindeki geçişlerin eşzamanlı olarak tetiklenmediği varsayılmıştır. Yani bir anda sadece bir geçişin tetiklenebildiği kabul edilmektedir.

5.1.1 Bir giriş ve bir çıkış geçişine sahip mevkinin otomata karşılığı

İki geçiş ve bir mevkiden oluşan basit PN modelleri ve bu modellerin otomata karşılıkları Şekil 5.1'de görülmektedir. Başlangıçta içerisinde jeton bulunan ve jeton bulunamayan mevkiler için otomata karşılıkları sırası ile Şekil 5.1.(a) ve Şekil 5.1.(b)'de görülmektedir. Şekil 5.1.(a)'da başlangıçta t1 geçişi tetiklenmeye açıktır. t1 geçişinin tetiklenmesi ile p1 mevkisine bir jeton depolanmaktadır. p1 mevkisinde jeton olması ile t2 geçişi tetiklenmeye açık hale gelir ve t2 geçişinin tetiklenmesi ile p1 mevkisinde bulunan jeton alınır. Bu modele ait otomata karşılığı Şekil 5.1.(a)'da görüldüğü gibi iki adet durum (state) içermektedir. Bunlardan s0 durumu p1 mevkisinde jeton olmadığını simgeleyen başlangıç (initial state) ve işaretlenmiş (marker state) durumdur. s1 ise p1 mevkisinin içerisinde jeton bulunan durumu simgelemektedir. Başlangıç durumunda bulunan otomata, 'a' olayının gerçekleşmesi ile s0 durumundan s1 durumuna geçmektedir. s1 durumundayken 'b' olayının gerçekleşmesi ile otomata s0 durumuna geri dönmektedir.

Başlangıçta jeton içeren mevkiye ait PN modeli ve otomata karşılığı Şekil 5.1.(b)'de görülmektedir. Bu PN modelinde başlangıçta sadece t2 geçişi tetiklemeye açıktır. t2 geçişinin tetiklenmesi ile mevkide bulunan jeton alınmaktadır. p1 mevkisi içerisinde jeton olmaması ile t1 geçişi tetiklenmeye açık hale gelir ve t1 geçişinin tetiklenmesi ile p1 mevkisine jeton depolanır. Şekil 5.1.(b)'de görülen otomata karşılığında, iki durum mevcuttur. Bu durumlardan s0 başlangıç durumunu yani p1 mevkisinde jeton olduğunu, s1 durumu ise mevkide jeton olmadığını simgelemektedir. Otomata, başlangıç durumunda iken 'b' olayının gerçekleşmesi ile s1 durumuna geçer. Benzer olarak, otomata s1 durumunda iken 'a' olayının gerçekleşmesi ile de s0 başlangıç durumuna geri dönmektedir.

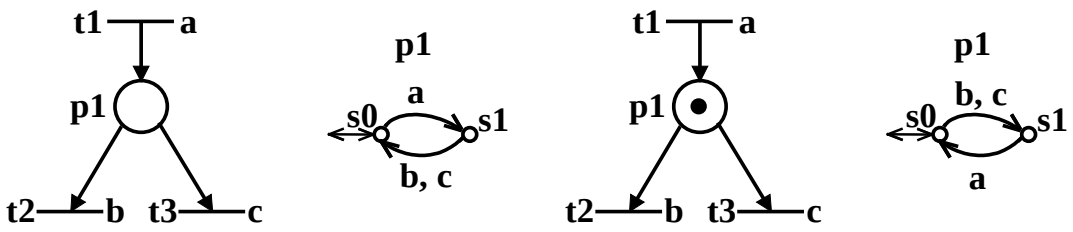


a) b)

Şekil 5.1 Bir giriş ve bir çıkış geçişine sahip bir PN mevkisi ve otomata karşılığı
a) Başlangıç jetonu bulunmayan $M_0(p1)=0$, b) Başlangıç jetonu bulunan $M_0(p1)=1$

5.1.2 Seçim (choice) modeli için dönüşüm

Seçim modeli, iki veya daha fazla aktiviteden (işlemden) birinin çalışabileceği durumlarda kullanılmaktadır. Böyle durumları modelleyebilmek için birden fazla çıkış geçişine sahip mevki kullanılmaktadır. Şekil 5.2'de seçim modelleri ve bu modellerin otomata karşılıkları görülmektedir. Şekil 5.2.(a)'da başlangıç jeton dağılımı $M_0(p1)=0$ olduğu durum için seçim modeli, Şekil 5.2.(b)'de ise başlangıç jeton dağılımı $M_0(p1)=1$ olduğu durum için seçim modeli görülmektedir. Bu modellerde t1, t2 ve t3 geçişlerine a, b ve c harfleri kullanılarak olay etiketleri atanmıştır. Seçim modeli için önerilen dönüşümlerdeki durumlar arası geçişler ilgili olayların oluşumuna göre bir önceki dönüşüme benzer şekilde gerçekleşmektedir.

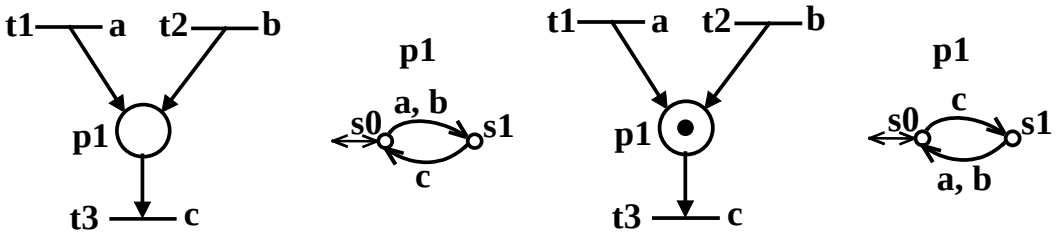


a) b)

Şekil 5.2 Seçim modelleri ve otomata karşılıkları
a) Başlangıç jeton dağılımı $M_0(p1)=0$, b) Başlangıç jeton dağılımı $M_0(p1)=1$

5.1.3 Birleştirme (merge) modeli için dönüşüm

Bu model kullanılarak bir veya birden fazla aktivite (işlem) birleştirilebilir. Bu birleştirme işlemini modelleyebilmek için birden fazla giriş geçişine sahip bir mevki kullanılmaktadır. Birleştirme modelleri ve bu modellerin otomata karşılıkları Şekil 5.3'te görülmektedir. Şekil 5.3.(a)'da başlangıç jeton dağılımı $M_0(p1)=0$ olduğu durum için birleştirme modeli, Şekil 5.3.(b)'de ise başlangıç jeton dağılımı $M_0(p1)=1$ olduğu durum için birleştirme modeli görülmektedir. Bu PN modellerindeki t1, t2 ve t3 geçişlerine sırası ile 'a', 'b' ve 'c' harfleri ile etiketlenmiştir. Birleştirme modeli için önerilen dönüşümlerdeki durumlar arası geçişler, ilgili olayların oluşumuna göre bir önceki dönüşüme benzer şekilde gerçekleşmektedir.



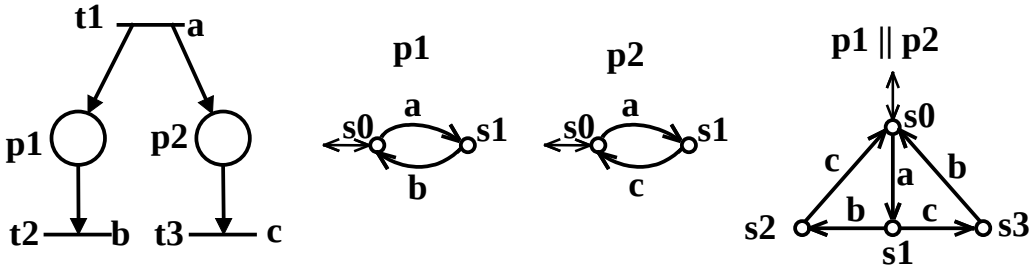
a) b)

Şekil 5.3 Birleştirme (merge) modelleri ve otomata karşılıkları
a) Başlangıç jeton dağılımı $M_0(p1)=0$, b) Başlangıç jeton dağılımı $M_0(p1)=1$

5.1.4 Çatal (fork) modeli için dönüşüm

İki veya daha fazla işlemin eş zamanlı olarak başlatıldığı durumları modellemek için çatal modeli kullanılmaktadır. Çatal (eş zamanlılık) modeli, iki veya daha fazla çıkış mevkisine sahip bir geçiş ile oluşturulabilmektedir. Şekil 5.4'te bir çatal modeli ve

otomata karşılığı görülmektedir. Başlangıçta p1 ve p2 mevkilerinde jeton bulunmamaktadır. Bu SOPN modelindeki t1, t2 ve t3 geçişleri ilgili olayları karakterize eden ‘a’, ‘b’ ve ‘c’ harfleri ile etiketlenmiştir. p1 ve p2 mevkilerinde jeton mevcut değilken t1 geçişinin tetiklenmesi ile bu iki mevkiye de birer jeton depolanmaktadır. Bu, p1 ve p2 ile simgelenen iki ayrı olayın aynı anda başlatılması anlamına gelmektedir. p1 mevkisinde jeton varken t2 geçişi tetiklenmeye açıktır. t2 geçişinin tetiklenmesi ile p1 mevkisinden jeton alınır. Benzer olarak, p2 mevkisinde jeton varken t3 geçişi tetiklenmeye açıktır. t3 geçişinin tetiklenmesi ile p2 mevkisindeki jeton alınmaktadır. Bu modeldeki her bir mevki Şekil 5.4’te görüldüğü gibi iki durumlu bir otomataya dönüştürülebilir. Çatal modeline ait toplam otomata p1 ve p2 mevkisine ait otomataların senkron çarpımı (synchronous product p1||p2) ile elde edilir. p1||p2 otomata modelinde Q = {s0, s1, s2, s3} olmak üzere dört durum mevcuttur. Bu durumlar, sırası ile SOPN modelindeki $M(p1,p2)^T$ = {00, 11, 01, 10} jeton işaretlemelerini simgelemektedir. Otomata modelindeki durumlar arası geçişler yukarıdaki modellerde anlatıldığı gibi gerçekleşmektedir.

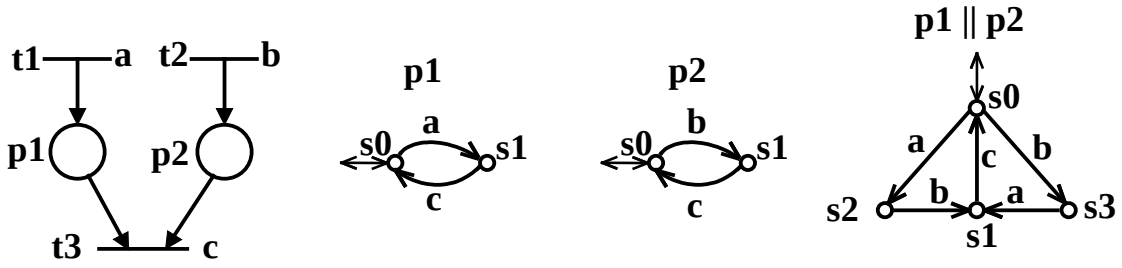


Şekil 5.4 Çatal modeli ve otomata karşılığı

5.1.5 Bağlama (join) modeli için dönüşüm

İki veya daha fazla farklı işlemin senkronize edilerek tek bir işlem haline getirilmesi bağlama (join) olarak bilinmektedir. Bağlama (Senkronizasyon) modeli, iki veya daha fazla giriş mevkisine sahip bir geçiş ile oluşturulabilmektedir. Şekil 5.5’te örnek bir bağlama modeli ve otomata karşılığı görülmektedir. Başlangıçta p1 ve p2 mevkilerinde jeton bulunmamaktadır. Bu SOPN modelindeki t1, t2 ve t3 geçişleri ilgili olayları karakterize eden ‘a’, ‘b’ ve ‘c’ harfleri ile etiketlenmiştir. Bu PN modelinde başlangıçta t1 ve t2 geçişlerinin ikisi de tetiklenmeye açıktır. t1 geçişinin tetiklenmesi ile p1 mevkisine, t2 geçişinin tetiklenmesi ile de p2 mevkisine bir jeton depolanmaktadır. p1 ve p2 mevkilerinde jeton olduğu durumda t3 geçişi tetiklenmeye açık hale gelir ve t3

geçişinin tetiklenmesi ile birlikte bu mevkilerden jetonlar alınır. Bu işlem iki aktivitenin senkronize edildiği anlamına gelmektedir. Bu modelin otomata karşılığı, modelde bulunan her bir mevkinin iki durumlu otomata karşılığının senkron çarpımı (synchronous product p1||p2) ile Şekil 5.5'te görüldüğü gibi elde edilir. p1||p2 otomata modelinde Q = {s0, s1, s2, s3} olmak üzere dört durum mevcuttur. Bu durumlar sırası ile SOPN modelindeki $M(p1,p2)^T$ = {00, 11, 10, 01} jeton işaretlemelerini simgelemektedir. Otomata modelindeki durumlar arası geçişler yukarıdaki modellerde anlatıldığı gibi gerçekleşmektedir.

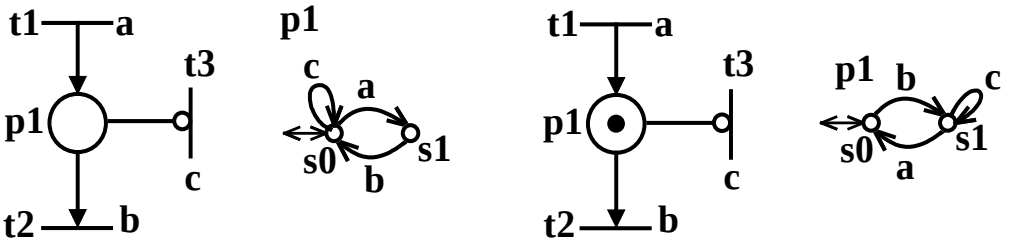


Şekil 5.5 Bağlama modeli ve otomata karşılığı

5.1.6 Yasaklama oku için dönüşüm

SOPN'de bir mevkide jeton olmadığı yasaklama oku kullanılarak test edilebilmektedir. Şekil 5.6'da bir giriş, bir çıkış ve bir In(p1,t3) yasaklama okuna sahip mevkiden oluşan SOPN modelleri ve otomata karşılıkları görülmektedir. Şekil 5.6.(a)'da başlangıç jeton dağılımı $M_0(p1)=0$ olduğu durum için, Şekil 5.6.(b)'de ise başlangıç jeton dağılımı $M_0(p1)=1$ olduğu durum için yasaklama oku modeli görülmektedir. Bu SOPN modelindeki t1, t2 ve t3 geçişleri ilgili olayları karakterize eden 'a', 'b' ve 'c' harfleri ile etiketlenmiştir. Şekil 5.6.(a)'da görülen SOPN için başlangıçta t1 ve t3 geçişleri tetiklenmeye açıktır. t1 geçişinin tetiklenmesi ile p1 mevkisine jeton depolanmaktadır. p1 mevkisinde jeton bulunduğu durumda t2 geçişi tetiklenmeye açıktır. Ancak, bu durumda t3 geçişi tetiklenmeye açık değildir. t2 geçişinin tetiklenmesi ile p1 mevkisindeki jeton alınmaktadır. Kısacası, p1 mevkisinde jeton bulunmadığı durumda t3 geçişi tetiklenmeye açıktır, p1 mevkisinde jeton bulunduğu durumda ise tetiklenmeye kapalıdır. Bu modelin otomata karşılığı Şekil 5.6.(a)'da görüldüğü gibi s0 ve s1 olmak üzere iki durumdan oluşmaktadır. Bunlardan s0 durumu p1 mevkisinde jeton olmadığını simgeleyen başlangıç (initial state) ve işaretlenmiş (marker state) durumdur. s1 ise p1 mevkisinin içerisinde jeton bulunan durumu simgelemektedir. Başlangıç durumu s0'a

[s0,c,s0] ile ifade edilen bir öz-döngü (self-loop) eklenerek bu durumda ‘c’ olayının gerçekleşmesi mümkün hale getirilmiştir. Başlangıç durumunda bulunan otomata, ‘a’ olayının gerçekleşmesi ile s0 durumundan s1 durumuna geçmektedir. s1 durumundayken ‘c’ olayının gerçekleşmesine izin verilmemektedir. Bu durumda iken ‘b’ olayının gerçekleşmesi ile otomata s0 durumuna geri dönmektedir. Şekil 5.6.(b)’deki SOPN modelinde başlangıçta sadece t2 geçişi tetiklenmeye açıktır. t2 geçişinin tetiklenmesi ile p1 mevkisindeki jeton alınır ve bu durumda t1 ve t3 geçişleri tetiklenmeye açık hale gelir. t1 geçişinin tetiklenmesi ile p1 mevkisine jeton depolanmaktadır. Bu modelin otomata karşılığı Şekil 5.6.(b)’de görüldüğü gibi s0 ve s1 olmak üzere iki durumdan oluşmaktadır. Bunlardan s0 durumu p1 mevkisinde jeton bulunduğu durumu simgeleyen başlangıç (initial state) ve işaretlenmiş (marker state) durumdur. s1 ise p1 mevkisinin içerisinde jeton bulunmayan durumu simgelemektedir. Başlangıçta s0 durumundan ‘b’ olayının gerçekleşmesi ile otomata s1 durumuna geçmektedir. s1 durumuna [s1,c,s1] ile ifade edilen bir öz-döngü (self-loop) eklenerek bu durumda ‘c’ olayının gerçekleşmesi mümkün hale getirilmiştir. s1 durumunda bulunan otomata, ‘a’ olayının gerçekleşmesi ile s0 durumuna dönmektedir. s0 durumundayken ‘c’ olayının gerçekleşmesine izin verilmemektedir.



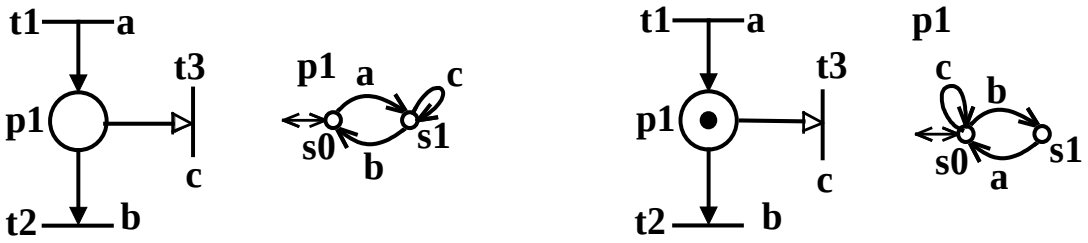
a) b)

Şekil 5.6 Yasaklama oku içeren SOPN modelleri ve otomata karşılıkları
a) Başlangıç jeton dağılımı $M_0(p1)=0$, b) Başlangıç jeton dağılımı $M_0(p1)=1$

5.1.7 Yetkileme oku için dönüşüm

SOPN’de bir mevkide jeton olduğu yetkileme oku kullanılarak test edilebilmektedir. Şekil 5.7’de bir giriş, bir çıkış ve bir En(p1,t3) yetkileme okuna sahip mevkiden oluşan SOPN modelleri ve otomata karşılıkları görülmektedir. Şekil 5.7.(a)’da başlangıç jeton dağılımı $M_0(p1)=0$ olduğu durum için, Şekil 5.7.(b)’de ise başlangıç jeton dağılımı $M_0(p1)=1$ olduğu durum için yetkileme oku modeli görülmektedir. Bu SOPN modelindeki t1, t2 ve t3 geçişleri ilgili olayları karakterize eden ‘a’, ‘b’ ve ‘c’ harfleri ile etiketlenmiştir. Şekil 5.7.(a)’da görülen SOPN modelinde başlangıçta sadece t1

geçişi tetiklenmeye açıktır. t3 geçişi p1 mevkisinde jeton olana kadar tetiklenmeye açık değildir. t1 geçişinin tetiklenmesi ile p1 mevkisine bir jeton depolanmaktadır. p1 mevkisinde jeton bulunduğu durumda t2 ve t3 geçişleri tetiklenmeye açık hale gelir. t2 geçişinin tetiklendiği durumda p1 mevkisindeki jeton alınır. t3 geçişinin tetiklendiği durumda ise p1 mevkisinden jeton alınmaz. Bu modelin otomata karşılığı Şekil 5.7.(a)'da görüldüğü gibi s0 ve s1 olmak üzere iki durumdan oluşmaktadır. Bunlardan s0 durumu p1 mevkisinde jeton olmadığını simgeleyen başlangıç (initial state) ve işaretlenmiş (marker state) durumdur. s1 ise p1 mevkisinin içerisinde jeton bulunan durumu simgelemektedir. Otomata başlangıç durumunda iken 'c' olayının oluşmasına izin verilmez. s0 durumunda iken 'a' olayının gerçekleşmesi ile otomata s1 durumuna geçmektedir. s1 durumuna [s1,c,s1] ile ifade edilen bir öz-döngü (self-loop) eklenerek bu durumda 'c' olayının gerçekleşmesi mümkün hale getirilmiştir. Bu durumda iken 'b' olayının gerçekleşmesi ile otomata s0 durumuna geri dönmektedir. s0 durumundayken 'c' olayının gerçekleşmesine izin verilmemektedir. Şekil 5.7.(b)'deki SOPN modelinde başlangıçta t2 ve t3 geçişleri tetiklenmeye açıktır. t3 geçişinin tetiklenmesi ile p1 mevkisindeki jetonda bir değişim olmaz. t2 geçişinin tetiklenmesi ile p1 mevkisindeki jeton alınır ve bu durumda t1 geçişi tetiklenmeye açık hale gelirken t3 geçişi tetiklenmeye kapalı hale gelmektedir. Bu modelin otomata karşılığı Şekil 5.7.(b)'de görüldüğü gibi s0 ve s1 olmak üzere iki durumdan oluşmaktadır. Bunlardan s0 durumu p1 mevkisinde jeton bulunduğu durumu simgeleyen başlangıç (initial state) ve işaretlenmiş (marker state) durumdur. s1 ise p1 mevkisinin içerisinde jeton bulunmadığı durumu simgelemektedir. s0 durumuna [s0,c,s0] ile ifade edilen bir öz-döngü (self-loop) eklenerek bu durumda 'c' olayının gerçekleşmesi mümkün hale getirilmiştir. s0 başlangıç durumunda bulunan otomata, 'b' olayının gerçekleşmesi ile s1 durumuna geçmektedir. s1 durumundayken 'c' olayının gerçekleşmesine izin verilmemektedir. s1 durumundayken 'a' olayının gerçekleşmesi ile s0 durumuna geri dönülmektedir.

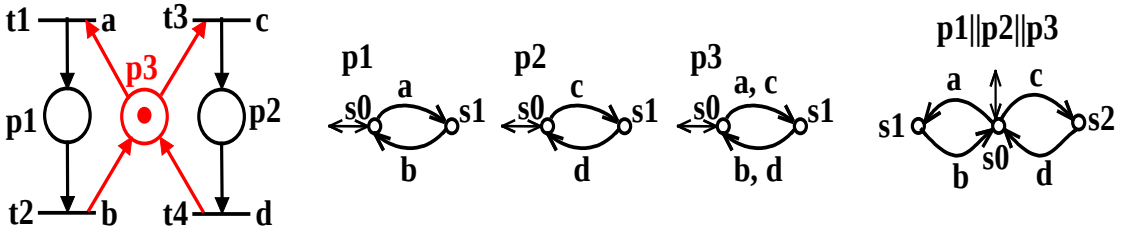


a) b)

Şekil 5.7 Yetkileme oku içeren SOPN modelleri ve otomata karşılıkları
a) Başlangıç jeton dağılımı $M_0(p1)=0$, b) Başlangıç jeton dağılımı $M_0(p1)=1$

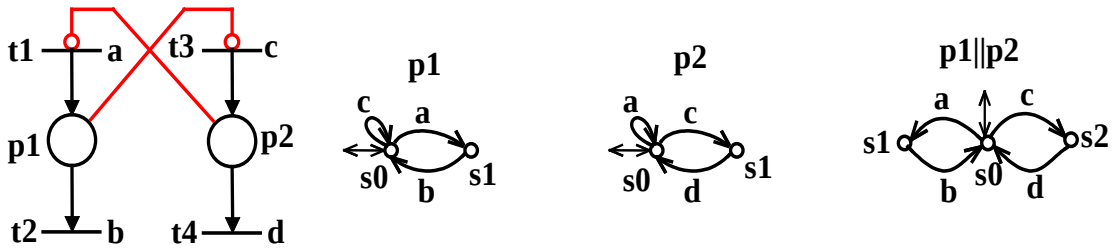
5.1.8 Karşılıklı dışlama (mutual exclusion) modeli için dönüşüm

Üretim sistemlerinde, robot, makine vb. ortak kaynakları iki veya daha fazla işlem aynı anda kullanmaya çalışabilir. Bu durumu önlemek için karşılıklı dışlama (mutual exclusion) kullanılmaktadır. Şekil 5.8'de bir karşılıklı dışlama örneği ve otomata karşılığı görülmektedir. Başlangıçta p1 ve p2 mevkilerinde jeton yoktur, p3 mevkisinde ise jeton bulunmaktadır. Bu SOPN modelindeki t1, t2, t3 ve t4 geçişleri ilgili olayları karakterize eden 'a', 'b', 'c' ve 'd' harfleri ile etiketlenmiştir. Bu SOPN modelinde, p1 ve p2 ile simgelenen iki işlem p3 mevkisindeki jeton ile simgelenen ortak kaynağı kullanmaktadır. Burada, p3 mevkisi M(p1)+M(p2)+ M(p3) = 1 koşulunu sağlamaktadır. Yani, her hangi bir anda p1, p2 ve p3 mevkilerindeki jeton toplamı birdir. Başlangıçta t1 ve t2 geçişleri tetiklenmeye açıktır. t1 geçişinin tetiklenmesi ile p3 mevkisinden jeton alınarak p1 mevkisine bir jeton depolanmaktadır. p3 mevkisinde jeton varken t3 geçişi tetiklenirse p3 mevkisinden jeton alınarak p2 mevkisine bir jeton depolanmaktadır. Bunun anlamı, başlangıçta t1 veya t3 ile ifade edilen aktivitelerden sadece birinin gerçekleşebileceğidir. p1 mevkisinde jeton varken t2 geçişi tetiklenmeye açılır ve t2 geçişinin tetiklenmesi ile p1 mevkisinden jeton alınarak p3 mevkisine depolanır. Benzer olarak, p2 mevkisinde jeton varken t4 geçişi tetiklenmeye açılır ve t4 geçişinin tetiklenmesi ile p2 mevkisinden jeton alınarak p3 mevkisine depolanır. Bu modelin otomata karşılığını bulmak için öncelikle her bir mevkinin iki durumlu otomata karşılığı Şekil 5.8'de görüldüğü gibi belirlenir. Karşılıklı dışlama modeline ait toplam otomata p1, p2 ve p3 mevkilerine ait otomataların senkron çarpımı (synchronous product p1||p2||p3) ile elde edilir. p1||p2||p3 otomata modelinde Q = {s0, s1, s2} olmak üzere üç durum mevcuttur. Bu durumlar sırası ile SOPN modelindeki $M(p1,p2,p3)^T$ = {001, 100, 010} jeton işaretlemelerini simgelemektedir. Otomata modelindeki durumlar arası geçişler yukarıdaki modellerde anlatıldığı gibi gerçekleşmektedir.



Şekil 5.8 Karşılıklı dışlama (mutual exclusion) modeli ve otomata karşılığı

Şekil 5.8'de görülen p3 mevkisi literatürde, kontrol özelliklerini (spesifikasyon) yerine getirecek kontrol mevkisi veya monitör olarak adlandırılmaktadır. Bu örnekte kontrol spesifikasyonu M(p1) + M(p2)≤ 1'dir. Bu, p1 ve p2 mevkilerindeki jetonların toplamı bire eşit veya küçük olması gerektiği anlamına gelmektedir. Aynı özellik, Şekil 5,9'da görülen alternatif bir karşılıklı dışlama (mutual exclusion) modeli kullanılarak sağlanabilir. Başlangıçta p1 ve p2 mevkilerinde jeton olmadığı için, t1 ve t3 geçişlerinin ikisi de tetiklenmeye açıktır. t1 geçişinin tetiklenmesi ile p1 mevkisine bir jeton depolanmaktadır. p1 mevkisinde jeton olması ile In(p1,t3) yasaklama okundan dolayı t3 geçişi tetiklenmeye yasaklanır. Benzer olarak, başlangıçta t3 geçişi tetiklenirse p2 mevkisine jeton depolanır. p2 mevkisinde jeton olması ile In(p2,t1) yasaklama okundan dolayı t1 geçişi tetiklenmeye yasaklanır. p1 mevkisi içerisinde jeton olması ile t2 geçişi, p2 mevkisinde jeton olması ile de t4 geçişi tetiklenmeye açık hale gelir. t2 geçişinin tetiklenmesi ile p1 mevkisindeki jeton, t4 geçişinin tetiklenmesi ile de p2 mevkisindeki jeton alınmaktadır. Bu modelin otomata karşılığını bulmak için öncelikle her bir mevkinin iki durumlu otomata karşılığı Şekil 5.9'da görüldüğü gibi belirlenir. Karşılıklı dışlama modeline ait toplam otomata p1 ve p2 mevkilerine ait otomataların senkron çarpımı (synchronous product p1||p2) ile elde edilir. p1||p2 otomata modelinde Q = {s0, s1, s2}, olmak üzere üç durum mevcuttur. Bu durumlar sırası ile SOPN modelindeki $M(p1,p2)^T$ = {00, 10, 01} jeton işaretlemelerini simgelemektedir. Şekil 5.8'de görülen p1||p2||p3 otomatası ile Şekil 5.9'da görülen p1||p2 otomatasının aynı olduğu rahatlıkla görülebilmektedir. Bir başka deyişle, bu iki otomata aynı dili üretmektedir. Şekil 5.8 ve 5.9'da görülen SOPN modelleri aynı kontrol spesifikasyonunu sağlamaktadır. Bu nedenle bu modellere kontrol eşdeğerdir (control equivalent) denilebilir.



Şekil 5.9 M(p1) + M(p2) ≤ 1 spesifikasyonunu sağlayan alternatif karşılıklı dışlama (mutual exclusion) modeli ve otomata karşılığı

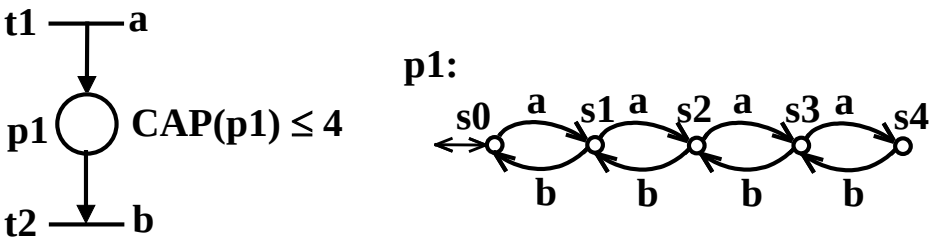
5.2 Ağırlıklı Oka Sahip Petri Ağlardan Sonlu Durum Otomatalarına Dönüşüm

Bu kısımda, ağırlıklı oka sahip Petri ağlarının sonlu durum otomatalarına çevrilmesinde kullanılacak bazı dönüşümler önerilmektedir. Ağırlıklı giriş, çıkış, yasaklama ve yetkileme okuna sahip PN modellerinden FSA'ya dönüşüm için ilk şart dikkate alınan PN modelinin sınırlı olmasıdır. Aksi durumda dönüşüm yapılamaz. Dönüşümde ilk olarak verilen PN modelindeki *p* mevkisinin jeton kapasitesi CAP(p) bilinmelidir. Daha sonra PN modelinde bulunan her bir *p* mevkisi ayrı olarak ok bağlantıları ile dikkate alınır. Jeton kapasitesi CAP(*p*) ≤ b sınırlı olan bir *p* mevkisi b+1 duruma sahip bir tampona dönüştürülebilir. Otomatanın durumları arasındaki geçişler ve öz-döngüler (selfloops) PN modelindeki geçişleri simgelemektedir. PN yapısında bulunan her bir geçiş etiketlenerek, otomata yapısında bir olay (event) ile karakterize edilmektedir. Bu çalışmada gösterilen örneklerde her bir olaya karşılık gelen PN geçişine harfler etiket olarak atanmıştır. Ayrıca, bu kısımda kullanılan örnek PN modellerindeki geçişlerin tümü kontrol edilemeyen olarak gösterilmiştir. Fakat önerilen dönüşümler kontrol edilebilir geçişleri içeren PN modelleri için de kullanılabilir. Bu çalışmada kullanılan PN modellerindeki geçişlerin eşzamanlı olarak tetiklenmediği varsayılmıştır. Yani bir anda sadece bir geçişin tetiklenebildiği kabul edilmektedir. Önerilen dönüşümleri göstermek için çeşitli ok bağlantılarına sahip bir adet mevki dikkate alınmıştır.

5.2.1 Jeton kapasitesi CAP(*p*) ≤ b olan bir mevkinin otomata karşılığı

Bu kısımda iki adet dönüşüm örneği dikkate alınmıştır. Şekil 5.10'dan görüleceği gibi ilk örnek, jeton kapasitesi CAP(*p1*) ≤ 4 olan, Pre(t1,p1) giriş okuna ve Post(p1,t2) çıkış okuna sahip p1 mevkisinin otomata karşılığına dönüştürülmesini içerir. Başlangıçta p1 mevkisinde jeton bulunmamaktadır. p1 mevkisinde en çok 4 jeton oluncaya kadar, t1 geçişinin her tetiklenmesinde bir jeton p1 mevkisine eklenir. p1 mevkisinde jeton bulunması durumunda ise t2'nin her tetiklenmesiyle p1'deki jeton sayısı bir azalır. Bu PN modelindeki t1 ve t2 geçişleri, ilgili olayları karakterize eden 'a' ve 'b' harfleri ile etiketlenmiştir. p1 mevkisinin jeton kapasitesi 4 olduğu için bu PN modeline ait otomata karşılığında 5 adet durum bulunmaktadır. Bunlardan s0 durumu p1 mevkisinde jeton olmadığını simgeleyen başlangıç (initial state) ve işaretlenmiş (marker state) durumdur. Sırasıyla s1, s2, s3 ve s4 durumları da ise p1 mevkisinde bulunan 1, 2, 3 ve 4 adet

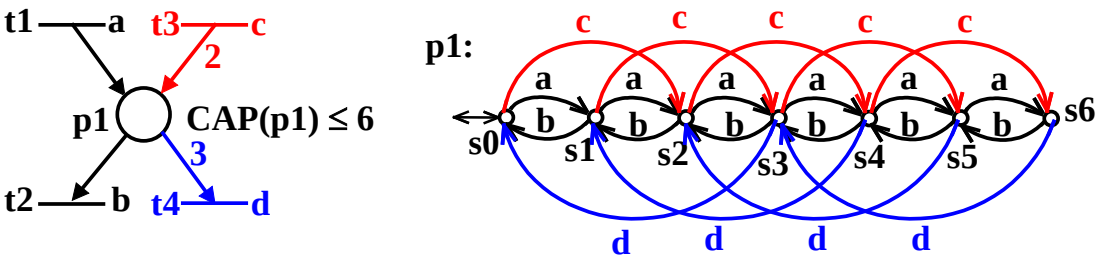
jetonu simgelemektedir. Otomata modelinde, t1 geçişinin tetiklenmesini temsil eden ‘a’ olayının gerçekleşmesi ile jeton sayısının bir değer artması anlamına gelirken t2 geçişinin tetiklenmesini temsil eden ‘b’ olayının gerçekleşmesi ile jeton sayısının bir değer azalması anlamındadır.



Şekil 5.10 Jeton kapasitesi CAP(*p1*) ≤ 4 olan mevki içeren PN modeli ve otomata karşılığı

İkinci örnekte Şekil 5.11’de görülen iki giriş ve iki çıkış geçişine sahip ve jeton kapasitesi 6 olan bir PN mevkisi ve otomata karşılığı incelenmektedir. Bu örnek PN’deki p1 mevkisinin jeton kapasitesi 6 [CAP(p1) ≤ 6] ve bu mevkinin giriş oklarının ağırlıkları, Pre(t1,p1)=1, Pre(t3,p1)=2 ve çıkış oklarının ağırlıkları, Post(p1,t2)=1, Post(p1,t4)=3 olarak verilmiştir. Buna göre, p1’deki jeton sayısı 6’dan küçükse t1 geçişinin her bir tetiklenmesi p1’deki jeton sayısını bir arttırır. Benzer şekilde, p1’deki jeton sayısı 6’dan küçükse ve tetiklenme sonucunda 6’yı geçmeyecekse t3 geçişinin her bir tetiklenmesi p1’deki jeton sayısını iki değer arttırır. p1’de jeton bulunması durumunda ise t2’nin her tetiklenmesiyle p1’deki jeton sayısı bir azalır. Benzer şekilde, p1’deki jeton sayısının 3’e eşit ya da 3’ten fazla olması durumunda ise t4’ün her tetiklenmesiyle p1’deki jeton sayısı üç değer azalır. Bu PN modelindeki t1, t2, t3 ve t4 geçişleri, ilgili olayları karakterize eden ‘a’, ‘b’ , ‘c’ ve ‘d’ harfleri ile etiketlenmiştir. p1 mevkisinin jeton kapasitesi 6 olduğu için bu PN modeline ait otomata karşılığında 7 adet durum bulunmaktadır. Bunlardan s0 durumu p1 mevkisinde jeton olmadığını simgeleyen başlangıç (initial state) ve işaretlenmiş (marker state) durumdur. Sırasıyla s1, s2, s3, s4, s5 ve s6 durumları da ise p1 mevkisinde bulunan 1, 2, 3, 4, 5 ve 6 adet jetonu simgelemektedir. Otomata modelinde, t1 geçişinin tetiklenmesini gösteren ‘a’ olayının gerçekleşmesi ile jeton sayısının bir değer artması temsil edilirken t2 geçişinin tetiklenmesini gösteren ‘b’ olayının gerçekleşmesi ile jeton sayısının bir değer azalması temsil edilmektedir. Benzer şekilde, t3 geçişinin tetiklenmesini gösteren ‘c’ olayının gerçekleşmesi ile jeton sayısının iki değer artması temsil edilirken t4 geçişinin

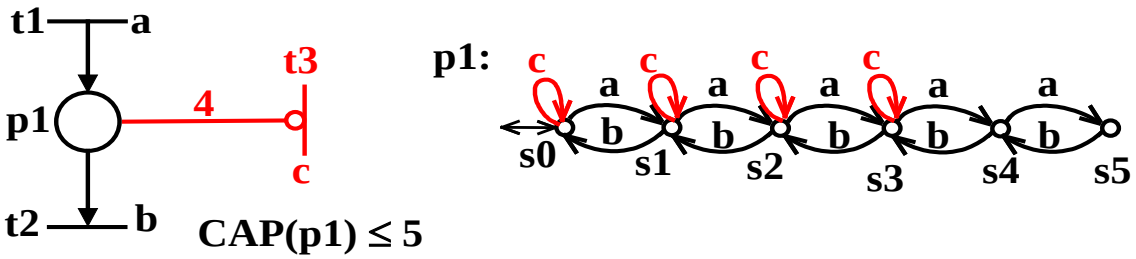
tetiklenmesini gösteren ‘d’ olayının gerçekleşmesi ile jeton sayısının üç değer azalması temsil edilmektedir.



Şekil 5.11 İki giriş ve iki çıkış geçişine sahip ve jeton kapasitesi CAP(*p1*) ≤ 6 olan bir PN modeli ve otomata karşılığı

5.2.2 Ağırlıklı yasaklama oku için dönüşüm

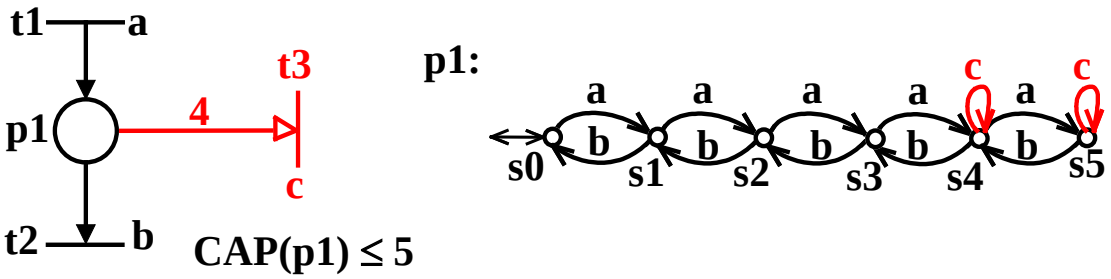
Bu kısımda, Petri ağlarında bulunan ağırlıklı yasaklama okunun sonlu durum otomatasına dönüşümü bir örnek ile açıklanmaktadır. Şekil 5.12’de bir giriş, bir çıkış ve bir In(p1,t3)=4 ağırlıklı yasaklama okuna sahip ve jeton kapasitesi CAP(*p1*) ≤ 5 olan bir mevkiden oluşan örnek PN modeli ile bu modele ait otomata karşılığı görülmektedir. Bu PN modelindeki t1, t2 ve t3 geçişleri, ilgili olayları karakterize eden ‘a’, ‘b’ ve ‘c’ harfleri ile etiketlenmiştir. p1 mevkisindeki jeton sayısının 5’ten az olması durumunda t1 geçişinin her tetiklenmesi ile p1 mevkisine bir jeton depolanmaktadır. p1 mevkisinde jeton bulunduğu durumda ise t2 geçişinin her bir tetiklenmesi ile p1 mevkisinden bir adet jeton eksiltilmektedir. p1 mevkisindeki jeton sayısının 4’ten az olması durumunda t3 geçişi tetiklemeye açıktır. t3 geçişinin tetiklenmesi p1 mevkisindeki jeton sayısını değiştirmez. Öte yandan p1 mevkisindeki jeton sayısının 4 veya 5 olması durumda ise t3 geçişi tetiklenmeye kapalıdır. p1 mevkisinin jeton kapasitesi 5 olduğu için bu PN modeline ait otomata karşılığında 6 adet durum bulunmaktadır. Bunlardan s0 durumu p1 mevkisinde jeton olmadığını simgeleyen başlangıç (initial state) ve işaretlenmiş (marker state) durumdur. Sırasıyla s1, s2, s3, s4 ve s5 durumları da ise p1 mevkisinde bulunan 1, 2, 3, 4 ve 5 adet jetonu simgelemektedir. [s0,c,s0], [s1,c,s1], [s2,c,s2] ve [s3,c,s3] öz-döngüleri (self-loop) eklenerek s0, s1, s2 ve s3 durumlarında ‘c’ olayının gerçekleşmesi mümkün hale getirilmiştir. Fakat s4 ve s5 durumlarındayken ‘c’ olayının gerçekleşmesine izin verilmemektedir.



Şekil 5.12 Ağırlıklı yasaklama oku içeren örnek bir PN modeli ve otomata karşılığı

5.2.3 Ağırlıklı yetkileme oku için dönüşüm

Bu kısımda, Petri ağlarında bulunan ağırlıklı yetkileme okunun sonlu durum otomatasına dönüşümü bir örnek ile açıklanmaktadır. Şekil 5.13'te bir giriş, bir çıkış, ağırlığı En(p1,t3)=4 olan bir ağırlıklı yetkileme okuna sahip ve jeton kapasitesi CAP(*p1*) ≤ 5 olan bir mevkiden oluşan örnek PN modeli ve bu modele ait otomata karşılığı görülmektedir. Bu PN modelindeki t1, t2 ve t3 geçişleri, ilgili olayları karakterize eden 'a', 'b' ve 'c' harfleri ile etiketlenmiştir. p1 mevkisindeki jeton sayısının 5'ten az olması durumunda t1 geçişinin her tetiklenmesi ile p1 mevkisine bir jeton depolanmaktadır. p1 mevkisinde jeton bulunduğu durumda ise t2 geçişinin her bir tetiklenmesi ile p1 mevkisinden bir adet jeton eksiltilmektedir. p1 mevkisindeki jeton sayısının 4'e eşit ya da 4'ten fazla olması durumunda t3 geçişi tetiklemeye açıktır. t3 geçişinin tetiklenmesi p1 mevkisindeki jeton sayısını değiştirmez. Öte yandan p1 mevkisindeki jeton sayısının 4'ten az olması durumunda ise t3 geçişi tetiklenmeye kapalıdır. p1 mevkisinin jeton kapasitesi 5 olduğu için bu PN modeline ait otomata karşılığında 6 adet durum bulunmaktadır. Bunlardan s0 durumu p1 mevkisinde jeton olmadığını simgeleyen başlangıç (initial state) ve işaretlenmiş (marker state) durumdur. Sırasıyla s1, s2, s3, s4 ve s5 durumları da ise p1 mevkisinde bulunan 1, 2, 3, 4 ve 5 adet jetonu simgelemektedir. [s4,c,s4] ve [s5,c,s5] öz-döngüleri (self-loop) eklenerek s4 ve s5 durumlarında 'c' olayının gerçekleşmesi mümkün hale getirilmiştir. Fakat s0, s1, s2 ve s3 durumlarındayken 'c' olayının gerçekleşmesine izin verilmemektedir.



Şekil 5.13 Ağırlıklı yetkileme oku içeren örnek bir PN modeli ve otomata karşılığı

5.3 Petri Ağı Temelli Denetleyicilerin TCT Kullanılarak Analizi

TCT yazılımı kullanılarak önerilen analiz yöntemi aşağıdaki şekildedir:

1. Kontrol edilmemiş Petri ağı modeli (Uncontrolled Petri Net Model - UPNM), geri besleme kontrol elemanları (kontrol mevkileri kümesi) ve ilgili yasaklanmış durum spesifikasyonlarının verildiği varsayılır.
2. Verilen UPNM'nin sınırlı olduğu ve mevkilerinin sınırları belirlenir.
3. UPNM ve spesifikasyonlar eşdeğer tampon modellerine dönüştürülür. UPNM için elde edilen otomata modeli PLANT, spesifikasyonlar için elde edilen model ise SPEC olarak adlandırılır.
4. Verilen problem için Gözetimli Kontrol Teorisi ile Ramadge-Wonham gözeticisi (RWSUPER), PLANT ve SPEC kullanılarak elde edilir.
5. Verilen geri besleme elemanları (C1, C2, C3, … olarak isimlendirilen kontrol mevkileri) eşdeğer tampon modellerine dönüştürülür.
6. PLANT ve C1, C2, C3, … kontrol mevkileri kullanılarak kontrol edilmiş model PNSUPER elde edilir.
7. RWSUPER ve PNSUPER karşılaştırılarak, eşit olup olmadıkları test edilir.

Bu çalışmada, yasaklanmış durum spesifikasyonlarını sağlayan Petri ağı temelli denetleyicilerin doğruluğunun test edilmesine yönelik yeni bir yöntem önerilmektedir. Ayrıca, önerilen yöntem ile canlılığı (liveness) sağlayan geri besleme elemanların da analizini yapmak mümkündür. Bu tür problemlerde spesifikasyon, başlangıç durumuna bloklanma olmaksızın geri dönebilmek olarak ifade edilir. Sıradan veya ağılıklı giriş çıkış okuna sahip kontrol (monitor) mevkileri, geri besleme elemanı olarak literatürde yaygın biçimde kullanılmaktadır. Buna karşın, bazı problemlerde aynı kontrol işlemi yasaklama oku içeren diğer denetleyicilerle de sağlanabilmektedir. Gözetimli Kontrol

Teorisi için gereken hesaplamalar standart yazılımlarla yapılabilir. Bu çalışmada TCT yazılımı kullanılmıştır. TCT yazılımında kullanılan bazı komutlar ve açıklamaları tezin ikinci kısmında mevcuttur.

İkinci adımda yapılması gereken sınırlılık hesaplaması, standart PN yazılımlarının kullanımıyla veya basit durumlarda gözlemle ya da tamsayı programlamanın (Wonham 2010) el ile uygulamasıyla hesaplanabilir. Üçüncü adımdaki eşdeğer tampon modelleri önceki kısımda açıklanan dönüşümler kullanılarak elde edilebilir. Kontrol edilmemiş PN modelini PLANT'a dönüştürmek için ilk olarak, her bir mevki kendisine bağlı tüm oklar (giriş, çıkış, yasaklama ve yetkileme) dikkate alınarak tampon modellerine dönüştürülür. Sonra, TCT yazılımında bulunan *Create(.)* komutu kullanılarak her bir mevki için elde edilen otomata modeli tanımlanır. Daha sonra tanımlanan otomatalar, *Sync(.)* komutu ile sağlanan senkron çarpım işlemi ile birleştirilerek PLANT elde edilir. PLANT otomata modeli kontrol edilmemiş Petri ağının (UPNM) erişilebilirlik çizgesi ile eş yapıya sahiptir. Yasaklanmış durum probleminde spesifikasyonlar için de bu 3 adım takip edilerek SPEC oluşturulabilir. İlk olarak, spesifikasyon birkaç alt modelden oluşuyorsa (SPEC1, SPEC2, ...) bu modeller *Create(.)* komutu ile tanımlanır. İkinci olarak, PLANT modelinde bulunup ancak spesifikasyonlarda bulunmayan olaylara ait etiketler spesifikasyonlara özdöngü (selfloop) olarak TCT yazılımda bulunan *Selfloop(.)* komutu ile eklenmelidir. Son olarak, bu alt otomata modelleri TCT yazılımdaki *Meet(.)* komutuyla birleştirilerek spesifikasyonlara ait SPEC otomata modeli elde edilir. Eğer problem, canlılığı sağlama problemi (liveness enforcing problem) ise, spesifikasyonların başlangıç durumuna göre bloklanmasız olması gerekmektedir. Bu durumda, bloklanma olmamasını sağlayacak spesifikasyon *Allevents(.)* komutu ile elde edilen otomatadır. *Allevents(.)* komutuyla, dikkate alınan modelde bulunan tüm olay etiketleri ile özdöngülenmiş bir adet mevki elde edilmektedir.

Eğer, problem yasaklanmış durum problemi ise, dördüncü adım; *Supcon(.)* komutu kullanılarak elde edilir: *RWSUPER=Supcon(PLANT, SPEC)*. Eğer problem canlılığı sağlama problemi ise, dördüncü adım, *RWSUPER=Supcon(PLANT,ALL)* işlemi ile yerine getirilir. Bu işlemlerin her ikisinde de elde edilen RWSUPER yapıları *Supcon(.)* komutu ile hesaplandığından doğal olarak mükemmeldir (trim, kontrol edilebilir ve bloklanmasızdır). Ayrıca, *Supcon(.)* komutu sonucunda elde edilen RWSUPER'in maksimum müsaade edici olduğu da Wonham tarafından açıkça gösterilmiştir [71]. PN

temelli geri besleme elemanlarının doğruluğu elde edilen mükemmel (proper) RWSUPER gözeticileri referans alınarak test edilmektedir.

Beşinci adımda gerekli olan dönüşümler için önceki kısımda önerilen haritalamalar kullanılmaktadır. PN modellerindeki geri besleme elemanlarının dönüşümü için ilk olarak, her bir geri besleme elemanı ilgili ok bağlantıları (giriş, çıkış, yetkileme ve yasaklama okları) dikkate alınarak eşdeğer tampon modellerine çevrilir. Daha sonra bu otomata modelleri C1, C2, C3, ... olarak adlandırılarak *Create(.)* komutu ile tanımlanır.

Altıncı adımda, kontrol edilmiş PNSUPER modeli, PLANT ve C1, C2, C3, ... yardımıyla elde edilir. Bu işlem için senkron çarpım komutu *Sync(.)* kullanılır. PNSUPER, PLANT'in PN temelli denetleyicinin gözetimi altındaki kontrol edilmiş davranışını simgelemektedir.

Son adımda, RWSUPER ve PNSUPER'in eş olup olmadıkları test edilir. Bu işlemi yapmak için *Isomorph(RWSUPER, PNSUPER)* komutu kullanılır. Bu komut, iki yapının eş olması durumunda true(doğru), eş olmaması durumunda ise false(yanlış) çıkışı üretmektedir. Test sonucunun doğru olması, PN temelli geri besleme elemanın doğruluğunu kanıtlamaktadır. Bu durumda PN temelli geri besleme elemanından elde edilen PNSUPER mükemmel (proper) dir yani maksimum müsaade edici ve bloklanmasızdır. Eğer canlılığı sağlama problemi ile ilgileniliyorsa, tüm geçişlerin kontrol edilebilir olduğu durumda, *TPLANT=Trim(PLANT)* komutu ile elde edilen trim PLANT modeli, PNSUPER(veya RWSUPER) ile eş yapılı olacaktır. Buna karşın, kontrol edilebilir ve kontrol edilemeyen geçişlerin her ikisinin de bulunduğu durumda, trim PLANT ile PNSUPER (veya RWSUPER) eş yapılı değildir. Bu durumda, RWSUPER, maksimum müsaade edici ve bloklanmasız davranışı tanımlayan en iyi şekilde kontrol edilebilir bir alt dil (supremal controllable sublanguage) dir. Isomorph komutu ile G1 ve G2 olarak adlandırılan iki AOS'nin kapalı (closed) ve işaretlenmiş (marked) davranışının eşitliği test edilebilir. Eğer *Isomorph(G2,G2)=true* ise L(G1)=L(G2) ve Lm(G1)=Lm(G2)'dir. Buna karşın, eğer, *Isomorph(G2,G2)=False* ise halen dil eşitliğinden söz edilebilir. Bunun için ikinci bir test işlemi gerekmektedir. MGi=Minstate(Gi) olmak üzere, *Isomorph(MG1,MG2)= true* ise (*Isomorph(G2,G2)* testinin sonucunun False olduğu durumda) dil eşitliği halen sağlanıyor demektir. Ancak,

Isomorph(MG1,MG2)=false ise (Isomorph(G2, G2) testinin sonucunun false olduğu durumda) dil eşitliğinde hata vardır. Bu durumlar [71]'de detaylı olarak açıklanmıştır.

Buraya kadar, PN temelli optimal denetleyicilerin doğruluk analizinin nasıl yapılabileceği açıklanmıştır. Ancak, PN literatüründe verilen problem için PN tabanlı optimum altı çözümlerle de karşılaşılabilir. PN temelli optimum altı denetleyicilerin doğruluk analizi yapılırken de RW tipi denetleyicilerin referans olarak kullanılması mümkündür. Bu doğruluk analizinin yapılabilmesi için yukarıda açıklanan işlemler yedinci adıma kadar takip edilir. Bu durumda, PN temelli kontrol edilmiş modelin ürettiği dilin, RWSUPER'in bir alt dili olduğu görülmektedir. İlk olarak PNSUPER'in ürettiği alt dilin RWSUPER ile ikilem oluşturup oluşturmadığı *Nonconflict(PNSUPER,RWSUPER)* komutu ile test edilir. Bu testin sonucunun true (doğru) olması gerekir. İkinci olarak, PNSUPER ve RWSUPER'in erişilebilir kartezyen çarpımı *MRW_PN=Meet(RWSUPER,PNSUPER)* komutu ile hesaplanır. Sonuç ayrık olay sistemi olan MRW_PN, PNSUPER'e eşit olmalıdır. Çünkü, bu durumda, herhangi bir optimum altı çözüm olan PNSUPER'in, optimum çözüm olan RWSUPER'in bir alt dili olması beklenir. Kısaca, *Isomorph(MRW_PN, PNSUPER)=true* olması gerekir.

5.4 Uygulama Örnekleri

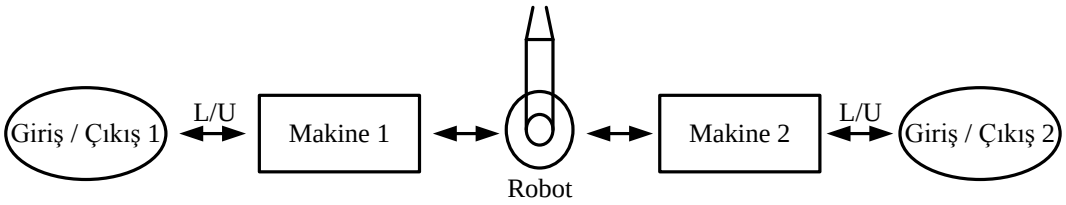
Bu kısımda, önerilen analiz metodunun uygulanabilirliğini göstermek üzere üç örnek dikkate alınmıştır. Detaylı hesaplama adımları her bir örnek için TCT log dosyası biçiminde gösterilmiştir.

5.4.1 Örnek 1

İlk örnek olarak kilitlenme (canlılığı sağlama) problemi bulunan basit bir esnek üretim sistemi (Flexible Manufacturing System - FMS) dikkate alınmıştır. Kilitlenme problemini çözen iki farklı denetleyicinin doğru olduğu gösterilmiştir. Bu esnek üretim sistemi Şekil 5.14'te görüldüğü gibi, bir anda sadece bir parça işleyebilen Makine 1 ve 2 ile bir anda sadece bir parça taşıyabilen bir robottan oluşmaktadır. Parçalar FMS'e yükleme/boşaltma (L/U) birimleri yardımıyla aktarılmaktadır. Sistemde P1 ve P2 olmak üzere iki farklı parça işlenmektedir. Başlangıçta sistemde her hangi bir parça olmadığı kabul edilmektedir. İki farklı parça için üretim işlem sıraları aşağıdaki gibidir:

P1: M1 → Robot → M2

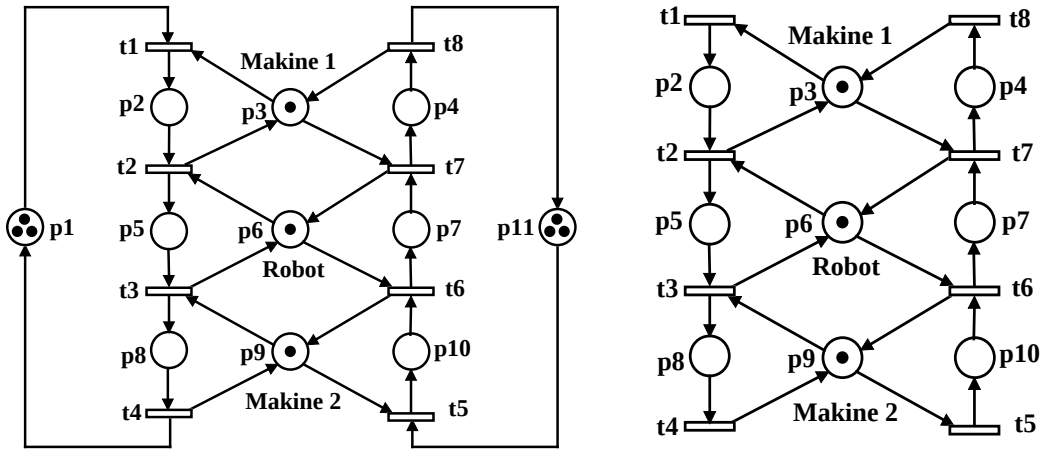
P2: M2 → Robot → M1



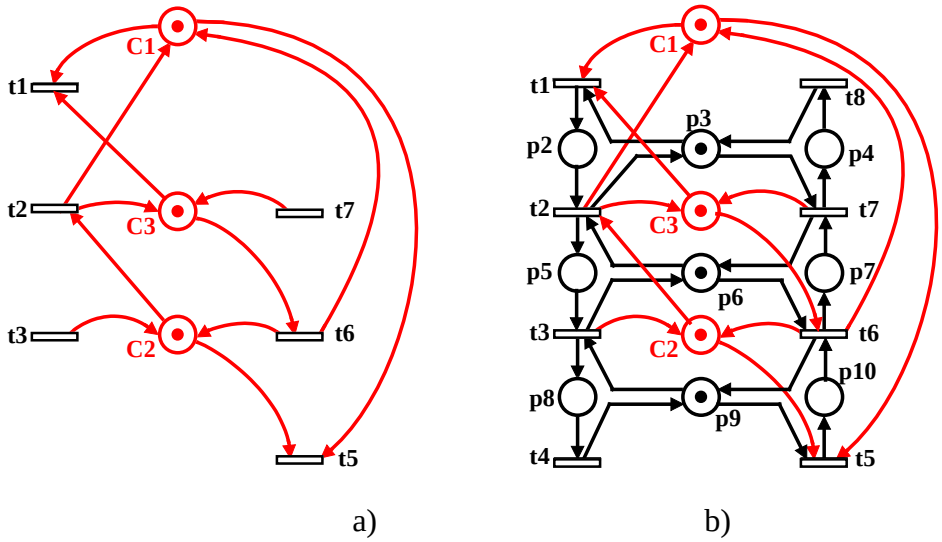
Şekil 5.14 Örnek bir FMS

Verilen üretim işlemleri için oluşturulmuş PN modeli Şekil 5.15.(a)'da görülmektedir [44]. Bu modelde 11 mevki P = {p1, p2, …, p11} ve 8 geçiş T = {t1, t2, …, t8} bulunmaktadır. Tüm geçişlerin kontrol edilebilir olduğu kabul edilmektedir. p2, p5 ve p8 mevkileri sırası ile M1, Robot ve M2'nin P1 üretim işlemleri için çalışmasını ifade etmektedir. p1 mevkisinde bulunan M(p1) = 3 jeton işaretlemesi P1 üretim işlemleri için eşzamanlı olarak yapılabilecek aktivitelerin sayısını belirtmektedir. Benzer olarak p10, p7 ve p4 mevkileri M2, Robot ve M1'in P2 üretim işlemleri için çalışmasını ifade etmektedir. p11 mevkisinde bulunan M(p11) = 3 jeton işaretlemesi P2 üretim işlemleri için eşzamanlı olarak yapılabilecek aktivitelerin sayısını belirtmektedir. p3, p6 ve p9 mevkileri ise paylaşılan kaynakları M1, Robot and M2 ifade etmektedir. Bu üretim sisteminde her bir üretim işleminin tamamlanması beklenmektedir. Bu sistemde kilitlenme sorunu olduğu gösterilmiştir. Bu sistem için kontrol spesifikasyonu sistemin canlı (live) olmasıdır. Bu özellik SCT terminolojisinde bloklanmasız (nonblocking) çalışma olarak adlandırılmaktadır. Modelde bulunan p1 ve p11 mevkileri, bu modelden kaldırıldığında bir problem oluşturmazlar. Bu mevkilerin modelden kaldırılmış biçimi Şekil 5.15.(b)'de görülmektedir.

FMS'e ait SOPN modelinde de kilitlenme mevcuttur. Bu modelin erişilebilirlik çizgesinde 20 durum vardır. Bunlardan 15 tanesi sistemin optimal ve canlı davranışını simgeleyen iyi durumlardır. Geriye kalan 5 durum ise 2 tanesi kilitlenme durumu olmak üzere kötü durumlardır. Bu FMS için [44]'te elde edilen maksimum müsaade edici (maximally permissive) denetleyici Şekil 5.16.(a)'da görülmektedir. Bu denetleyicinin kontrol edilmemiş model ile birleştirilmiş hali Şekil 5.16.(b)'de görülmektedir. Kontrol edilmiş modelin canlı ve maksimum müsaade edici olduğu doğrulanmıştır.



Şekil 5.15 a) FMS için PN modeli, b) FMS için oluşturulan SOPN modeli



Şekil 5.16 a) [44]'te elde edilen denetleyici, b) Sistemin kontrol edilmiş modeli

Bu örnek için, önceki kısımda önerilen analiz yöntemi aşağıdaki gibi uygulanmıştır:

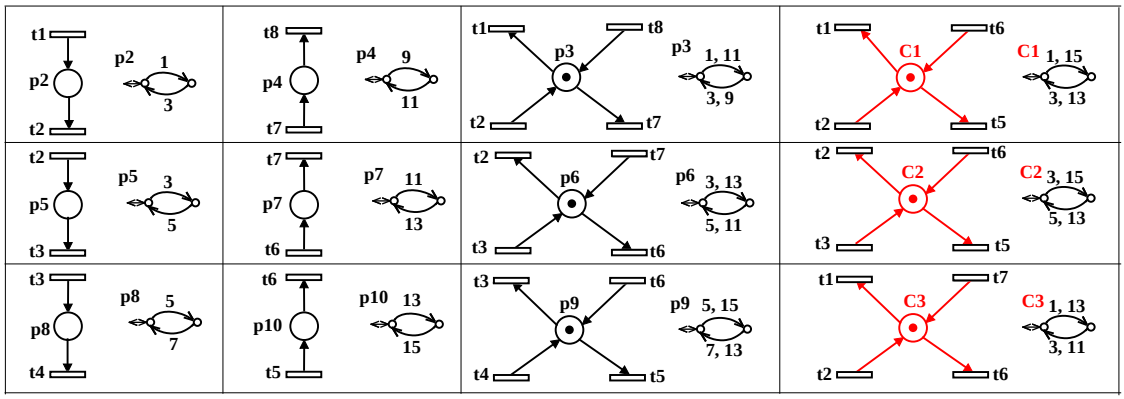
1. Adım: Kontrol edilmemiş PN modeli (UPNM) Şekil 5.15.(b)'de görülmektedir. Bu örnekte geri besleme elemanı Şekil 5.16.(a)'da görülen üç adet kontrol mevkisidir. Kontrol spesifikasyonları UPNM'nin başlangıç durumuna göre bloklanmasız olmasıdır.

2. Adım: UPNM modelinde bulunan mevkilerin jeton kapasiteleri p2:1, p3:1, p4:1, p5:1, p6:1, p7:1, p8:1, p9:1 ve p10:1 olmak üzere sınırlıdır. UPNM'nin SOPN olduğu görülmektedir.

3.Adım: UPNM modelinde bulunan her bir mevkinin otomata eşdeğeri Şekil 5.17'de görülmektedir. TCT yazılımında olay etiketleri rakamlardan oluşmaktadır. Kontrol edilebilir olaylar için tek rakamlar, kontrol edilemeyen olaylar için ise çift rakamlar kullanılmaktadır. TCT yazılımındaki bu gereksinim nedeni ile PN modelindeki olay etiketleri için aşağıda yazılı rakam kodlamaları kullanılmıştır.

FMS için TCT olay kodlaması:

FMS:	t1	t2	t3	t4	t5	t6	t7	t8
TCT:	1	3	5	7	15	13	11	9



Şekil 5.17 FMS için PN modelindeki mevkiler ve otomata dönüşümleri

Doğrulama işlemi için, aşağıdaki işlemler TCT yazılımı kullanılarak gerçekleştirilmiştir. Burada; PLANT, sistemin kontrol edilmemiş davranışını yani kontrol edilmemiş sistemi tanımlamaktadır.

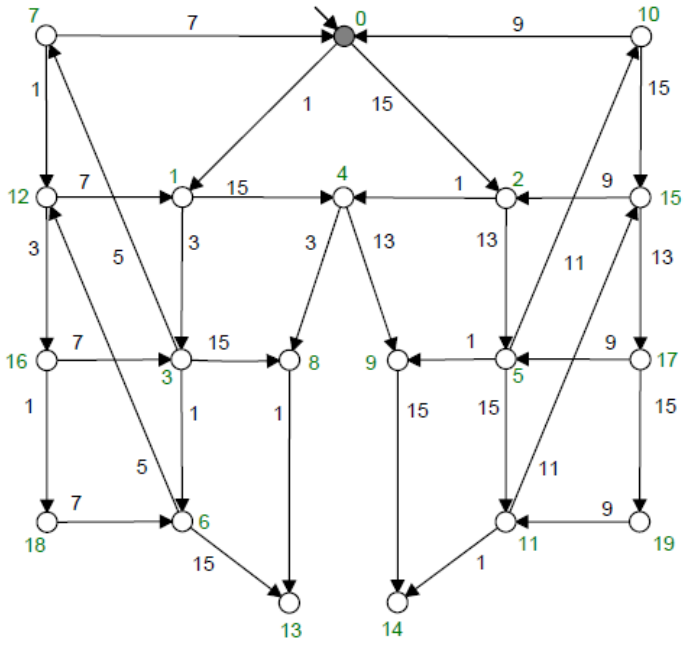
İlk olarak PN modelindeki her bir mevki aşağıdaki gibi otomata olarak oluşturulmuştur.

p2 = Create(p2,[mark 0],[tran [0,1,1],[1,3,0]]) (2,2)
p3 = Create(p3,[mark 0],[tran [0,1,1],[0,11,1],[1,3,0],[1,9,0]]) (2,4)
p4 = Create(p4,[mark 0],[tran [0,11,1],[1,9,0]]) (2,2)
p5 = Create(p5,[mark 0],[tran [0,3,1],[1,5,0]]) (2,2)
p6 = Create(p6,[mark 0],[tran [0,3,1],[0,13,1],[1,5,0],[1,11,0]]) (2,4)
p7 = Create(p7,[mark 0],[tran [0,13,1],[1,11,0]]) (2,2)
p8 = Create(p8,[mark 0],[tran [0,5,1],[1,7,0]]) (2,2)

p9 = Create(p9,[mark 0],[tran [0,5,1],[0,15,1],[1,7,0],[1,13,0]]) (2,4)
p10 = Create(p10,[mark 0],[tran [0,15,1],[1,13,0]]) (2,2)

Bu işlemlerin ardından PLANT tüm mevkilerin senkron çarpımı PLANT = p2||p3||p4||p5||p6||p7||p8||p9||p10 alınarak aşağıda görüldüğü gibi oluşturulmuştur. Oluşturulan PLANT'ın otomata gösterimi Şekil 5.18'de verilmiştir.

PLANT = Sync(P2,P3) (4,6) Blocked_events = None
PLANT = Sync(PLANT,P4) (3,4) Blocked_events = None
PLANT = Sync(PLANT,P5) (6,10) Blocked_events = None
PLANT = Sync(PLANT,P6) (12,20) Blocked_events = None
PLANT = Sync(PLANT,P7) (8,12) Blocked_events = None
PLANT = Sync(PLANT,P8) (16,30) Blocked_events = None
PLANT = Sync(PLANT,P9) (32,60) Blocked_events = None
PLANT = Sync(PLANT,P10) (20,34) Blocked_events = None



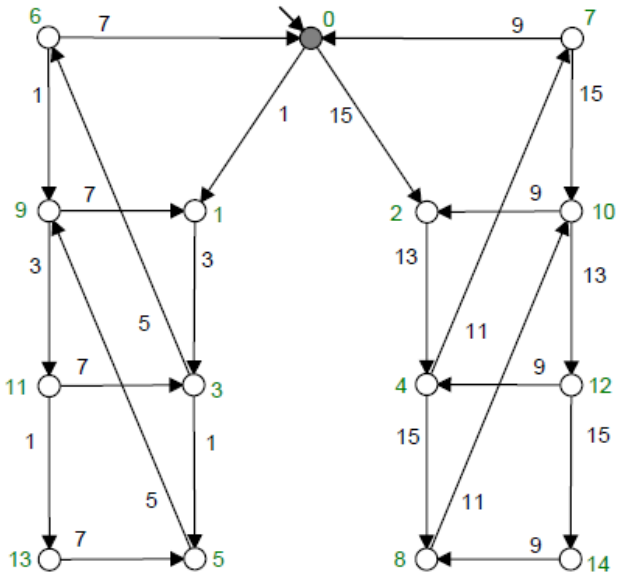
Şekil 5.18 FMS'in kontrol edilmemiş PN modelinin otomata karşılığı

Bu problem, canlılığı sağlama problemdir. Bu nedenle spesifikasyon, başlangıç durumuna göre bloklanmasızlığın sağlanmasıdır. Spesifikasyon, *ALL=Allevents(PLANT) (1,8)* komutu ile elde edilmektedir. Ayrıca aşağıdaki testten de görüldüğü gibi kontrolün yokluğunda sistem bloklanmaktadır.

False=Nonconflict(PLANT;ALL).

4. Adım: Bu problem için RW tipi gözetici aşağıdaki gibi sentezlenmiştir. Elde edilen RW tipi gözetici RWSUPER'in otomata modeli Şekil 5.19'da görülmektedir.

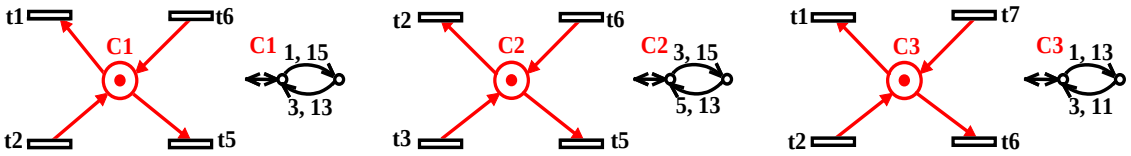
RWSUPER = Supcon(PLANT,ALL) (15,24)



Şekil 5.19 Örnek FMS için hesaplanan RW tipi gözetici RWSUPER'in otomata modeli

5. Adım: Verilen geri besleme elemanları (kontrol mevkileri) eşdeğer otomata modellerine Şekil 5.20'de görüldüğü dönüştürülmüştür. Kontrol mevkileri TCT yazılımında aşağıdaki gibi oluşturulur.

C1 = Create(C1,[mark 0],[tran [0,1,1],[0,15,1],[1,3,0],[1,13,0]]) (2,4)
C2 = Create(C2,[mark 0],[tran [0,3,1],[0,15,1],[1,5,0],[1,13,0]]) (2,4)
C3 = Create(C3,[mark 0],[tran [0,1,1],[0,13,1],[1,3,0],[1,11,0]]) (2,4)



Şekil 5.20 Kontrol mevkilerinin otomata karşılıkları

6. Adım: Kontrol edilmiş model PNSUPER, kontrol mevkileri kullanılarak CM=PLANT||C1||C2||C3 işlemi ile aşağıda görüldüğü gibi oluşturulur.

PNSUPER = Sync(PLANT,C1) (17,26) Blocked_events = None
PNSUPER = Sync(CM,C2) (16,25) Blocked_events = None

PNSUPER = Sync(CM,C3) (15,24) Blocked_events = None

7. Adım: Son olarak, aşağıdaki test işlemi kullanılarak, kontrol edilmiş modelin (PNSUPER), gözeticiye (RWSUPER) eşit olduğu gösterilmiştir. Bu, PN domeninde elde edilmiş olan gözeticinin mükemmel (proper) gözetici olduğu ve ayrıca bloklanmasız ve maksimum müsaade edici (maximally permissive) olduğu anlamına gelmektedir.

true = Isomorph(PNSUPER,RWSUPER;identity)

Aşağıdaki işlemlerden görülebileceği gibi optimal kontrol edilmiş PNSUPER daha önce açıklandığı gibi, basitçe trim edilmiş PLANT tarafından üretilen bir dildir. Bu özellik, tüm geçişlerin kontrol edilebilir olduğu durumda geçerlidir.

TPLANT = Trim(PLANT) (15,24)
true = Isomorph(TPLANT,PNSUPER;identity)

Bu sistem için alternatif bir denetleyici oluşturulabilir. Kontrol mevkileri C1, C2 ve C3'ün detayları incelendiğinde, p2 veya p5 mevkilerinde jeton olduğunda t5 geçişinin tetiklenmeye müsaade edilmemekte olduğu yani yasaklanmakta olduğu anlaşılır. Benzer olarak, p7 veya p10 mevkilerinde jeton olduğunda t1 geçişi tetiklenmeye müsaade edilmemektedir yani yasaklanmaktadır. Sonuç olarak, bu kontrol işlemleri Şekil 5.21.(a)'da görülen yapılar kullanılarak gerçekleştirilebilir. Bu kontrol yapıları, sistemin kontrol edilmemiş modeli ile birleştirilerek Şekil 5.21.(b)'de görülen kontrol edilmiş SOPN modeli elde edilir.

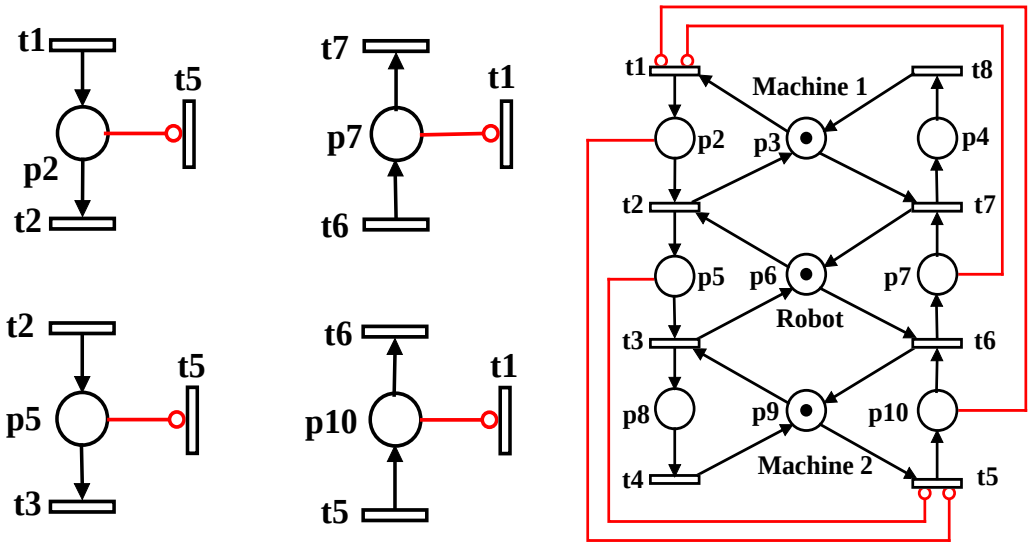
Bu yeni denetleyici yapısının doğruluğunun testi, yukarıda önerilen yaklaşımın 4. adımdan sonrası takip edilerek aşağıdaki gibi yapılabilir.

5. Adım: Alternatif denetleyicinin PN modelindeki her bir mevkinin Şekil 5.22'de görülen otomata karşılıkları aşağıdaki gibi TCT'de tanımlanmıştır.

SP1 = Create(SP1,[mark 0],[tran [0,1,1],[0,15,0],[1,3,0]]) (2,3)
SP2 = Create(SP2,[mark 0],[tran [0,3,1],[0,15,0],[1,5,0]]) (2,3)
SP3 = Create(SP3,[mark 0],[tran [0,1,0],[0,13,1],[1,11,0]]) (2,3)
SP4 = Create(SP4,[mark 0],[tran [0,1,0],[0,15,1],[1,13,0]]) (2,3)

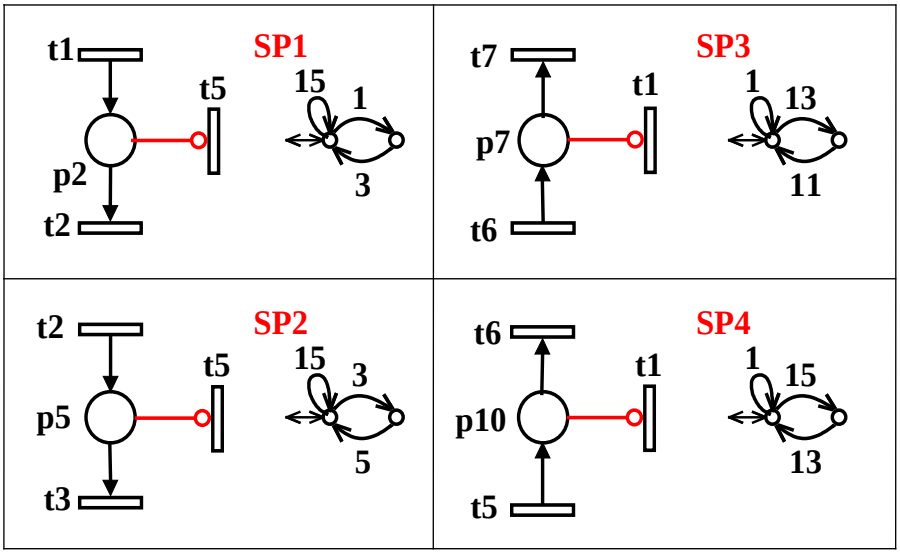
6. Adım: Kontrol edilmiş model CM1, CM1=PLANT||SP1||SP2||SP3||SP4 işlemi ile aşağıdaki gibi oluşturulmaktadır.

CM1 = Sync(PLANT,SP1) (20,31) Blocked_events = None
CM1 = Sync(CM1,SP2) (20,30) Blocked_events = None
CM1 = Sync(CM1,SP3) (19,28) Blocked_events = None
CM1 = Sync(CM1,SP4) (15,24) Blocked_events =None



a) b)

Şekil 5.21 a) FMS için alternatif bir denetleyici,
b) Alternatif denetleyici ile sistemin kontrol edilmiş modeli



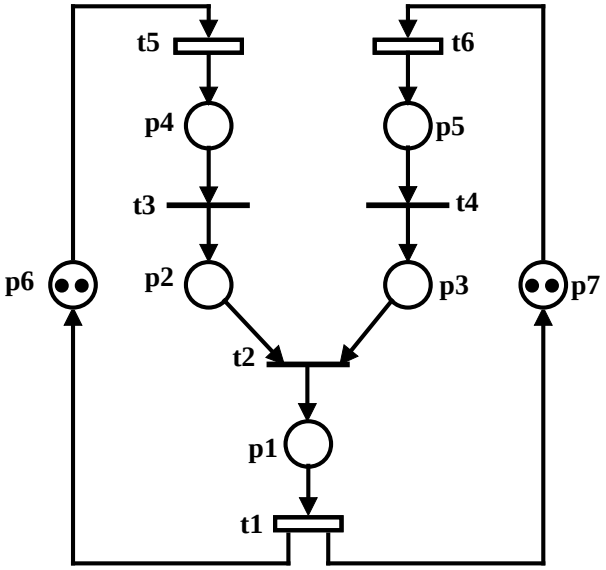
Şekil 5.22 FMS için oluşturulan alternatif denetleyici mevkilerinin otomata karşılıkları

7. Adım: Son olarak, aşağıdaki test işlemi kullanılarak, kontrol edilmiş yeni modelin (CM1), gözeticiye (RWSUPER) eşit olduğu gösterilmiştir. Bu yeni denetleyici aynı zamanda PNSUPER'e de eşittir. Bir başka deyişle, Şekil 5.16.(b) ve 5.21.(b)'de görülen her iki kontrol edilmiş model aynı dili üretmektedir. Yani aynı spesifikasyonu sağlamaktadırlar. Bu bağlamda bu iki model kontrol eşdeğerdir denilmektedir.

true = Isomorph(RWSUPER,CM1;identity)

5.4.2 Örnek 2

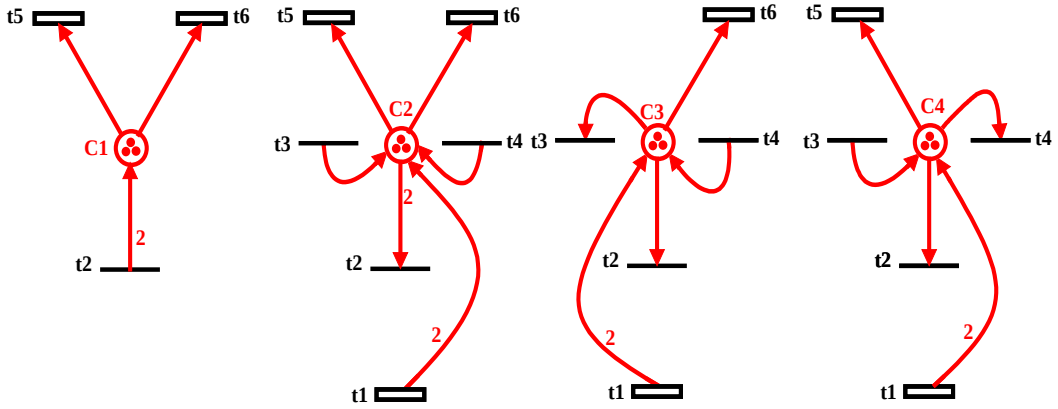
İkinci örnek olarak, Petri ağı modellerinde uygulanan genelleştirilmiş karşılıklı dışlama kısıtı (GMEC - generalised mutual exclusion constraint) incelenmiştir. Bu örnek [51]'den alınmış olup kontrol edilmemiş Petri ağı modeli (UPNM) Şekil 5.23'te görülmektedir. Bu modelde P = {p1, p2, ..., p7} olmak üzere yedi mevki ve T = {t1, t2, ..., t6} olmak üzere altı geçiş bulunmaktadır. Başlangıçtaki jeton işaretlemesi M_0 = (0, 0, 0, 0, 0, 2, 2)^T'dir. UPNM modeli, sınırlı, canlı (live), tersinir (reversible) ve korunumludur (conservative). Bu modeldeki t1, t5 ve t6 geçişleri kontrol edilebilir geçişler olup içi boş dikdörtgen ile gösterilmiştir. Bu geçişler denetleyici tarafından yasaklanabilir veya yetkilendirilebilir. Modelde bulunan t2, t3 ve t4 geçişleri kontrol edilemeyen geçişlerdir. Bu geçişlere denetleyicinin bir müdahalesi söz konusu değildir. Bu örnek için üç farklı çözüm incelenmiştir. Bunlar iki farklı optimal çözüm ve bir optimal altı çözümdür.



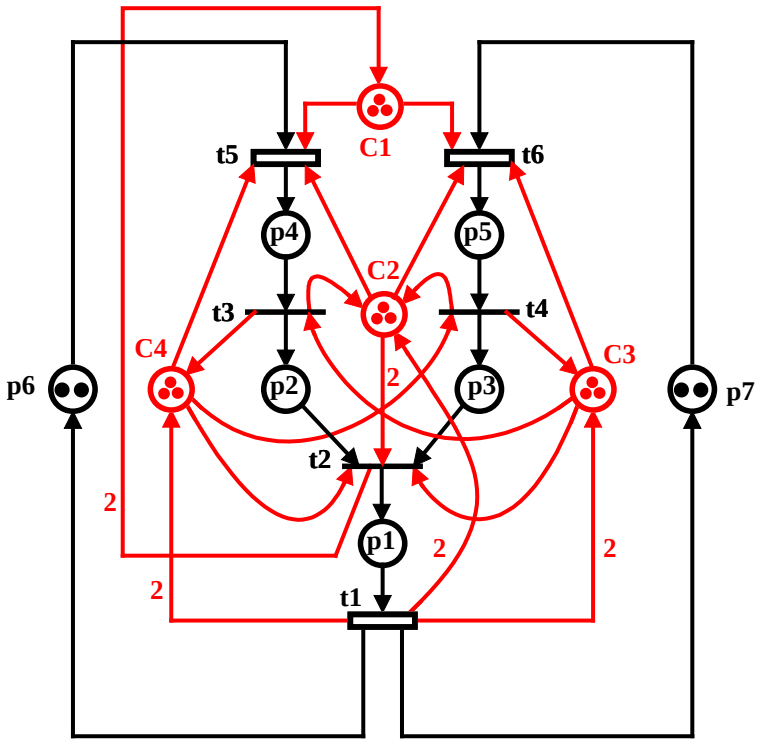
Şekil 5.23 Kontrol edilebilir (t1, t5, t6) ve kontrol edilemeyen (t2, t3, t4) geçişlere sahip bir PN modeli [51]

5.4.2.1 Örnek 2 - optimal çözüm 1

Şekil 5.23'te görülen PN modelinde $m(p_1) \leq 1$ GMEC şartını yerine getiren kontrol mevkileri C1, C2, C3 ve C4, Şekil 5.24'te görülmektedir [94]. Şekil 5.25'te görülen kontrol edilmiş PN modelinin ürettiği kontrol edilebilir alt dil 32 durum ve 62 geçişe sahiptir. Kısaca, Şekil 5.25'te görülen kontrol edilmiş model maksimum müsaade edici ve bloklanmasızdır.



Şekil 5.24 Şekil 5.23'te görülen PN modelinde $m(p_1) \leq 1$ kısıtını yerine getiren dört adet kontrol mevkisi [94]



Şekil 5.25 Şekil 5.23'te görülen modele eklenen kontrol mevkileri ile elde edilen kontrol edilmiş model [94]

Bu örnek için, elde edilmiş olan C1, C2, C3 ve C4 kontrol mevkilerinin doğruluğu aşağıdaki işlemler yapılarak test edilmiştir.

1. Adım: UPNM Şekil 5.23'te görülen modeldir. Bu modeldeki geri besleme elemanları ise Şekil 5.24'te görülen C1, C2, C3 ve C4 kontrol mevkileridir. Spesifikasyon ise $m(p_1) \leq 1$ yani p1 mevkisindeki jeton sayısının en çok 1 olmasıdır.

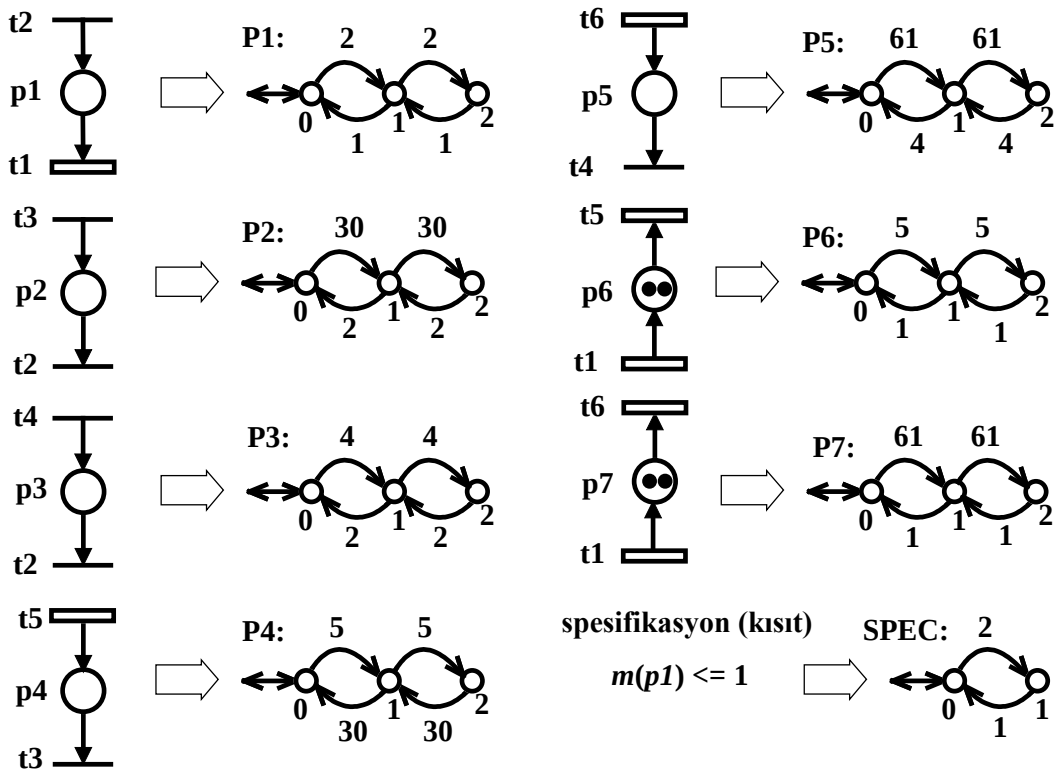
2. Adım: UPNM modelindeki mevkilerin jeton kapasiteleri p1:2, p2:2, p3:2, p4:2, p5:2, p6:2 ve p7:2 olmak üzere sınırlıdır.

3. Adım: UPNM modelindeki mevkiler ve bu mevkilerin önerilen dönüşümler kullanılarak oluşturulan otomata karşılıkları Şekil 5.26'da görülmektedir. Spesifikasyonun ($m(p_1) \leq 1$) da otomata olarak ifadesi yine bu şekilde verilmiştir. TCT yazılımında olay etiketleri rakamlardan oluşmaktadır. Kontrol edilebilir olaylar için tek rakamlar, kontrol edilemeyen olaylar için ise çift rakamlar kullanılmaktadır. Buna göre UPNM'deki geçişler TCT yazılımında kullanılmak üzere aşağıda görüldüğü gibi kodlanmıştır:

UPNM:	t1	t2	t3	t4	t5	t6
TCT:	1	2	30	4	5	61

Bu problemin SCT yardımıyla çözümü için öncelikle TCT yazılımı kullanılarak Create(.) komutuyla tüm mevkiler ve spesifikasyon oluşturulur:

```
P1 = Create(P1,[mark 0],[tran [0,2,1],[1,1,0],[1,2,2],[2,1,1]])  (3,4)
P2 = Create(P2,[mark 0],[tran [0,30,1],[1,2,0],[1,30,2],[2,2,1]])  (3,4)
P3 = Create(P3,[mark 0],[tran [0,4,1],[1,2,0],[1,4,2],[2,2,1]])  (3,4)
P4 = Create(P4,[mark 0],[tran [0,5,1],[1,5,2],[1,30,0],[2,30,1]])  (3,4)
P5 = Create(P5,[mark 0],[tran [0,61,1],[1,4,0],[1,61,2],[2,4,1]])  (3,4)
P6 = Create(P6,[mark 0],[tran [0,5,1],[1,1,0],[1,5,2],[2,1,1]])  (3,4)
P7 = Create(P7,[mark 0],[tran [0,61,1],[1,1,0],[1,61,2],[2,1,1]])  (3,4)
```



Şekil 5.26 İkinci örneğe ait UPNM modelindeki mevkiler ve otomata dönüşümleri

Bu işlemlerin ardından PLANT tüm mevkilerin senkron çarpımı (PLANT=p1||p2||p3||p4||p5||p6||p7) alınarak aşağıda görüldüğü gibi oluşturulmaktadır.

PLANT = Sync(P1,P2) (9,16) Blocked_events = None
PLANT = Sync(PLANT,P3) (27,62) Blocked_events = None
PLANT = Sync(PLANT,P4) (81,222) Blocked_events = None
PLANT = Sync(PLANT,P5) (243,774) Blocked_events = None
PLANT = Sync(PLANT,P6) (90,232) Blocked_events = None
PLANT = Sync(PLANT,P7) (46,104) Blocked_events = None

Spesifikasyon $m(p_1) \leq 1$ yani p1 mevkisindeki jeton sayısının en çok 1 olmasıdır. Bu spesifikasyona ait otomata modeli Şekil 5.26'da görülmektedir. Bu otomata aşağıdaki TCT komutu ile tanımlanmıştır.

SPEC = Create(SPEC,[mark 0],[tran [0,2,1],[1,1,0]]) (2,2)

Bu adımların ardından, PLANT modelinde bulunan ancak spesifikasyon tarafından kısıtlanmayan olay etiketleri her duruma öz döngü olarak eklenmelidir. Bu olaylar, '4', '5', '30' ve '61' etiketli olaylar olup aşağıdaki TCT komutu kullanılarak SPEC modelindeki diğer durumlara eklenmiştir.

SPEC = Selfloop(SPEC,[4,5,30,61]) (2,10)

4. Adım: RWSUPER aşağıdaki gibi elde edilmiştir.

RWSUPER = Supcon(PLANT,SPEC) (32,62)

Aşağıda detayları verilen RWSUPER denetleyicisinin otomata modeli Şekil 5.27'de görülmektedir.

RWSUPER # states: 32 state set: 0 ... 31 initial state: 0
marker states: 0 vocal states: none # transitions: 62
transitions:
[0,5,1][0,61,2][1,5,3][1,30,4][1,61,5][2,4,6][2,5,5][2,61,7][3,30,8][3,61,9][4,5,8]
[4,61,10][5,4,11][5,5,9][5,30,10][5,61,12][6,5,11][6,61,13][7,4,13][7,5,12][8,30,14]
[8,61,15][9,4,16][9,30,15][10,4,17][10,5,15][10,61,18][11,5,16][11,30,17][11,61,19]
[12,4,19][12,30,18][13,4,20][13,5,19][14,61,21][15,4,22][15,30,21][16,30,22][17,2,23]
[17,5,22][17,61,24][18,4,24][19,4,25][19,30,24][20,5,25][21,4,26][22,2,27][22,30,26]
[23,1,0][23,5,27][23,61,28][24,2,28][24,4,29][5,30,29][26,2,30][27,1,1][27,30,30]
[28,1,2][28,4,31][29,2,31][30,1,4][31,1,6]

5. Adım: Şekil 5.24'te görülen C1, C2, C3 ve C4 kontrol mevkileri eşdeğer tampon otomata modellerine Şekil 5.28'de görüldüğü gibi dönüştürülür. Bu geri besleme elemanları aşağıdaki TCT komutları yardımıyla tanımlanmıştır.

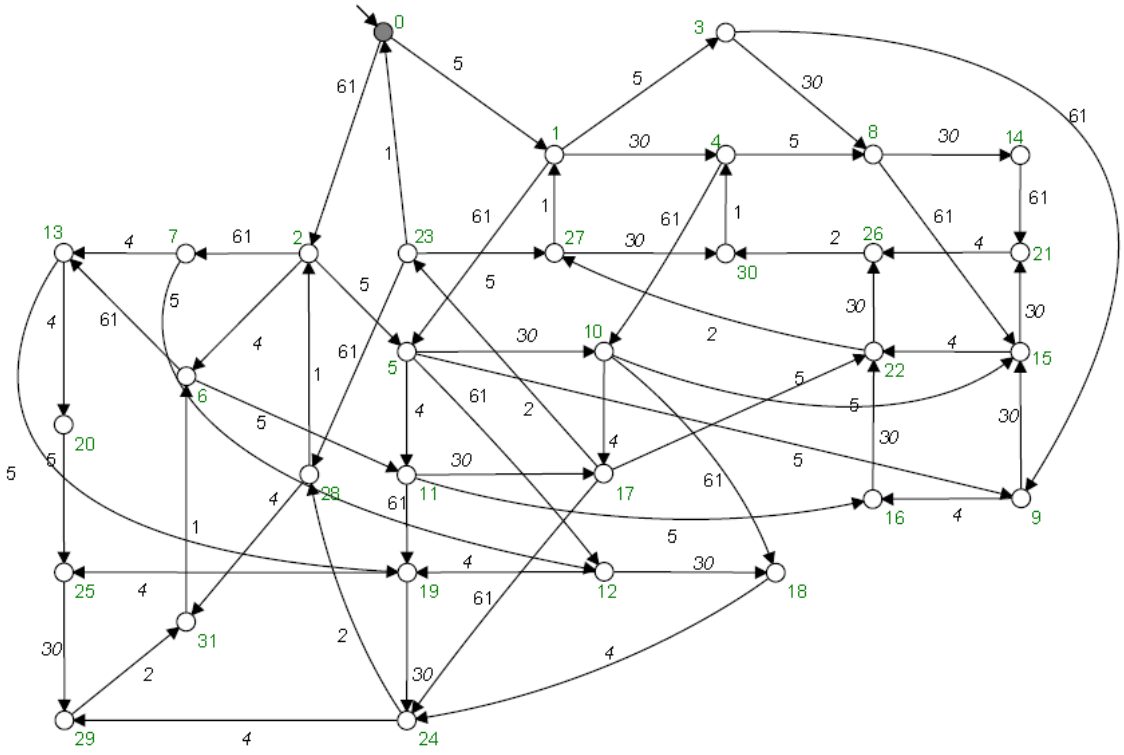
C1 = Create(C1,[mark 0], [tran [0,5,1], [0,61,1], [1,5,2], [1,61,2], [2,2,0], [2,5,3], [2,61,3], [3,2,1]]) (4,8)
C2 = Create(C2,[mark 0],[tran [0,2,2], [0,5,1], [0,61,1], [1,2,3], [1,4,0], [1,5,2], [1,30,0], [1,61,2], [2,1,0], [2,4,1], [2,5,3], [2,30,1], [2,61,3], [3,1,1], [3,4,2], [3,30,2]]) (4,16)
C3 = Create(C3,[mark 0], [tran [0,2,1], [0,30,1], [0,61,1], [1,2,2], [1,4,0], [1,30,2], [1,61,2], [2,1,0], [2,2,3], [2,4,1], [2,30,3], [2,61,3],[3,1,1],[3, 4,2]]) (4,14)
C4 = Create(C4,[mark 0], [tran [0,2,1], [0,4,1], [0,5,1], [1,2,2], [1,4,2], [1,5,2], [1,30,0], [2,1,0], [2,2,3], [2,4,3], [2,5,3], [2,30,1], [3,1,1], [3,30,2]]) (4,14)

6. Adım: PLANT modeli ve geri besleme elemanları kullanılarak (PNSUPER1 = PLANT||C1||C2||C3||C4) senkron kompozisyon işlemi ile kontrol edilmiş model aşağıdaki gibi elde edilir.

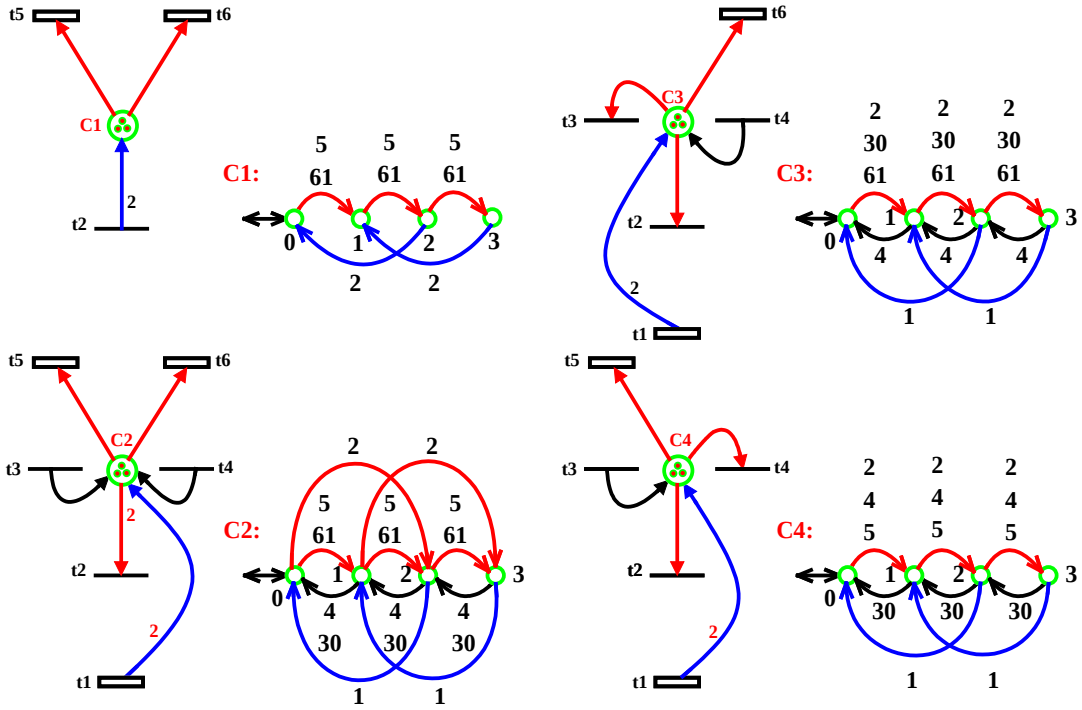
PNSUPER1 = Sync(PLANT,C1) (37,76) Blocked_events = None
PNSUPER1 = Sync(PNSUPER1,C2) (35,69) Blocked_events = None
PNSUPER1 = Sync(PNSUPER1,C3) (34,66) Blocked_events = None
PNSUPER1 = Sync(PNSUPER1,C4) (32,62) Blocked_events = None

7. Adım: Aşağıdaki test işlemleri ile kontrol edilmiş model PNSUPER1'in RWSUPER ile eş yapılı olduğu gösterilmiştir. Bunun anlamı, PN domeninde elde edilmiş olan denetleyicinin mükemmel yani bloklanmasız ve maksimum müsaade edici olduğudur.

true = Isomorph(PNSUPER1,RWSUPER;identity)



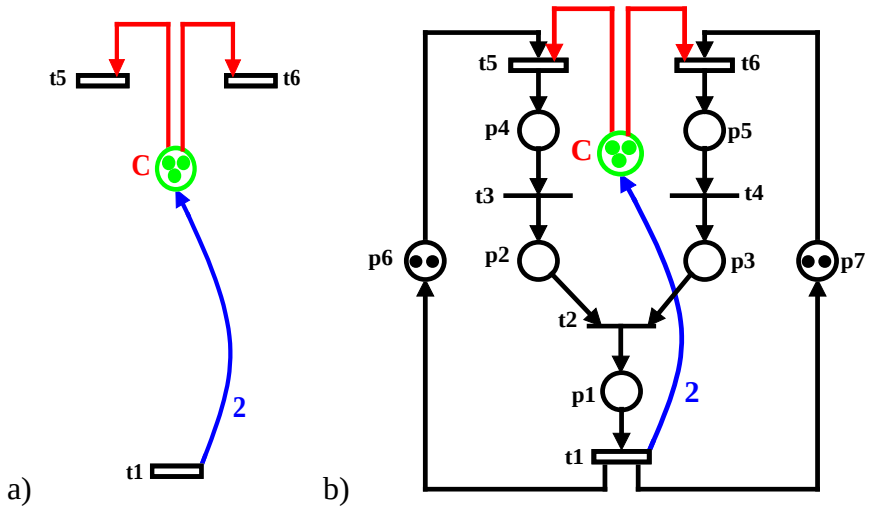
Şekil 5.27 Hesaplanan RW tipi gözetici RWSUPER'in otomata modeli



Şekil 5.28 Geri besleme elemanları C1, C2, C3 ve C4'ün otomata karşılıkları

5.4.2.2 Örnek 2 - optimal çözüm 2

GMEC şartı $m(p_1) \leq 1$'i yerine getirecek optimal çözüm için hesaplanan kontrol mevkisi Şekil 5.29.(a)'da görülmektedir [94]. Bu geri besleme elemanın Şekil 5.23'te görülen UPNM modeline eklenmesi ile elde edilen kontrol edilmiş model Şekil 5.29.(b)'de görülmektedir.

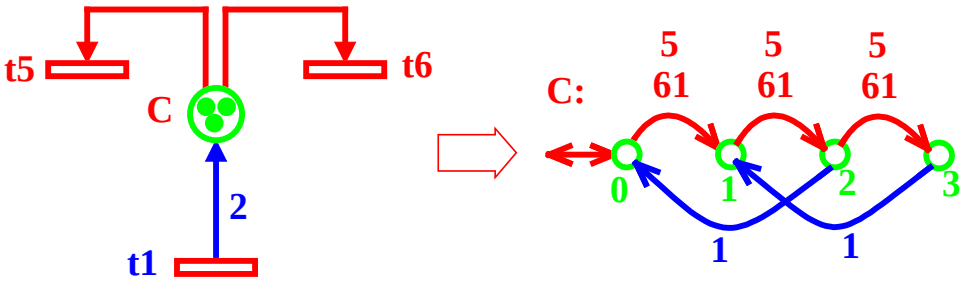


Şekil 5.29 a) [94]'te hesaplanmış olan kontrol mevkisi, b) Kontrol edilmiş model

Yapılacak analiz işlemindeki ilk dört adım önceki örnekte olduğu gibidir. Bu örnek için doğruluk analizi aşağıdaki adımlar takip edilerek yapılmıştır.

5. Adım: Verilen kontrol mevkisi ve bu kontrol mevkisinin otomata karşılığı Şekil 5.30'da görülmektedir. C kontrol mevkisi TCT yazılımında aşağıdaki komut kullanılarak tanımlamıştır.

C = Create(C,[mark 0],[tran[0,5,1],[0,61,1],[1,5,2],[1,61,2],[2,1,0],[2,5,3],[2,61,3],[3,1,1]]) (4,8)



Şekil 5.30 Kontrol mevkisi C ve otomata karşılığı

6. Adım: Kontrol edilmiş model olan PNSUPER2 (PNSUPER2 = PLANT||C) senkron kompozisyonu ile aşağıdaki TCT işlemi ile elde edilmiştir.

PNSUPER2 = Sync(PLANT,C) (32,62) Blocked_events = None

7. Adım: Aşağıdaki test işlemi ile kontrol edilmiş model PNSUPER2'nin RWSUPER ile eş yapılı olduğu gösterilmiştir. Bunun anlamı, PN domeninde elde edilmiş olan denetleyici PNSUPER2'nin mükemmel yani bloklanmasız ve maksimum müsaade edici olduğudur.

true = Isomorph(PNSUPER2,RWSUPER;identity)

5.4.2.3 Örnek 2 - optimal altı çözüm

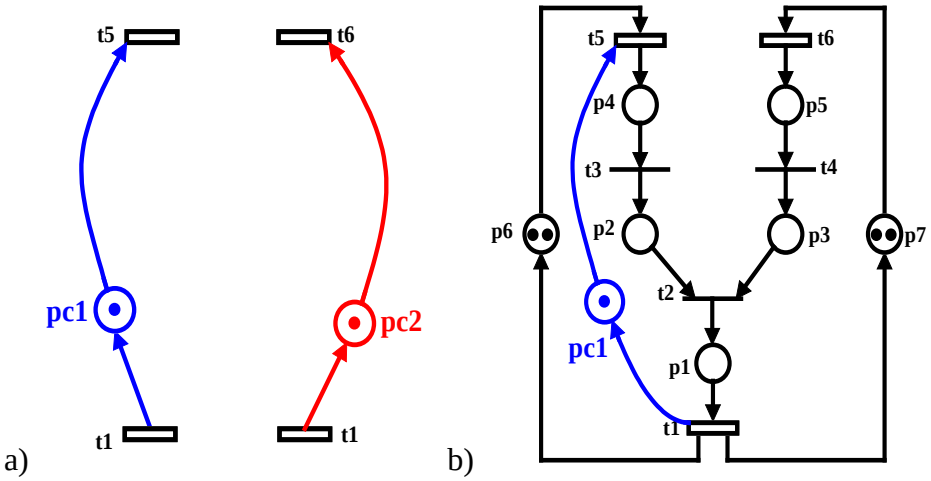
Bu kısımda, Şekil 5.23'te görülen UPNM modelinde GMEC şartı $m(p_1) \leq 1$'i yerine getiren bir optimum altı çözüm incelenmiştir. Basille ve diğerleri tarafından [51]'de iki adet mevki pc1 ve pc2 optimum altı çözüm olarak önerilmiştir. Bu kontrol mevkileri Şekil 5.31.(a)'da görülmektedir. Bu mevkilerden yalnızca birinin kullanımı GMEC

şartı $m(p_1) \leq 1$'i yerine getirmektedir. Bu kısımda, Şekil 5.31.(b)'de görülen ve pc1'i içeren kontrol edilmiş model analiz edilmiştir.

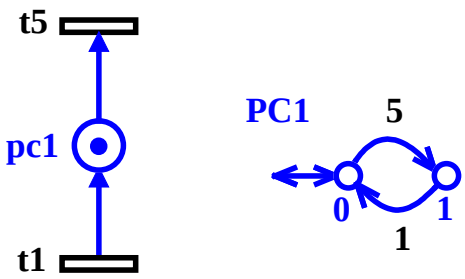
Yapılacak analiz işlemindeki ilk dört adım önceki örnekte olduğu gibidir. Bu örnek için doğruluk analizi aşağıdaki adımlar takip edilerek yapılmıştır.

5. Adım: Verilen kontrol mevkisi ve bu kontrol mevkisinin otomata karşılığı Şekil 5.32'de görülmektedir. PC1 kontrol mevkisi TCT yazılımda aşağıdaki komut kullanılarak tanımlanmıştır.

PC1 = Create(PC1,[mark 0],[tran [0,5,1],[1,1,0]]) (2,2)



Şekil 5.31 a) [51]'de hesaplanan kontrol mevkileri, b) Kontrol edilmiş model



Şekil 5.32 Kontrol mevkisi pc1 ve otomata karşılığı

6. Adım: Kontrol edilmiş model olan SPNSUPER (SPNSUPER = PLANT||PC1) senkron kompozisyon işlemi kullanılarak aşağıdaki TCT işlemi ile elde edilmiştir.

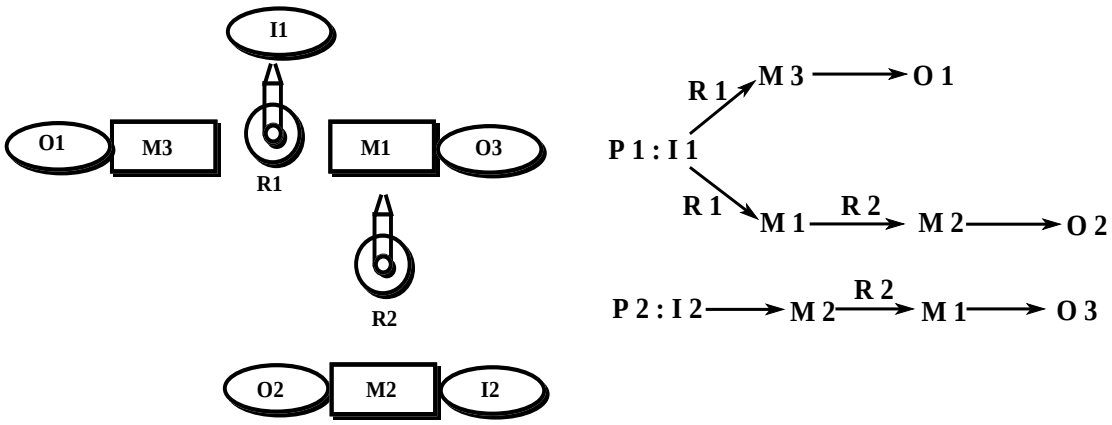
SPNSUPER = Sync(PLANT,PC1) (21,38) Blocked_events = None

7. Adım: Aşağıdaki test işlemleri ile kontrol edilmiş model SPNSUPER'in, optimum çözüm RWSUPER'in alt dili olduğu görülmektedir. Bunun anlamı, PN domeninde elde edilmiş olan denetleyici SPNSUPER'in GMEC şartı $m(p_1) \leq 1$'i yerine getiren optimum altı bir denetleyici olduğudur.

true = Nonconflict(SPNSUPER,RWSUPER)
MRW_PN = Meet(SPNSUPER,RWSUPER) (21,38)
true = Isomorph(MRW_PN,SPNSUPER;identity)

5.4.3 Örnek 3

Bu kısımda, kilitlenme problemi olan bir FMS sistemi örnek olarak incelenmektedir. Bu üretim sistemi Şekil 5.33.(a)'da, bu sistem için tanımlanmış olan üretim sıralamaları ise Şekil 5.33.(b)'de görülmektedir. Sistemde iki robot (R1 ve R2) ve üç makine (M1, M2 ve M3) bulunmaktadır. Robotlar bir anda sadece bir parça taşıyabilmektedir. M3 bir anda sadece bir parça işleyebilmekteyken diğer makineler (M1 ve M2) aynı anda iki parça işleyebilmektedir. Sistemin yüklenmesi için iki adet yükleme tamponu (I1 ve I2) ve üç adet yük boşaltma tamponu (O1, O2 ve O3) mevcuttur. Basitleştirme amacı ile, giriş parçalarının I2'den M2'ye, çıkış parçalarının ise M3'ten O1'e, M2'den O2'ye ve M1'den O3'e aktarılma işlemlerinin makinelerin kendi içerisinde yapıldığı varsayılmıştır. Bu üretim sisteminde, Şekil 5.31.(b)'den de görülebileceği gibi iki parça (P1 ve P2) işlenmektedir. P1 parçası, R1 tarafından I1'den alınarak M1 veya M3'ten birine aktarılır. Eğer P1, M3'e aktarılmışsa, M3 makinesi bu parçayı işleyerek O1'e aktarır. Eğer P1, M1 makinesine aktarılmışsa, M1 makinesinin işini tamamlamasıyla R2 robotu ile M2 makinesine aktarılır. M2 makinesinin de işini tamamlamasıyla P1 parçası M2 makinesi tarafından O2'ye aktarılır. P2 parçası ise I2'den M2'ye alınıp M2 tarafından işlemin bitilmesinden sonra, R2 robotu ile M2'den alınıp M1'e aktarılır. M1'in işleminin bitmesinin ardından P2 parçası M1 tarafından O3'e aktarılır.

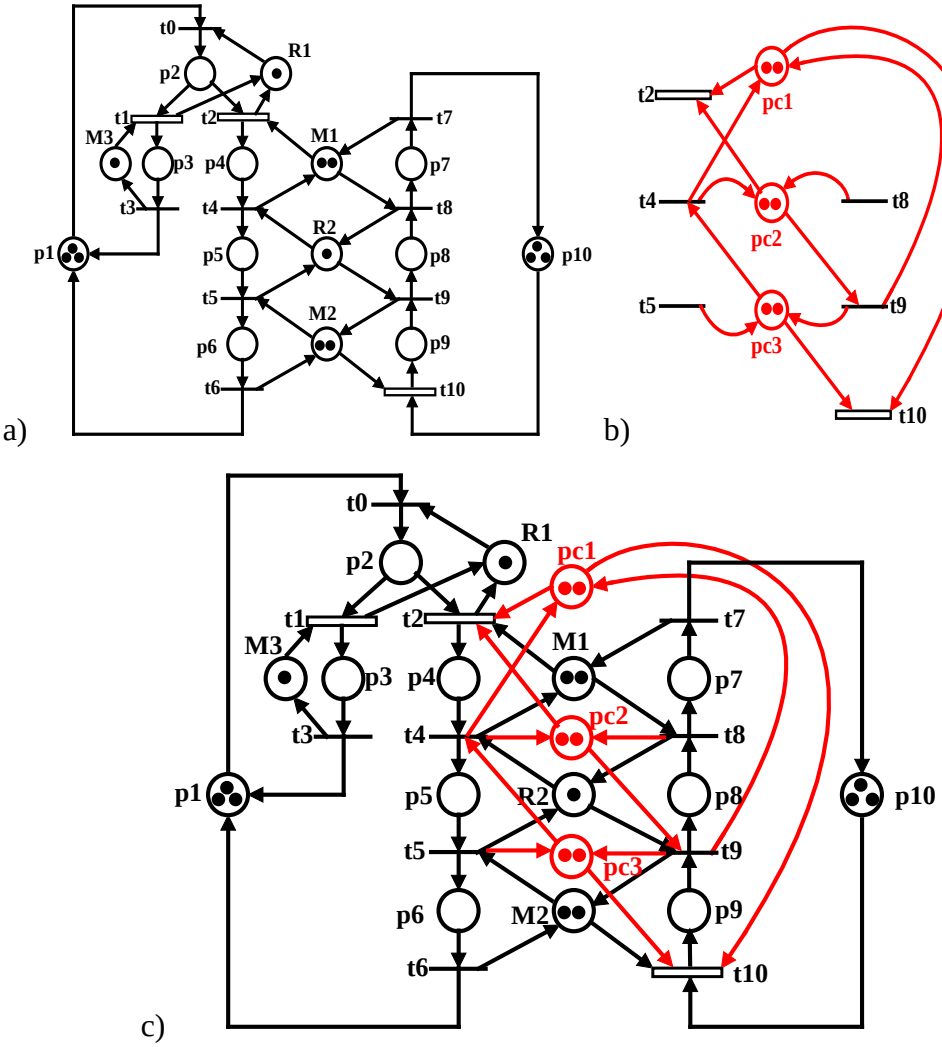


a) b)

Şekil 5.33 a) Örnek bir FMS sistemi, b) Üretim sıralamaları

FMS sistemin PN modeli Şekil 5.34.(a)'da görülmektedir. Başlangıçta sistemde parça olmadığı varsayılmaktadır. Bu modelde 15 mevki P = {p1, p2, …, p10, R1, R2, M1, M2, M3} ve 11 geçiş T = {t0, t1, t2, …, t10} bulunmaktadır. Başlangıçtaki jeton işaretlemesi $M_0 = (3, 0, 0, 0, 0, 0, 0, 0, 0, 3, 1, 1, 2, 2, 1)^T$'dir. Bu modeldeki t1, t2 ve t10 geçişleri kontrol edilebilir geçişler olup içi boş dikdörtgen ile gösterilmiştir. Bu geçişler denetleyici tarafından yasaklanabilir veya yetkilendirilebilir. Modelde bulunan t0, t3, t4, t5, t6, t7, t8 ve t9 geçişleri kontrol edilemeyen geçişlerdir. Bu geçişlere denetleyicinin bir müdahalesi söz konusu değildir. Bu modeldeki P = {p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, R1, R2, M1, M2, M3} mevkilere ait jeton kapasiteleri sırasıyla şu şekildedir: CAP(P) = {3, 1, 1, 2, 1, 2, 2, 1, 2, 3, 1, 1, 2, 2, 1}. p2 ve p3 mevkileri sırası ile R1 ve M3'ün P1 üretim işlemi için çalışmasını ifade etmektedir. p1 mevkisinde bulunan {M(p1) = 3} jeton işaretlemesi P1 üretim işlemi için eşzamanlı olarak yapılabilecek aktivitelerin sayısını belirtmektedir. Benzer şekilde; p2, p4, p5 ve p6 mevkileri sırası ile R1, M1, R2 ve M2'nin P1 üretim işlemi için çalışmasını ifade etmektedir. p9, p8 ve p7 mevkileri sırası ile M2, R2 ve M1'in P2 üretim işlemi için çalışmasını ifade etmektedir. p10 mevkisinde bulunan {M(p10) = 3} jeton işaretlemesi P2 üretim işlemi için eşzamanlı olarak yapılabilecek aktivitelerin sayısını belirtmektedir. R1 ve M3 mevkileri ile aynı isimli paylaşılmayan kaynaklar ifade edilmekteyken, M1, R2 ve M2 mevkileri ile aynı isimli paylaşılan kaynaklar ifade edilmektedir. Bu üretim sisteminde her bir üretim işleminin tamamlanması beklenmektedir. Bu sistemde kilitlenme sorunu mevcuttur. Bu sistem için kontrol spesifikasyonu sistemin maksimum müsaade edici bir şekilde canlı (live) olmasıdır. Canlılık özelliği SCT terminolojisinde bloklanmasız (nonblocking) çalışma olarak

adlandırılmaktadır. Bu FMS sisteminde canlılığı sağlayacak üç adet geri besleme elemanı Ghaffari ve diğerleri tarafından hesaplanmıştır [45]. Hesaplanan kontrol mevkileri pc1, pc2 ve pc3, Şekil 5.34.(b)'de görülmektedir. Bu kontrol mevkilerinin PN modeline eklenmesi ile elde edilen kontrol edilmiş PN modeli Şekil 5.34.(c)'de görülmektedir.



Şekil 5.34 a) FMS sistemine ait PN modeli [45],
b) Hesaplanmış üç kontrol mevkisi, c) Kontrol edilmiş model

Bu örnek için, pc1, pc2 ve pc3 kontrol mevkilerinin doğruluğu aşağıdaki işlemler yapılarak test edilmiştir.

1. Adım: UPNM, Şekil 5.34.(a)'da görülen modeldir. Bu modeldeki geri besleme elemanları ise Şekil 5.34.(b)'de görülen pc1, pc2 ve pc3 kontrol mevkileridir. Spesifikasyon ise canlılığın sağlanmasıdır.

2. Adım: UPNM modelindeki mevkilerin jeton kapasiteleri p1:3, p2:1, p3:1, p4:2, p5:1, p6:2, p7:2, p8:1, p9:2, p10:3, R1:1, R2:1, M1:2, M2:2 ve M3:1 olmak üzere sınırlıdır.

3. Adım: UPNM modelinde bulunan her bir mevki Şekil 5.35'te görüldüğü gibi eşdeğer tampon modellerine dönüştürülmüştür. TCT yazılımındaki ve PN modelindeki olay etiketleri için aşağıda verilen rakam kodlamaları kullanılmıştır.

FMS için TCT olay kodlaması:

FMS:	t0	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10
TCT:	0	1	21	30	4	50	6	70	8	90	101

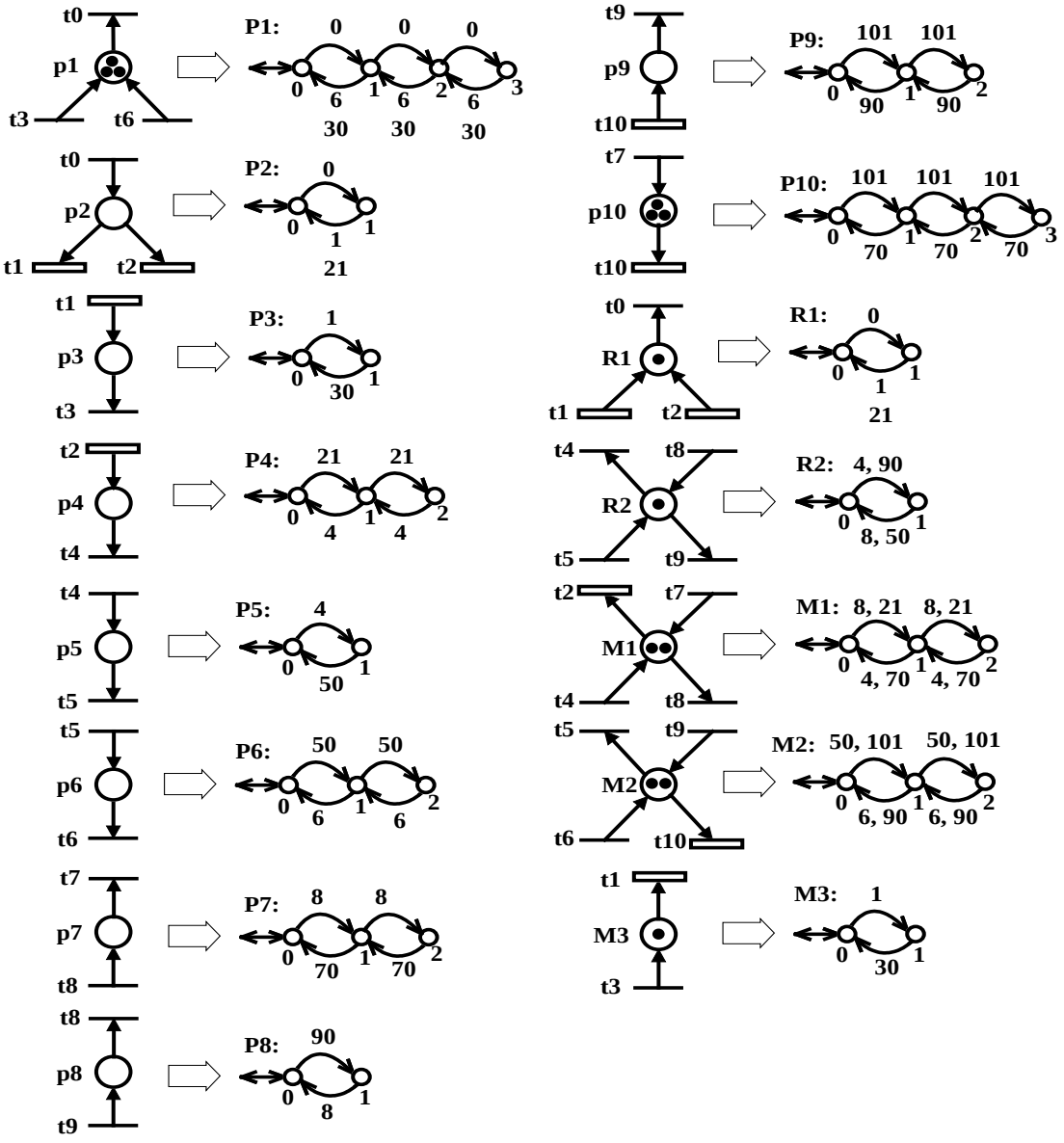
TCT yazılımı kullanılarak *Create(.)* komutuyla Şekil 5.35'te görülen her bir mevkiye ait otomata modeli aşağıdaki gibi oluşturulur:

P1 = Create(P1,[mark0],[tran[0,0,1],[1,0,2],[1,6,0],[1,30,0],[2,0,3], [2,6,1],[2,30,1], [3,6,2], [3,30,2]]) (4,9)
P2 = Create(P2,[mark 0],[tran [0,0,1],[1,1,0],[1,21,0]]) (2,3)
P3 = Create(P3,[mark 0],[tran [0,1,1],[1,30,0]]) (2,2)
P4 = Create(P4,[mark 0],[tran [0,21,1],[1,4,0],[1,21,2],[2,4,1]]) (3,4)
P5 = Create(P5,[mark 0],[tran [0,4,1],[1,50,0]]) (2,2)
P6 = Create(P6,[mark 0],[tran [0,50,1],[1,6,0],[1,50,2],[2,6,1]]) (3,4)
P7 = Create(P7,[mark 0],[tran [0,8,1],[1,8,2],[1,70,0],[2,70,1]]) (3,4)
P8 = Create(P8,[mark 0],[tran [0,90,1],[1,8,0]]) (2,2)
P9 = Create(P9,[mark 0],[tran [0,101,1],[1,90,0],[1,101,2],[2,90,1]]) (3,4)
P10 = Create(P10,[mark 0],[tran [0,101,1],[1,70,0],[1,101,2],[2,70,1],[2,101,3],[3,70,2]]) (4,6)
R1 = Create(R1,[mark 0],[tran [0,0,1],[1,1,0],[1,21,0]]) (2,3)
R2 = Create(R2,[mark 0],[tran [0,4,1],[0,90,1],[1,8,0],[1,50,0]]) (2,4)
M1 = Create(M1,[mark 0],[tran [0,8,1],[0,21,1],[1,4,0],[1,8,2],[1,21,2],[1,70,0],[2,4,1], [2,70,1]]) (3,8)
M2 = Create(M2,[mark 0],[tran [0,50,1],[0,101,1],[1,6,0],[1,50,2],[1,90,0],[1,101,2],[2,6,1], [2,90,1]]) (3,8)
M3 = Create(M3,[mark 0],[tran [0,1,1],[1,30,0]]) (2,2)

Bu işlemlerin ardından FMS tüm mevkilerin senkron çarpımı (FMS = p1||p2||p3||p4||p5||p6||p7||p8||p9||p10||R1||R2||M1||M2||M3) alınarak aşağıda görüldüğü gibi oluşturulur.

FMS = Sync(P1,P2) (8,23) Blocked_events = None
FMS = Sync(FMS,P3) (16,36) Blocked_events = None
FMS = Sync(FMS,P4) (48,132) Blocked_events = None
FMS = Sync(FMS,P5) (96,280) Blocked_events = None
FMS = Sync(FMS,P6) (36,88) Blocked_events = None

FMS = Sync(FMS,P7) (108,408) Blocked_events = None
FMS = Sync(FMS,P8) (216,852) Blocked_events = None
FMS = Sync(FMS,P9) (648,2880) Blocked_events = None
FMS = Sync(FMS,P10) (504,2168) Blocked_events = None
FMS = Sync(FMS,R1) (504,2168) Blocked_events = None
FMS = Sync(FMS,R2) (426,1696) Blocked_events = None
FMS = Sync(FMS,M1) (341,1303) Blocked_events = None
FMS = Sync(FMS,M2) (261,933) Blocked_events = None
FMS = Sync(FMS,M3) (261,933) Blocked_events = None



Şekil 5.35 Şekil 5.34'te görülen PN modelindeki mevkiler ve otomata dönüşümleri

İncelenen problem canlılığı sağlama problemi olduğundan, spesifikasyon başlangıç durumuna göre bloklanmasızlığın sağlanmasıdır. Bu spesifikasyon, *Allevents(.)* komutu ile elde edilir:

ALL = Allevents(FMS) (1,11).

Ayrıca aşağıdaki test, kontrolün olmadığı durumda FMS'in bloklanmaya sahip olduğunu göstermektedir:

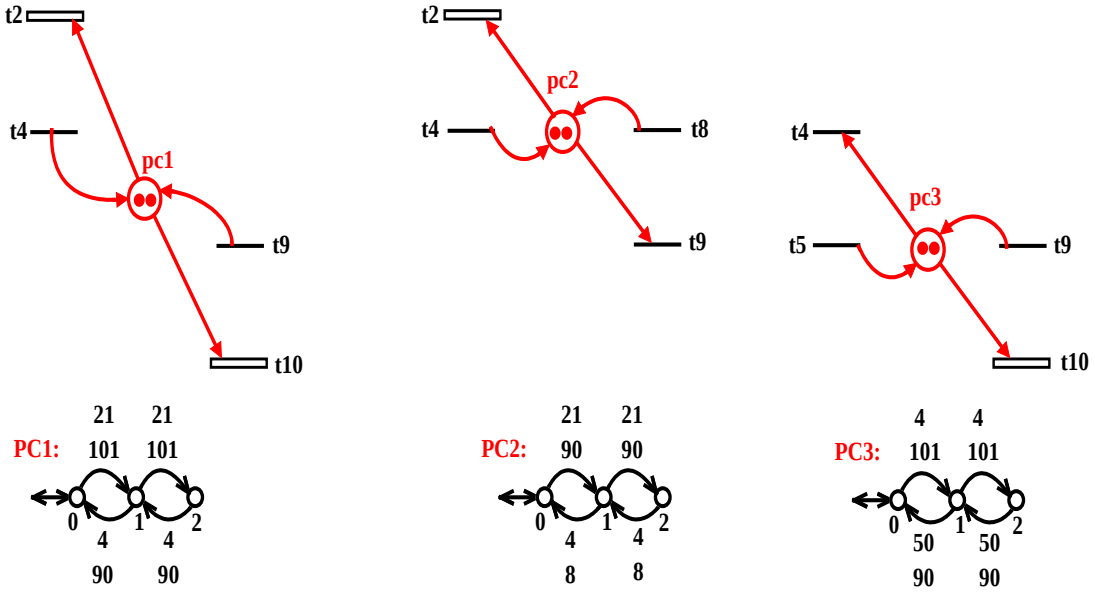
false = Nonconflict(FMS,ALL).

4. Adım: RW gözeticisi aşağıdaki gibi elde edilir:

RWSUPER = Supcon(FMS,ALL) (215,771)

5. Adım: Verilen geri besleme elemanları pc1, pc2 ve pc3, Şekil 5.36'da görüldüğü gibi otomata modellerine dönüştürülür. Bu otomata modelleri TCT yazılımında aşağıdaki gibi tanımlanır.

PC1 = Create(PC1,[mark 0],[tran [0,21,1], [0,101,1], [1,4,0], [1,21,2], [1,90,0], [1,101,2], [2,4,1], [2,90,1]]) (3,8)
PC2 = Create(PC2,[mark 0],[tran [0,21,1],[0,90,1],[1,4,0],[1,8,0],[1,21,2],[1,90,2],[2,4,1], [2,8,1]]) (3,8)
PC3 = Create(PC3,[mark 0],[tran [0,4,1], [0,101,1], [1,4,2], [1,50,0], [1,90,0], [1,101,2], [2,50,1], [2,90,1]]) (3,8)



Şekil 5.36 pc1, pc2 ve pc3 kontrol mevkileri ve otomata dönüşümleri

6. Adım: Kontrol edilmiş model olan PNSUPER, PNSUPER=FMS||PC1||PC2||PC3 senkron kompozisyonu ile aşağıdaki TCT işlemleri ile elde edilir:

PNSUPER = Sync(FMS,PC1) (227,800) Blocked_events = None
PNSUPER = Sync(PNSUPER,PC2) (223,793) Blocked_events = None
PNSUPER = Sync(PNSUPER,PC3) (215,771) Blocked_events = None

7. Adım: Aşağıdaki test işlemi ile kontrol edilmiş model PNSUPER'in RWSUPER ile eş yapılı olduğu gösterilmiştir. Bunun anlamı, PN domeninde elde edilmiş olan denetleyicinin mükemmel yani bloklanmasız ve maksimum müsaade edici olduğudur.

true = Isomorph(PNSUPER,RWSUPER;identity)

Aşağıdaki test sonucunda optimal kontrol edilmiş davranış PNSUPER (veya RWSUPER)'in trim edilmiş plant'ın ürettiği dil olmadığı görülmektedir. Bu durumda, optimum kontrol edilmiş davranış PNSUPER (veya RWSUPER) maksimum müsaade edici ve bloklanmasız davranışı ifade eden en iyi şekilde kontrol edilebilir bir alt dil (supremal controllable sublanguage) dir.

TRIMFMS = Trim(FMS) (232,838)
false = Isomorph(PNSUPER,TRIMFMS)

5.5 Sonuç

Bu kısımda yapılan çalışmalarda, PN temelli denetleyicilerin doğruluk analizini yapmak için TCT yazılımının kullanıldığı yeni bir yöntem önerilmiştir. Kontrol edilmemiş PN modeli, spesifikasyonlar ve PN temelli denetleyicinin verildiği durumda, önerilen yöntem ile denetleyicinin spesifikasyonları PN modeli üzerinde optimum olarak sağlayıp sağlamadığı test edilebilir. Önerilen yöntemin kullanımı üç farklı örnek ile gösterilmiştir.

BÖLÜM VI

AYRIK OLAY SİSTEMLERİ İÇİN MODÜLER MELEZ DENETLEYİCİLERİN HESAPLANMASI VE PLC İLE GERÇEKLEŞTİRİLMESİ

AOS'lerin kontrolü için [52]'de önerilen melez yöntemin gerçek endüstriyel sistemlerin kontrolüne uygulanması uzaktan kumandalı otomatik bir kapı modeli [67] ve deneysel bir endüstriyel üretim sistemi [67] kullanılarak gösterilmiştir. Bu çalışmaların her ikisinde de elde edilen gözeticiler yekpare biçimdedir. Modellenen sistemin karmaşıklaşması ile sistem modeli büyümekte ve hesaplamalar sonucunda elde edilen gözeticinin durum sayısında üstel bir artış meydana gelmektedir. Bu artışa durum patlaması problemi denilmektedir. Bu artış büyük ölçekli sistemler için hem yapılan hesaplamaları hem de gözetici yapısının karmaşıklaşmasını beraberinde getirmektedir. Bu artışın önlenebilmesi ve daha sade gözetici yapısının elde edilmesi için literatürde çeşitli yöntemler önerilmiştir. Bunlardan ilki gözetici yapısının küçültülmesidir. Gözetici yapısının sadeleştirilmesi için Su ve Wonham tarafından bir yöntem önerilmiştir [99]. Bu yöntem AOS'ler için gözetici hesaplaması yapılan TCT isimli yazılıma *Supreduce(.)* komutu olarak eklenmiştir. Bu komut kullanılarak gözetici yapısı her ne kadar sadeleştirilebilse de, sadeleşmiş olan gözeticinin gerçek sistemlerin kontrolünde kullanılmasında çığ etkisi gibi çeşitli problemlerle karşılaşılmaktadır [67]. Bu problem özellikle elde edilen gözeticinin fiziksel gerçekleştirme aşamasında ortaya çıkmaktadır.

Durum patlamasının sebep olduğu hesaplama zorluğu ve karmaşık gözetici elde edilmesi probleminin çözümü için hiyerarşik (hierarchical), dağıtık (decentralized) ve modüler yaklaşımlar da önerilmiştir. Bunlardan hiyerarşik yaklaşım orijinal sisteme göre daha az karmaşık olan modelleri (düşük seviyeli model-low level model) içeren sisteme ait ana modelin (high level model-yüksek seviyeli model) elde edilmesi prensibine dayanmaktadır [100-104].

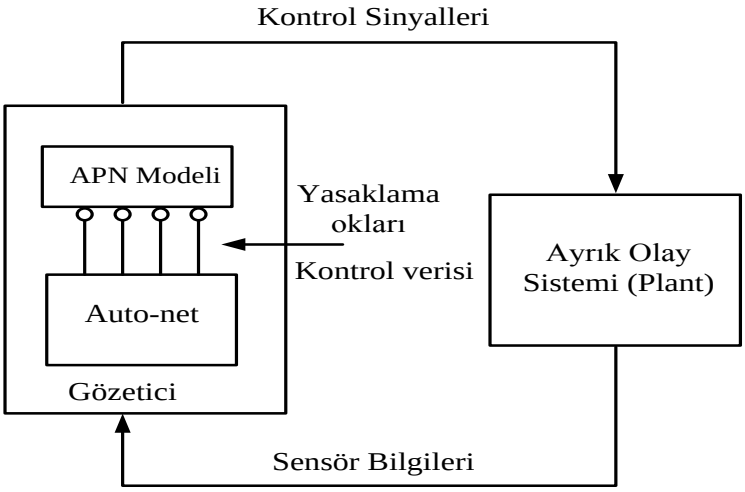
Durum patlaması probleminin çözümüne yönelik yaygın kullanılan diğer yaklaşımlar ise dağıtılmış (decentralized) ve modüler kontrol yaklaşımlarıdır [9, 82, 105,106]. Bu yöntemde, sistemin kontrol edilmiş davranışını belirleyecek her bir spesifikasyon için ana sistemin ilgili kısmı göz önünde bulundurularak lokal bir gözetici elde edilir. Bu

yöntemde karşılaşılabilecek en önemli problem, elde edilen lokal gözeticilerin birbirleri ile veya ana sistem ile ikilem oluşturmasıdır. Eğer ikilem mevcut ise bu ikilemin giderilmesi gerekmektedir. Bunun için literatürde koordinatör adı verilen ve elde edilen lokal gözeticileri organize eden ekstra bir gözeticinin hesabına yönelik çalışmalar mevcuttur [100].

AOS'lerin denetlenmesinde yekpare gözetici kullanımına alternatif olarak ortaya konan modüler gözetici kullanımını öngören metotların en önemli avantajı durum patlaması probleminin giderilmesi ve aynı zamanda elde edilen modüler gözeticilerin yekpare gözeticilere kıyasla kolaylıkla gerçekleştirilmesidir. Bu kapsamda [52]'de ortaya konan melez yöntem kullanılarak yüksek seviyeli bir imalat sisteminin modüler olarak denetlenmesinin mümkün olduğu gösterilmiş bulunmaktadır. Bu kısımda, [52]'de ortaya konan AOS'lerin modüler melez kontrolü ile ilgili yöntemin deneysel endüstriyel üretim sistemi kullanılarak uygulanabilirliği incelenmektedir.

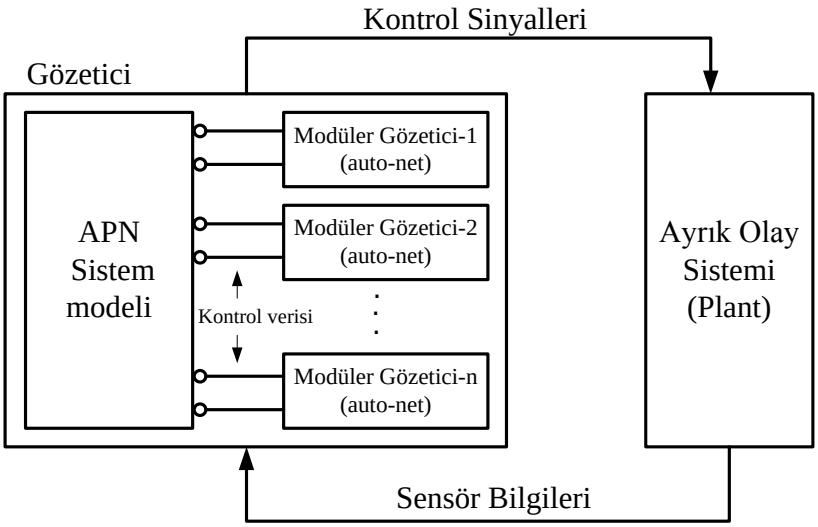
6.1 Ayrık Olay Sistemlerinin Modüler Gözeticili Melez Kontrolü

AOS'lerin denetlenmesinde gözeticinin rolü, sistemdeki sensörlerden gelen geri besleme bilgisini kullanarak, sistemdeki kontrol edilebilir olayları yasaklamak suretiyle sistemi denetlemektedir. Bu kapsamda daha önceki çalışmalarda önerilen yekpare gözetici yapısı Şekil 6.1'de görülmektedir. Gözeticinin yekpare olması durumunda sistemden sensör okumaları ile alınan bilgiler tek bir gözetici tarafından değerlendirilerek kontrol verisi oluşturulmaktadır. Yekpare gözeticinin tasarımı için ilk olarak kontrol edilecek sisteme (Plant) ait kontrol edilmemiş APN modelinin oluşturulması gerekir. Bu işlemin ardından; Petri ağı modeli, tampon modelleri kullanılarak otomata modellerine çevrilmeli ve spesifikasyonlar otomata olarak modellenmelidir. Bu işlemlerin ardından TCT yazılımı kullanılarak gözetici tasarlanmaktadır. Otomata olarak elde edilen gözeticinin auto-net olarak karşılığı bulunarak TCT yardımıyla elde edilen kontrol verisi auto-netin durumlarından Petri ağı modelinin geçişlerine bağlanan yasaklama okları ile gerçekleştirilir.



Şekil 6.1 AOS'lerin yekpare (monolithic) melez kontrolü

Gözeticilerin modüler olduğu durumda AOS'lerin kontrolünde Şekil 6.2'de görülen yapı söz konusu olmaktadır. Bu yapıdan görüleceği gibi kontrol edilecek sistemin (plant) kontrol edilmemiş davranışı APN ile modellenmektedir.



Şekil 6.2 AOS'lerin modüler melez kontrolü

Sistemi kontrol etmek için belirlenen spesifikasyonların her biri için sistemin ilgili parçası göz önünde bulundurularak TCT yazılımı ile modüler bir gözetici elde edilir. Otomata olarak elde edilen her bir modüler gözleticinin auto-net olarak karşılığı bulunarak TCT yardımıyla hesaplanan kontrol verisi auto-netin durumlarından Petri ağı modelinin geçişlerine bağlanan yasaklama okları ile gerçekleştirilir. Modüler gözetici tasarımı için takip edilecek işlem basamakları aşağıdaki gibi önerilmektedir.

1. Sistemin kontrol edilmemiş davranışı APN olarak modellenir.
2. APN modelinin sınırlı (bounded) olduğu ve mevkilerinin jeton kapasiteleri belirlenmelidir.
3. Eğer gerekliyse Petri ağı sadeleştirme kuralları ile APN sadeleştirilmelidir.
4. APN modelindeki mevkiler [90]'da önerilen yöntem kullanılarak eşdeğer tampon (buffer) otomata modellerine dönüştürülür.
5. Spesifikasyonlar otomata olarak modellenir.
6. Her bir spesifikasyon (SPECm) için sistemin ilgili parçası (PLANTm) belirlenerek modüler bir gözetici (MODSUPm) ve bir kontrol verisi (CDATm) elde edilir.
7. Elde edilen modüler gözeticilerin birbirileri ile ve sistem ile ikilem veya bloklanma oluşturup oluşturmadığı TCT yazılımı ile test edilir.
8. Elde edilen modüler gözeticiler auto-net yapılarına dönüştürülür.
9. Auto-net yapısındaki modüler gözeticiler (MODSUPm), kontrol verileri (CDATm) auto-netlerin mevkilerinden APN'in geçişlerine yasaklama okları ile bağlanarak modüler kapalı çevrim melez kontrol sistemi elde edilir.
10. Elde edilen modüler kapalı çevrim melez kontrol sistemi bir PLC ile gerçekleştirilir.

İkinci adımdaki sınırlılık hesaplaması standart PN yazılımları kullanılarak ya da basit durumlarda gözlemle yapılır. Üçüncü adımdaki, sadeleştirme kuralları literatürde mevcut olup, PN modelinde canlılık, sınırlılık ve tersinirlik gibi özellikleri değiştirmemektedir.

Dördüncü adımda gerekli olan dönüşüm işlemleri beşinci bölümde ve [90]'da sunulan yaklaşım kullanılarak yapılabilir. Bu yaklaşıma göre, eğer bir mevkinin jeton kapasitesi b ise, bu mevki, kapasitesi $b+1$ olan bir tampon modeline dönüştürülmektedir.

Beşinci ve altıncı adımdaki hesaplamalar TCT yazılımı [68] kullanılarak yapılabilir. Tezin ikinci bölümünde TCT yazılımının kullanımı detaylı olarak açıklanmıştır. APN modelinde bulunan mevkiler (p1, p2, p3, ..., pN) bu mevkilerin tampon karşılıklarının otomata modelleri P1, P2, P3, ..., PN olmak üzere beşinci ve altıncı adımda yapılacak işlemler aşağıdaki gibi özetlenebilir:

5.1 AOS'lerin Gözetimli Kontrol Teorisi yardımıyla modüler olarak kontrolünde yapılacak olan ilk işlem sistemi oluşturan tüm alt bileşenlerin (P1, P2, …, PN) otomata olarak oluşturulması ve bu otomata modellerinin TCT yazılımında *Create(.)* komutu kullanılarak gerçekleştirilmesi işlemidir:

P1=Create(P1)
P2=Create(P2)
.
.
PN=Create(PN)

5.2 Gözetici tasarımında yapılacak diğer işlemde, sistemin çalışmasında istenilen özelliklerin her biri yani spesifikasyonlar *Create(.)* komutuyla ayrı ayrı elde edilir (SPEC1, SPEC2, ..., SPECM Burada; m spesifikasyonların sayısıdır):

SPEC1=Create(SPEC1)
SPEC2=Create(SPEC2)
.
.
SPECM=Create(SPECm)

5.3 Dikkate alınacak her bir spesifikasyon için sistemin ilgili kısmı göz önünde bulundurularak PLANTM'ler, {x, y, z, w=1, 2, ..., n olmak üzere (burada n APN modelinin spesifikasyonu ilgilendiren kısımdaki mevkilerin sayısıdır)} aşağıdaki gibi oluşturulur:

PLANT1=Px||Py||Pz...Pw
PLANT2=Px||Py||Pz... Pw
.
.
PLANTM=Px||Py||Pz... Pw

5.4 Daha sonra elde edilen her bir spesifikasyon (SPECM) ve ilgili sistem parçası modeli (PLANTM) için ayrı bir gözetici (MODSUPM) aşağıdaki gibi elde edilir:

MODSUP1=Supcon(PLANT1, SPEC1)
MODSUP2=Supcon(PLANT2, SPEC2)
.
.
MODSUPM=Supcon(PLANTM, SPECM)

5.5 Bir önceki adımda elde edilen MODSUPM ve PLANTM'ler kullanılarak kontrol verileri CDATM'ler aşağıdaki gibi hesaplanır:

CDAT1=Condat(PLANT1, MODSUP1)
CDAT2=Condat(PLANT2, MODSUP2)

.
.
CDATM=Condat(PLANTM, MODSUPM)

Yedinci adımda, modüler gözeticilerin herhangi bir ikileme veya bloklanmaya sebep olup olmadığı TCT yardımıyla aşağıdaki gibi test edilir.

Öncelikle sistem için yekpare (monolithic) gözeticinin hesaplanması gerekmektedir. Bu hesaplama aşağıda görülen TCT işlemleri kullanılarak gerçekleştirilir.

PLANT = Sync(P1,P2)
PLANT = Sync(PLANT,P3)
.
.
PLANT = Sync(PLANT,PN)

SPEC = Sync(SP1,SP2)
SPEC = Sync(SPEC,SP3)
.
.
SPEC = Sync(SPEC,SPM)

MONOSUPER = Supcon(PLANT,SPEC)

TEST = Sync(MODSUP1,MODSUP2)
TEST = Sync(TEST,MODSUP3)
.
.
TEST = Sync(TEST,MODSUPM)
TEST = Sync(TEST,PLANT)

TEST ve MONOSUPER otomatalarının eş yapılı olması aşağıdaki TCT komutuyla test edilebilir. Eğer komut çıktısı *"true"* ise iki otomata eş yapılı, *"false"* ise eş yapılı değildir.

Isomorph(MONOSUPER,TEST)

Elde edilen gözeticilerin birbiri ile ikilem oluşturup oluşturmadığı TCT yazılımının *Nonconflict(.)* komutu ile test edilmektedir. Örneğin iki adet modüler gözetici olduğu durum için test aşağıdaki TCT komutu ile yapılabilir. Eğer komut çıktısı "true" ise iki otomata birbiri ile ikilemsizdir. Eğer çıktı "false" ise ikilem söz konusudur.

Nonconflict(MODSUP1,MODSUP2)

Sekizinci adımda, bir önceki adımda elde edilen otomata formundaki modüler gözeticiler, PN'lere benzer bir yapı olan auto-net formuna dönüştürülür. Bu işlemde, gözetici modelinde bulunan özdöngüler, durumlarda değişikliğe sebep olmadığından ihmal edilir.

Dokuzuncu adımda, modüler gözeticilere ait auto-net modelleri PLANT'a kontrol verisinde hesaplanan yasaklama okları ile bağlanarak kapalı çevrim melez model elde edilir.

Son adımda, modüler gözeticili kapalı çevrim melez kontrol sistemi PLC ile gerçekleştirilir.

6.2 Deneysel Endüstriyel Üretim Sistemi İçin Modüler Gözeticili Melez Denetleyicilerin Hesaplanması ve PLC ile Gerçekleştirilmesi

Bu kısmında, deneysel endüstriyel üretim sistemi için modüler gözeticili melez denetleyicilerin hesaplanması ve PLC ile gerçekleştirilmeleri anlatılmaktadır. Deneysel endüstriyel üretim sisteminin detayları ve PLC ile bağlantıları tezin ikinci bölümünde açıklanmıştır. Deneysel endüstriyel üretim sisteminde bulunan halka toplama kolunda en fazla beş halka bulunabilmektedir. Halka toplama kolu kapasitesinin değişik durumları için yapılan üç farklı uygulama aşağıda açıklanmıştır.

6.2.1 Halka toplama kolu kapasitesinin 1 olduğu durum

Halka toplama kolu kapasitesinin 1 olduğu durum için modüler gözeticili melez denetleyici aşağıdaki gibi hesaplanmıştır.

1. Adım: Deneysel endüstriyel üretim sisteminin kontrol edilmemiş APN modeli Şekil 6.3'te görülmektedir. Modelin detayları tezin üçüncü bölümünde açıklanmıştır.

2. Adım: Şekil 6.3'te görülen APN modelinin 1 sınırlı olduğu ve mevkilerinde en fazla bir jeton bulanabileceği kolaylıkla görülmektedir.

3. Adım: Şekil 6.3'te görülen APN modeli basit bir yapıya sahip olduğu için bir sadeleştirme tekniği uygulanmasına gerek görülmemiştir.

4. Adım: Şekil 6.3'te görülen sistemin kontrol edilmemiş APN modelinin otomataya çevrilmesi gerekmektedir. Bu modeldeki her bir mevkinin aşağıda verilen TCT olay kodlaması göz önünde bulundurularak elde edilen tampon otomata modelleri Şekil 6.4'te görülmektedir. TCT olay kodlaması yapılırken, TCT yazılımının gereksinimi olarak kontrol edilebilir olaylar için tek sayılar, kontrol edilemeyen olaylar için ise çift sayılar etiket olarak atanmıştır.

TCT olay etiketleri

APN geçişi :	t1	t2	t3	t4	t5	t6	t7
TCT etiketi :	12	22	32	42	52	61	71

5. Adım: Deneysel endüstriyel üretim sisteminin istenilen çalışma özellikleri (spesifikasyonlar) aşağıda sıralanmıştır:

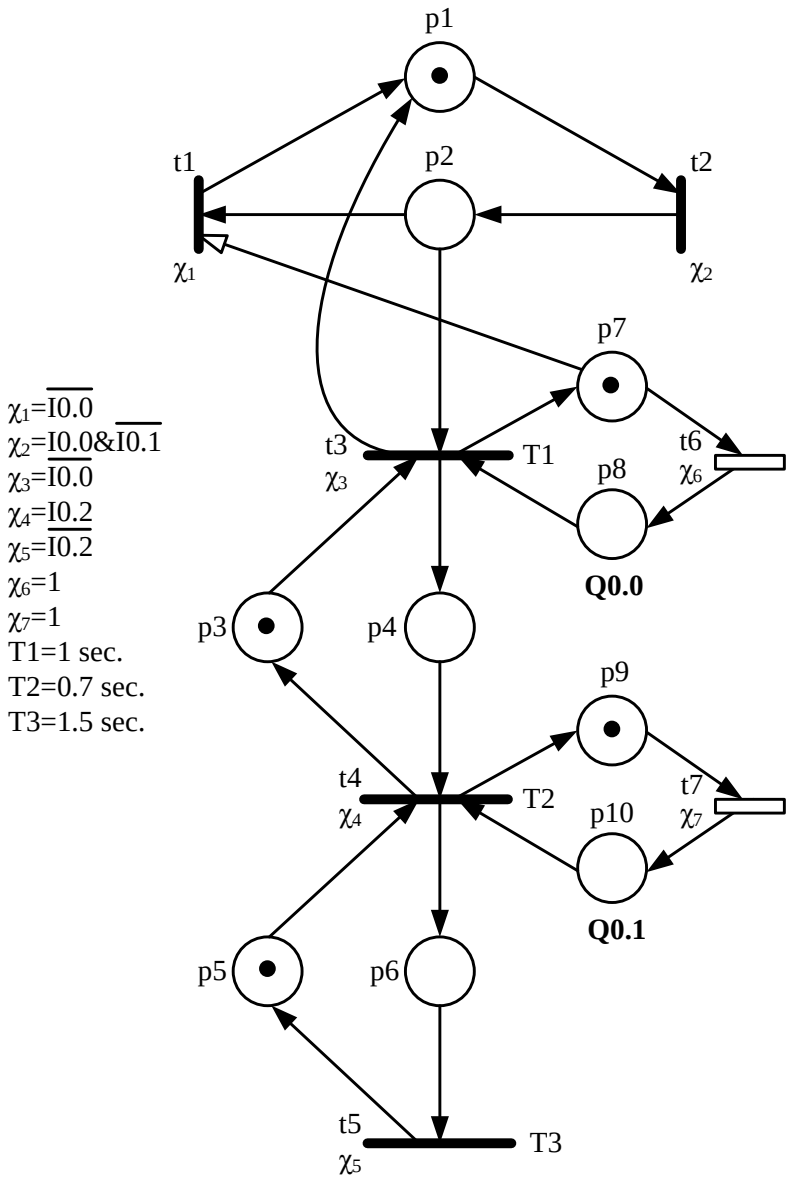
1. Seçme selenoidi (itici selenoid) seçme alanında bir halka bulunduğunda ve halka toplama kolu boş olduğunda çalışacaktır (SP1).
2. Birleştirme selenoidi (döner selenoid) halka toplama kolunda bir halka bulunduğunda ve birleştirme noktası boş olduğunda çalışacaktır (SP2).

Gözetici tasarımı için bu spesifikasyonların otomata olarak modellenmeleri gerekmektedir. Birinci spesifikasyonu iki alt spesifikasyonun birleşimi şeklinde düşünmek mümkündür. Şöyle ki: "Seçme selenoidini seçme alanında bir halka olduğunda çalıştır" VE "Seçme selenoidini halka toplama kolu boş olduğunda çalıştır". Bu kısımlar, sırası ile SP1A ve SP1B olarak isimlendirilen otomatalarla modellenir. Bu iki otomatanın senkron kompozisyon işlemiyle birleştirilmesi ile birinci spesifikasyona ait otomata modeli elde edilir. Birinci spesifikasyona ait otomata modelleri Şekil 6.5'te görülmektedir. Birinci spesifikasyonu elde etmek için gerçekleştirilen TCT işlemleri ise aşağıda görülmektedir.

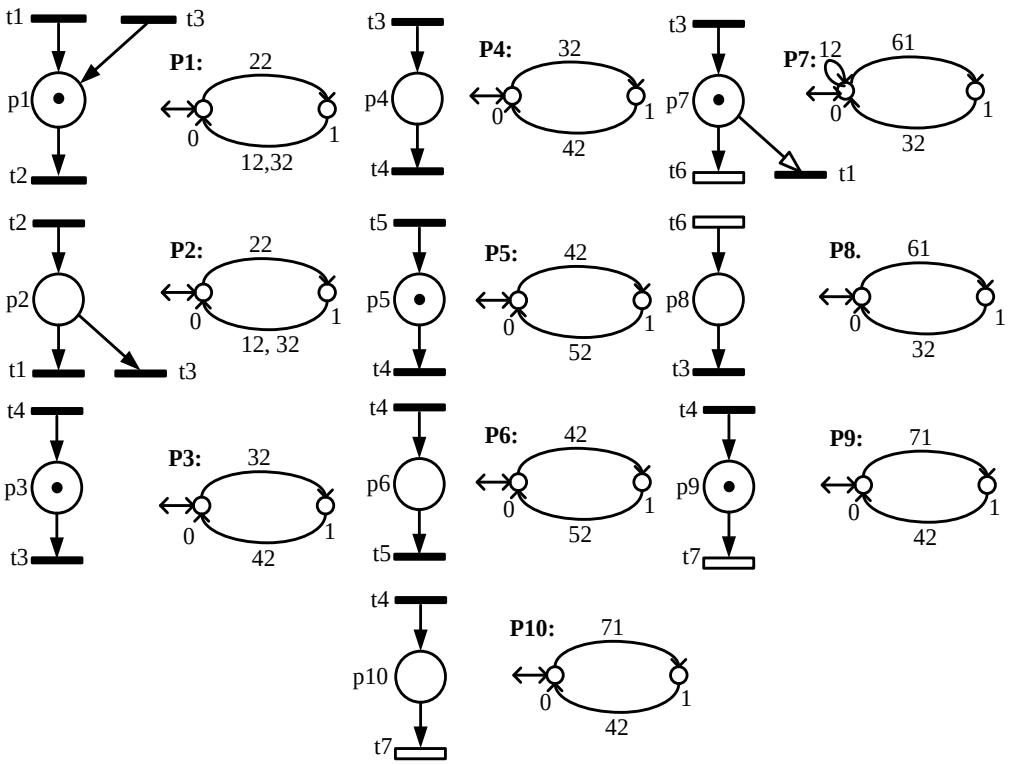
```
SP1A = Create(SP1A,[mark 0],[tran [0,22,1],[1,12,0],[1,32,0],[1,61,1]])  (2,4)
SP1B = Create(SP1B,[mark 0],[tran [0,32,1],[0,61,0],[1,42,0]])  (2,3)
SP1 = Sync(SP1A,SP1B)  (4,8)  Blocked_events = None
```

İkinci spesifikasyon da benzer şekilde iki alt spesifikasyonun birleşimi şeklinde düşünülebilir. Şöyle ki: "Birleştirme selenoidini halka toplama kolunda bir halka olduğunda çalıştır" VE "Birleştirme selenoidini birleştirme noktası boş olduğunda çalıştır". Birinci spesifikasyonda olduğu gibi bu iki parçaya ait otomata modelleri birleştirilir. İkinci spesifikasyona ait otomata modelleri Şekil 6.6'da görülmektedir. Bu spesifikasyonu elde etmek için gerçekleştirilen TCT işlemleri ise aşağıda verilmiştir.

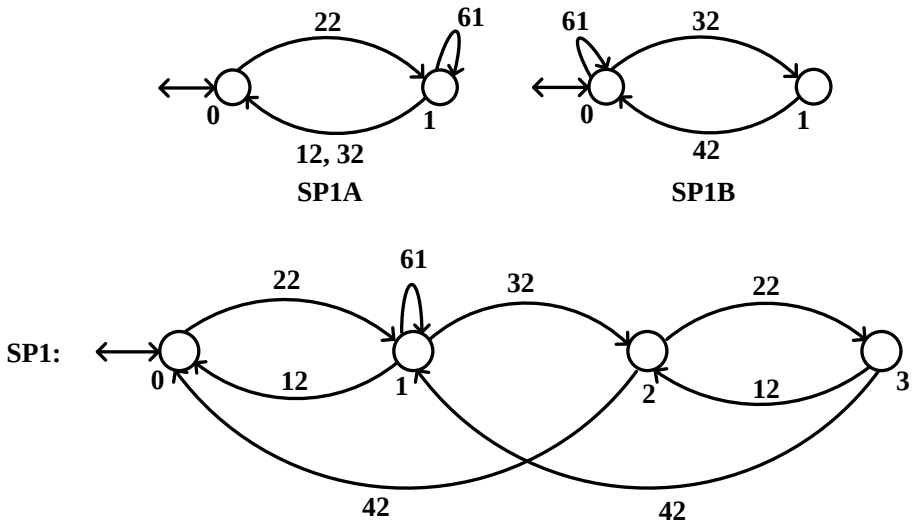
SP2A = Create(SP2A,[mark 0],[tran [0,32,1],[1,42,0],[1,71,1]]) (2,3)
SP2B = Create(SP2B,[mark 0],[tran [0,42,1],[0,71,0],[1,52,0]]) (2,3)
SP2 = Sync(SP2A,SP2B) (4,6) Blocked_events = None



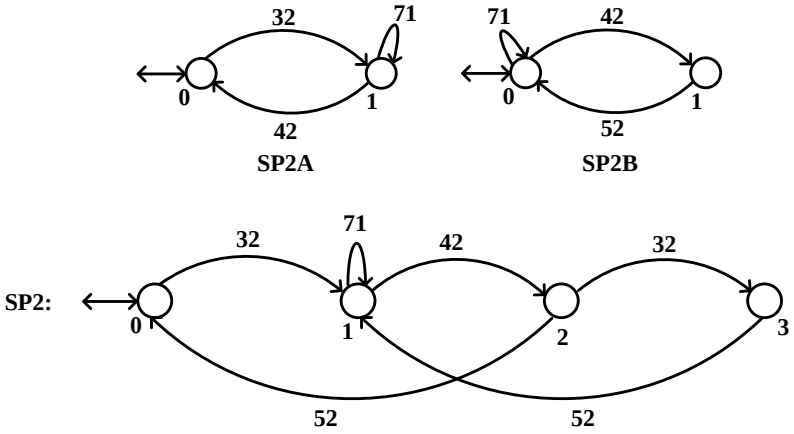
Şekil 6.3 Deneysel endüstriyel üretim sisteminin halka toplama kolu kapasitesinin 1 olduğu durum için kontrol edilmemiş APN modeli



Şekil 6.4 Şekil 6.3'te görülen APN modelindeki mevkilerin tampon (buffer) karşılıkları



Şekil 6.5 Birinci spesifikasyona ait otomata modelleri



Şekil 6.6 İkinci spesifikasyona ait otomata modelleri

6. Adım: Modüler gözeticileri elde etmek için yapılması gereken TCT işlemi Şekil 6.4'te görülen mevkilerin tampon otomata karşılıklarının oluşturulmasıdır. Bu işlemler aşağıda görülmektedir.

P1 = Create(P1,[mark 0],[tran [0,22,1],[1,12,0],[1,32,0]]) (2,3)
P2 = Create(P2,[mark 0],[tran [0,22,1],[1,12,0],[1,32,0]]) (2,3)
P3 = Create(P3,[mark 0],[tran [0,32,1],[1,42,0]]) (2,2)
P4 = Create(P4,[mark 0],[tran [0,32,1],[1,42,0]]) (2,2)
P5 = Create(P5,[mark 0],[tran [0,42,1],[1,52,0]]) (2,2)
P6 = Create(P6,[mark 0],[tran [0,42,1],[1,52,0]]) (2,2)
P7 = Create(P7,[mark 0],[tran [0,12,0],[0,61,1],[1,32,0]]) (2,3)
P8 = Create(P8,[mark 0],[tran [0,61,1],[1,32,0]]) (2,2)
P9 = Create(P9,[mark 0],[tran [0,71,1],[1,42,0]]) (2,2)
P10 = Create(P10,[mark 0],[tran [0,71,1],[1,42,0]]) (2,2)

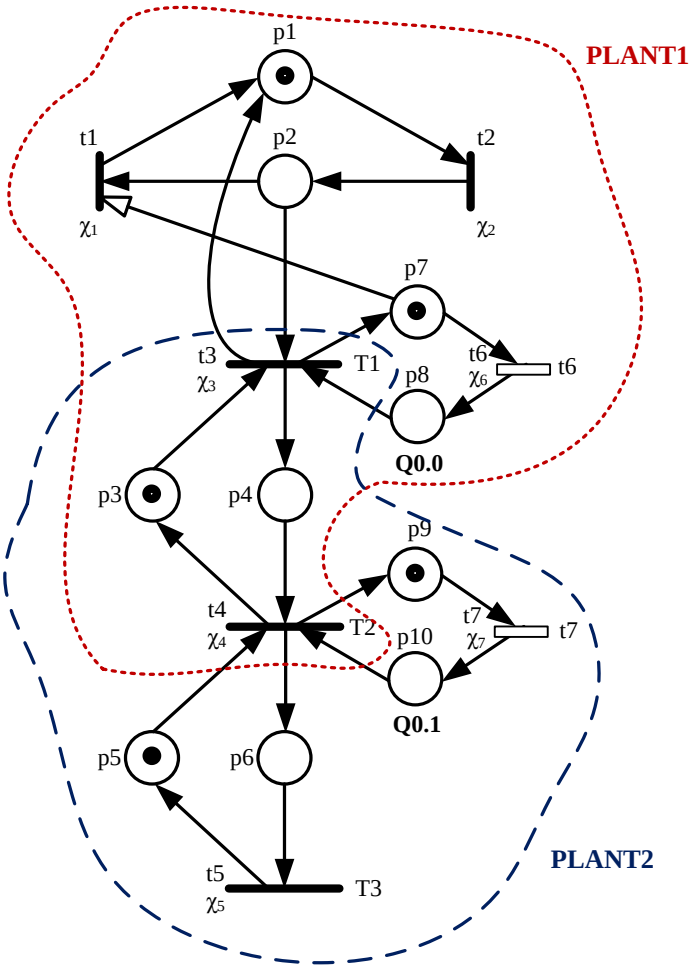
Bu adımdan sonra spesifikasyonlar göz önünde bulundurularak, her bir spesifikasyonla ilgili sistem parçası elde edilir ve bu elde edilen parça ile modüler gözetici hesaplanır. 5. adımda otomata olarak oluşturulan spesifikasyonlarla ilgili olan sistem parçaları Şekil 6.7'de görülmektedir.

Sisteme ait APN modelindeki mevkilerden, p1 ve p2 ile seçme alanı, p3 ve p4 ile halka kolu (kapasitesi bir), p5 ve p6 ile birleştirme alanı, p7 ve p8 ile itici selenoidin çalışması ve son olarak, p9 ve p10 mevkileri ile de döner selenoidin çalışması modellenmiştir. Birinci spesifikasyon (SP1), itici selenoid, seçme alanı ve halka kolu ile ilgilidir. Bu nedenle bu spesifikasyonla ilgili sistem parçası (PLANT1) p1, p2, p3, p4, p7 ve p8 mevkileri ve bağlantılarını içermektedir. Benzer olarak ikinci spesifikasyon ise halka toplama kolu, döner selenoid ve birleştirme alanı ile ilgili olduğundan bu spesifikasyona

ait sistem parçası (PLANT2) p3, p4 p5, p6, p9 ve p10 mevkileri ve bağlantılarını içermektedir.

Spesifikasyonlardan birincisiyle (SP1) ilgili olan sistem modeli parçası (PLANT1) Şekil 6.8'de görülmektedir. PLANT1'e ait otomata modeli p1, p2, p3, p4, p7 ve p8 tampon modellerinin senkron kompozisyonu ile aşağıda görüldüğü gibi elde edilmektedir.

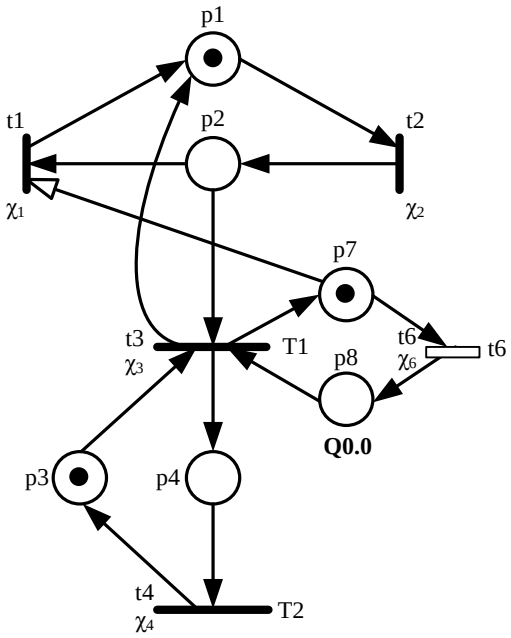
PLANT1 = Sync(P1,P2) (2,3) Blocked_events = None
PLANT1 = Sync(PLANT1,P3) (4,7) Blocked_events = None
PLANT1 = Sync(PLANT1,P4) (4,7) Blocked_events = None
PLANT1 = Sync(PLANT1,P7) (8,15) Blocked_events = None
PLANT1 = Sync(PLANT1,P8) (8,15) Blocked_events = None



Şekil 6.7 Sisteme ait spesifikasyonlarla ilgili sistem parçaları

RW gözeticisinin elde edilmesi için TCT yazılımında bulunan *Supcon(.)* ve *Condat(.)* komutları yardımıyla PLANT1 ve SP1 kullanılarak ilk spesifikasyonu yerine getirecek

modüler gözetici aşağıdaki gibi hesaplanmaktadır. Elde edilen gözeticinin otomata modeli Şekil 6.9'da görülmektedir.



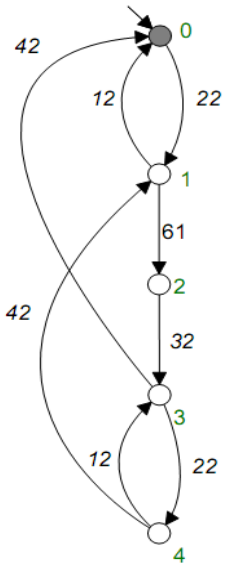
Şekil 6.8 İlk spesifikasyon SP1 ile ilgili sistem parçası (PLANT1)

MODSUP1 = Supcon(PLANT1,SP1) (5,8)
CDAT1 = Condat(PLANT1,MODSUP1) Controllable.

MODSUP1 # states: 5 state set: 0 ... 4 initial state: 0
marker states: 0 vocal states: none # transitions: 8
transitions:
[0, 22, 1] [1, 12, 0] [1, 61, 2] [2, 32, 3]
[3, 22, 4] [3, 42, 0] [4, 12, 3] [4, 42, 1]

Kontrol verisi aşağıda görülen liste şeklindedir. Bu listede gözeticinin hangi durumunda hangi olayın yasaklanacağı görülmektedir.

CDAT1:
Control data:
0: 61
3: 61
4: 61



Şekil 6.9 Birinci spesifikasyon SP1'e ait modüler gözeticinin (MODSUP1) otomata modeli

Spesifikasyonlardan ikincisiyle (SP2) ilgili olan sistem modeli parçası (PLANT2) Şekil 6.10'da görülmektedir. PLANT2'ye ait otomata modeli p3, p4, p5, p6, p9 ve p10 tampon modellerinin senkron kompozisyonu ile aşağıda görüldüğü gibi elde edilir.

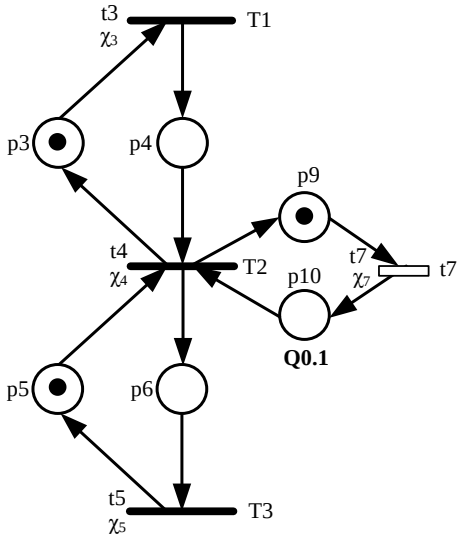
PLANT2 = Sync(P3,P4) (2,2) Blocked_events = None
PLANT2 = Sync(PLANT2,P5) (4,5) Blocked_events = None
PLANT2 = Sync(PLANT2,P6) (4,5) Blocked_events = None
PLANT2 = Sync(PLANT2,P9) (8,13) Blocked_events = None
PLANT2 = Sync(PLANT2,P10) (8,13) Blocked_events = None

İkinci spesifikasyona ait modüler gözetici aşağıdaki gibi hesaplanmıştır. Hesaplanan gözeticinin otomata modeli Şekil 6.11'de görülmektedir.

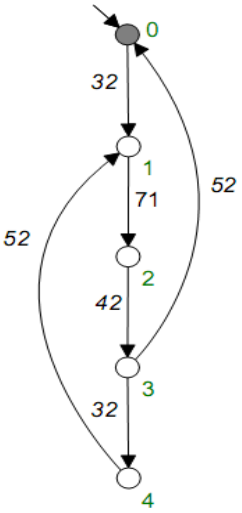
MODSUP2 = Supcon(PLANT2,SP2) (5,6)
CDAT2 = Condat(PLANT2,MODSUP2) Controllable.

MODSUP2 # states: 5 state set: 0 ... 4 initial state: 0
marker states:0 vocal states: none # transitions: 6
transitions:
[0, 32, 1] [1, 71, 2] [2, 42, 3] [3, 32, 4] [3, 52, 0] [4, 52, 1]

CDAT2
0: 71
3: 71
4: 71



Şekil 6.10 İkinci spesifikasyon SP2'yle ilgili sistem parçası (PLANT2)



Şekil 6.11 İkinci spesifikasyon SP2'ye ait modüler gözeticinin (MODSUP2) otomata modeli

7. Adım: Bu adımda elde edilen modüler gözeticilerin herhangi bir ikileme veya bloklanmaya sebep olup olmadığı TCT yardımıyla aşağıdaki gibi test edilir.

Öncelikle sistem için yekpare (monolithic) gözeticinin hesaplanması gerekmektedir. Bu hesaplama aşağıda görülen TCT işlemleri kullanılarak gerçekleştirilmiştir.

```
PLANT = Sync(P1,P2)  (2,3)  Blocked_events = None
PLANT = Sync(PLANT,P3)  (4,7)  Blocked_events = None
PLANT = Sync(PLANT,P4)  (4,7)  Blocked_events = None
PLANT = Sync(PLANT,P5)  (8,16)  Blocked_events = None
```

PLANT = Sync(PLANT,P6) (8,16) Blocked_events = None
PLANT = Sync(PLANT,P7) (16,34) Blocked_events = None
PLANT = Sync(PLANT,P8) (16,34) Blocked_events = None
PLANT = Sync(PLANT,P9) (32,80) Blocked_events = None
PLANT = Sync(PLANT,P10) (32,80) Blocked_events = None

SPEC = Sync(SP1,SP2) (8,20) Blocked_events = None
MONOSUPER = Supcon(PLANT,SPEC) (12,23)

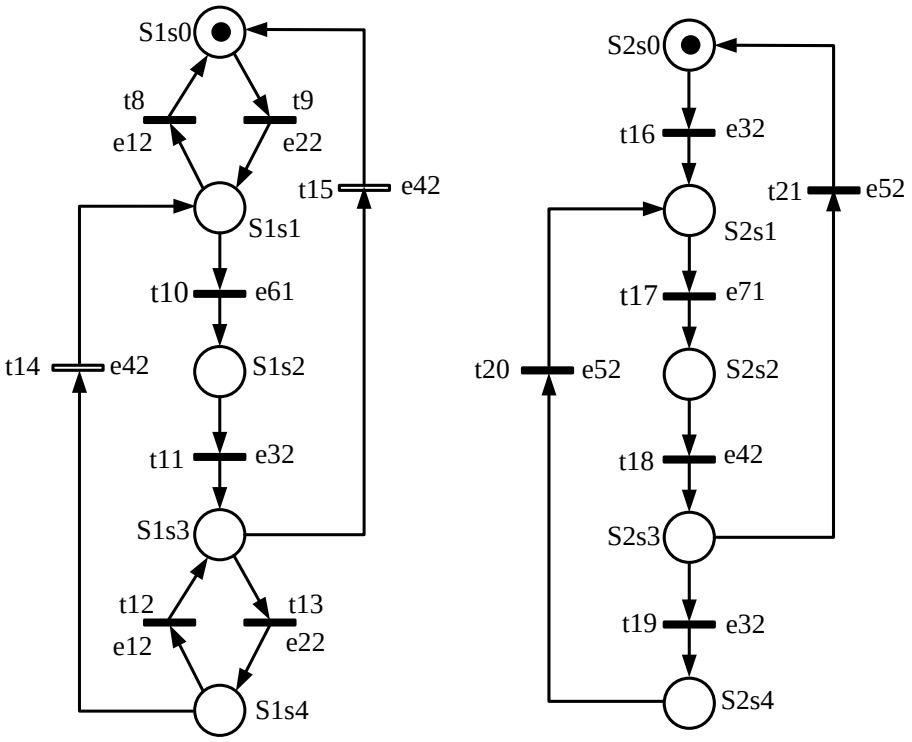
TEST = Sync(MODSUP1,MODSUP2) (12,23) Blocked_events = None
TEST = Sync(TEST,PLANT) (12,23) Blocked_events = None
true = Isomorph(MONOSUPER,TEST;identity)

Elde edilen iki gözeticinin birbiri ile ikilem oluşturmadığı TCT yazılımının *Nonconflict(.)* komutu ile test edilmektedir. Deneysel endüstriyel üretim sistemi için elde edilmiş olan iki modüler gözeticinin birbiri ile ikilem oluşturmadığı aşağıdaki TCT işleminden görülmektedir.

true = Nonconflict(MODSUP1,MODSUP2)

8. Adım: Elde edilen modüler gözeticiler MODSUP1 ve MODSUP2'ye ait auto-net modelleri sırası ile Şekil 6.12.(a) ve Şekil 6.12.(b)'de görülmektedir.

9. Adım: Bu adımda, Şekil 6.12.(a) ve (b)'de görülen modüler gözeticilerin auto-net modelleri, Şekil 5.49'da görülen sistemin APN modeline yasaklama okları ile bağlanarak, deneysel endüstriyel üretim sistemini kontrol edecek kapalı çevrim melez model elde edilir. TCT yardımıyla elde edilen kontrol verisi Şekil 6.13'te görüldüğü gibi yasaklama oklarına dönüştürülmektedir. Deneysel üretim sistemini kontrol edecek kapalı çevrim modüler melez denetleyici modeli Şekil 6.14'te görülmektedir.



Şekil 6.12 a) Şekil 6.9'da görülen MODSUP1'in auto-net modeli,
b) Şekil 6.11'de görülen MODSUP2'nin auto-net modeli

CDAT1	Yasaklama Oku	CDAT2	Yasaklama Oku
0: 61	S1s0 t6	0: 71	S2s0 t7
3: 61	S1s3 t6	3: 71	S2s3 t7
4: 61	S1s4 t6	4: 71	S2s4 t7

Şekil 6.13 Kontrol verilerinin (CDAT1 ve CDAT2) yasaklama oku karşılıkları

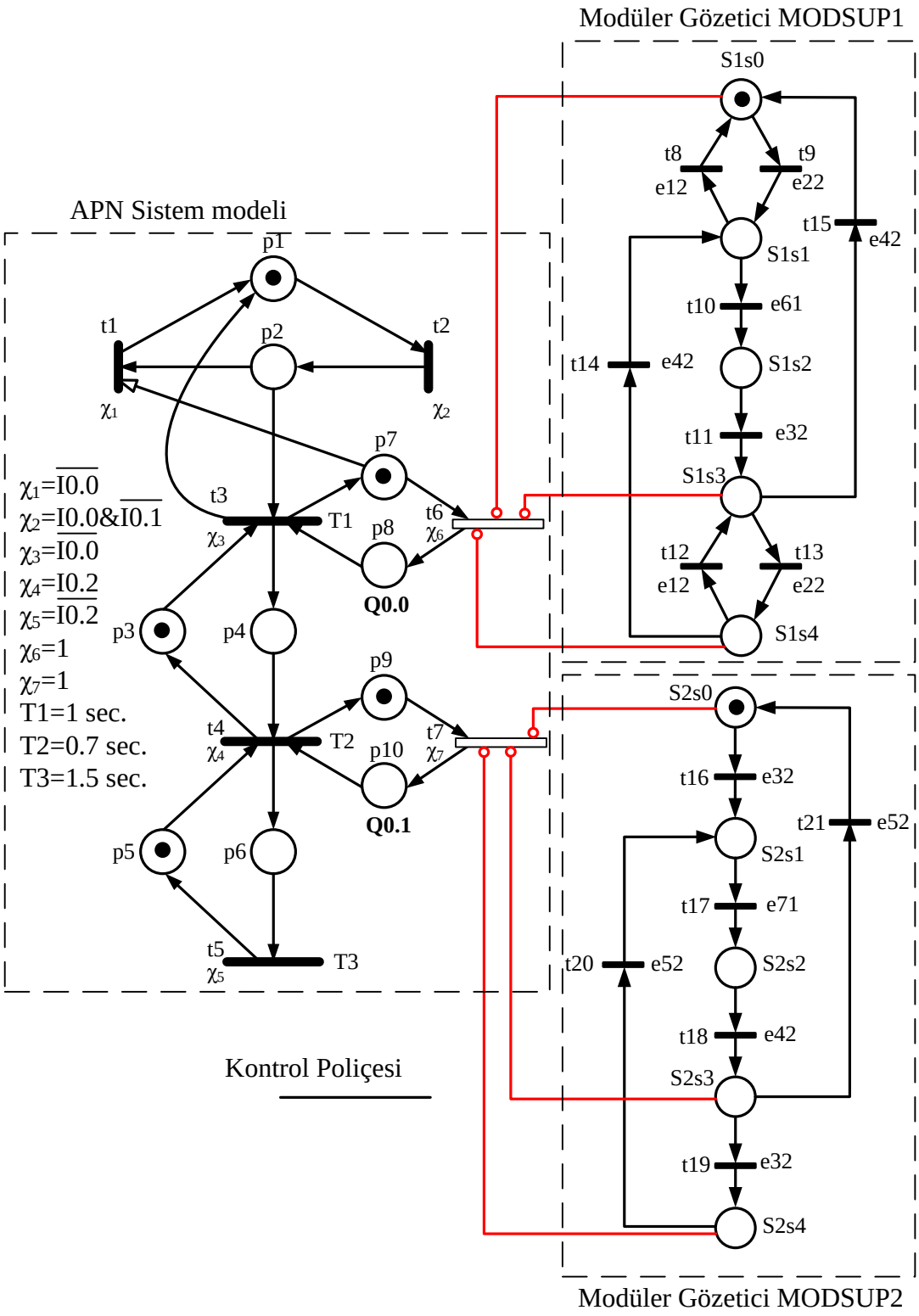
10. Adım: Son adımda elde edilen kapalı çevrim melez modelin PLC ile gerçekleştirilmesi gerekmektedir. Yekpare melez denetleyici modelinin PLC ile nasıl gerçekleştirileceği tezin üçüncü bölümünde ve [67]'de detaylı olarak açıklanmıştır. Şekil 6.14'te görülen kapalı çevrim modüler melez denetleyici yapısının PLC koduna çevrilebilmesi için her bir mevkiye bir hafıza bitinin atanması gerekmektedir. Şekil 6.14'te görülen melez model için kullanılan hafıza bitleri Çizelge 6.1'de görülmektedir.

Ayrıca, gözeticilerde kullanılan olayları (e12, e22, e32, e42, e52, e61 ve e71) simgelemek üzere birer hafıza biti atanmıştır.

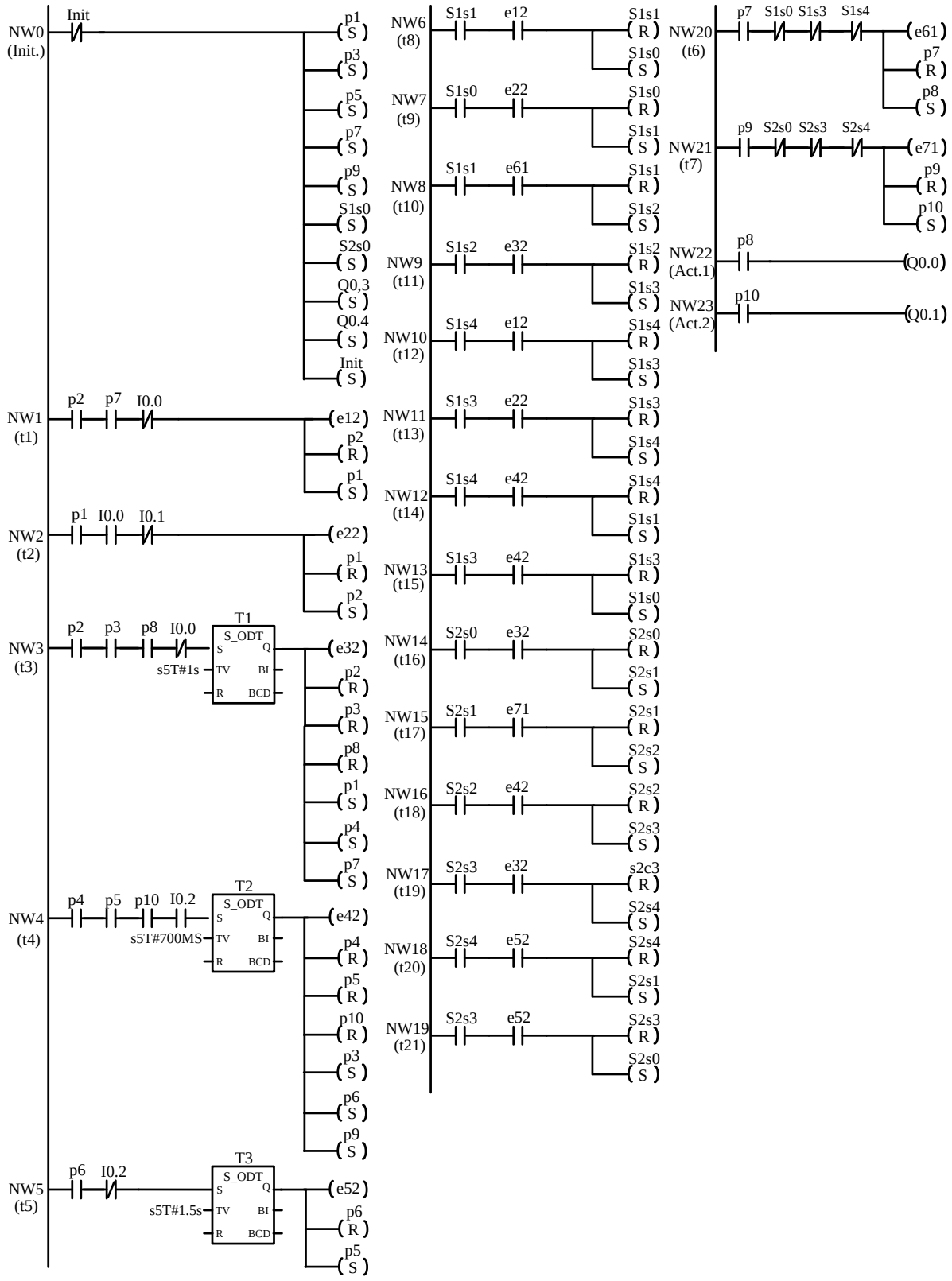
Bu uygulamada Siemens S7-300 PLC içeren deney düzeneği kullanılmıştır. APN modelinde bulunan zamanlı geçişler için "On Delay Timer" yani düz zaman rölesi kullanılmıştır. Düz zaman rölesinin çalışması Ek 1'de açıklanmıştır. T1 (1 s. zaman gecikmeli), T2 (0.7 s. zaman gecikmeli) ve T3 (1.5 s. zaman gecikmeli) düz zaman röleleri sırası ile APN modelindeki t3, t4 ve t5 geçişlerine atanmıştır.

Çizelge 6.1 PLC kodunda kullanılan hafıza bitleri

Sembolik ad	Hafıza biti	Sembolik ad	Hafıza biti
p1	M0.0	S2s0	M1.7
p2	M0.1	S2s1	M2.0
p3	M0.2	S2s2	M2.1
p4	M0.3	S2s3	M2.2
p5	M0.4	S2s4	M2.3
p6	M0.5	e12	M2.4
p7	M0.6	e22	M2.5
p8	M0.7	e32	M2.6
p9	M1.0	e42	M2.7
p10	M1.1	e52	M3.0
init	M1.1	e61	M3.1
S1s0	M1.2	e71	M3.2
S1s1	M1.3		
S1s2	M1.4		
S1s3	M1.5		
S1s4	M1.6		



Şekil 6.14 Halka toplama kolu kapasitesinin 1 olduğu durum için kapalı çevrim modüler melez denetleyici modeli



Şekil 6.15 Şekil 6.14'te görülen melez denetleyici modelinden elde edilen PLC merdiven diyagramı kodu

Şekil 6.14'te görülen melez denetleyici modelinden elde edilen PLC merdiven (ladder) diyagramı kodu Şekil 6.15'te görülmektedir. Şekil 6.15'te görülen PLC kodunun ilk basamağında başlangıç işlemleri yapılmıştır. Başlangıçta, APN modelinde bulunan p1, p3, p5, p7, p9 mevkileri ile modüler gözeticilerin auto-net modelindeki S1s0 ve S2s0 mevkilerinde jeton bulunduğundan, bu mevkilere ait hafıza bitleri set edilmektedir. Ayrıca bu ilk basamakta, Q0.3 ve Q0.4 çıkışları set edilerek alt ve üst konveyör motorları sürekli çalıştırılmaktadır.

Şekil 6.15'te görülen kodun NW1 ile NW5 arasındaki basamaklarında APN modelinde bulunan kontrol edilemeyen geçişler gerçekleştirilmiştir. Bu gerçekleştirme işleminde, jeton aktarma lojiği (Token Passing Logic) metodu kullanılmıştır. Bu yöntemle ilgili bilgiler tezin ikinci kısmında mevcuttur. Bu metoda göre, Petri ağı modelindeki her bir mevki için bir hafıza biti atanır. Atanan hafıza bitinin set olması o mevkide jeton bulunması, reset olması ise o mevkide jeton bulunmaması anlamına gelir. Petri ağı modelinde bulunan bir jetonun; bir mevkiden diğer bir mevkiye geçişi, harici tetikleme koşulunun sağlanması ile gerçekleşir. Merdiven diyagramı kodunda jetonun ağdaki dolaşımı, geçişe atanmış ilgili harici tetikleme koşulunun sağlanması ile giriş mevkisi için atanan hafıza bitinin reset, çıkış mevkisi için atanan hafıza bitinin de set edilmesi ile sağlanmaktadır. APN modelinde bulunan geçişlere karşılık gelen olay işaretleri (e12, e22, e32, e42, e52, e61 ve e71), her bir geçişin tetiklenmesi durumunda ilgili set ve reset işlemlerinden önce normal bir çıkış bobini kullanılarak ilgili hafıza bitine atama işlemi yapılarak elde edilmiştir. Örneğin, APN modelinde bulunan t1 geçişi NW1 nolu basamakta gerçekleştirilmiştir. APN modelinde, t1 geçişinin tetiklenebilmesi için, p2 ve p7 mevkilerinde jeton bulunması {p2 ve p7 etiketli hafıza bitlerinin set ('1') olması} ve $\chi_1 = \overline{I0.0}$ harici tetikleme şartının sağlanması gerekmektedir. Bu geçişin tetiklenmesi ile p2 mevkisinden jeton alınarak p1 mevkisine aktarılmaktadır. Yani, p2 etiketli hafıza biti resetlenip, p1 etiketli hafıza biti ise set edilmektedir. t1 geçişinin tetiklenmiş olduğunu bildiren "12" etiketli olayı simgelemek üzere kullanılan e12 hafıza bitine normal çıkış bobini ile atama yapılmıştır. Bu atama ile t1 geçişinin tetiklenmesi sırasında e12 hafıza biti çıkış bobini yardımıyla sadece bir PLC tarama süresince aktif olmakta ve tetikleme şartının kalkması ile bir sonraki PLC program taramasıyla beraber aktif olmaktan çıkmaktadır.

NW6-NW13 arasındaki basamaklarda ilk gözetici olan MODSUP1, NW14-NW19 arasındaki basamaklarda ise ikinci modüler gözetici olan MODSUP2 gerçekleştirilmektedir. APN modelinde bulunan kontrol edilebilir geçişler t6 ve t7 ise NW20 ve NW21'de gerçekleştirilmiştir. Bu basamaklardan da görüldüğü gibi modüler gözetici mevkilerinden bu geçişlere bağlı olan yasaklama okları (ilgili mevkileri karakterize eden hafıza bitlerinin reset olduğu) göz önünde bulundurularak kontrol poliçesi sağlanmaktadır. Son olarak NW22 ve NW23'te p8 ve p10 mevkilerine atanmış olan aksiyonlar gerçekleştirilmiştir. Bu kod, Simatic Manager isimli program kullanılarak derlenmiş ve PLC'ye yüklenerek denenmiştir. Yapılan denemelerde elde edilen modelin ve kodun istenilen spesifikasyonları yerine getirdiği gözlemlenmiştir.

Önerilen modüler yaklaşımla diğer yaklaşımları (yekpare ve indirgenmiş) karşılaştırmak için, aynı deney seti için elde edilen ve tezin üçüncü bölümünde yer alan (Şekil 3.14) kapalı çevrim yekpare denetleyici ile tezin dördüncü bölümünde yer alan (Şekil 4.14) indirgenmiş denetleyici modellerini birlikte değerlendirelim. Şekil 6.14'te görülen ve modüler yaklaşımla elde edilen kapalı çevrim denetleyici için **A** (modeldeki tüm mevki ve geçişlerin toplam sayısı) ***41*** (20 adet mevki 21 adet geçiş)'dir. Şekil 3.14'te görülen ve yekpare (monolithic) yaklaşımla elde edilen denetleyici için ise **A**'nın ***52*** (22 mevki ve 30 geçiş) olduğu kolaylıkla hesaplanabilmektedir. Deneysel endüstriyel üretim sistemi için hesaplanan indirgenmiş gözeticili kapalı çevrim melez kontrol sistemi (Şekil 4.14) için ise **A**'nın **45** (17 mevki ve 28 geçiş) olduğu görülmektedir. **A**'nın küçük olması gerçekleştirme aşamasında daha küçük bir denetleyici yapısı anlamına gelmektedir. Yani, modüler yaklaşım kullanılarak elde edilen denetleyici yapısı, hem yekpare yaklaşım hem de indirgenmiş yaklaşım ile elde edilen denetleyici yapılarından daha küçüktür. Denetleyici yapısının daha küçük olması, gerçekleştirme aşamasında daha az PLC kodu ile aynı görevin yerine getirileceğini göstermektedir.

Modüler yaklaşım ile elde edilen denetleyiciyi gerçekleştirmek için yazılan ve Şekil 6.15'te görülen PLC kodunun Siemens S7-300 (CPU319) PLC'nin hafızasında kapladığı alan **292** byte'dır. Şekil 3.14'te ve Şekil 4.14'te görülen yekpare ve indirgenmiş melez denetleyici yapılarının PLC ile gerçekleştirilmeleri için yazılan merdiven diyagramı kodları sırası ile Şekil 3.15'te ve Şekil 4.16'da görülmektedir. Bu kodların Siemens S7-300 (CPU319) PLC'nin hafızasında kapladığı alanlar sırası ile **382** ve **380** byte'tır. Bu karşılaştırmadan da görüldüğü gibi modüler yaklaşım kullanılarak

elde edilen merdiven diyagramı kodu daha az hafıza alanı kullanarak aynı kontrol işlemini yerine getirmektedir.

6.2.2 Halka toplama kolu kapasitesinin 2 olduğu durum

Halka toplama kolu kapasitesinin 2 olduğu durum için modüler gözeticiler aşağıdaki adımlar takip edilerek elde edilmiştir.

1. Adım: Deneysel endüstriyel üretim sisteminin kontrol edilmemiş APN modeli Şekil 6.16'da görülmektedir. Halka toplama kolu kapasitesinin bir olduğu durumu modelleyen APN modeline eklenen p11 ve p12 mevkileri ile halka toplama kolu kapasitesi bir arttırılmıştır.

2. Adım: Şekil 6.16'da görülen APN modelinin 1 sınırlı olduğu ve her mevkide en fazla bir jeton bulanabileceği kolaylıkla görülmektedir.

3. Adım: Şekil 6.16'da görülen APN modeli basit bir yapıya sahip olduğu için bir sadeleştirme tekniği uygulanmasına gerek görülmemiştir.

4. Adım: Şekil 6.16'da görülen sistemin kontrol edilmemiş APN modelinin otomataya çevrilmesi gerekmektedir. Bu APN modelindeki her bir mevkinin aşağıda verilen TCT olay kodlaması göz önünde bulundurularak elde edilen tampon otomata modelleri Şekil 6.17'de görülmektedir. TCT olay kodlaması yapılırken, TCT yazılımının gereksinimi olarak kontrol edilebilir olaylar için tek sayılar, kontrol edilemeyen olaylar için ise çift sayılar etiket olarak atanmıştır.

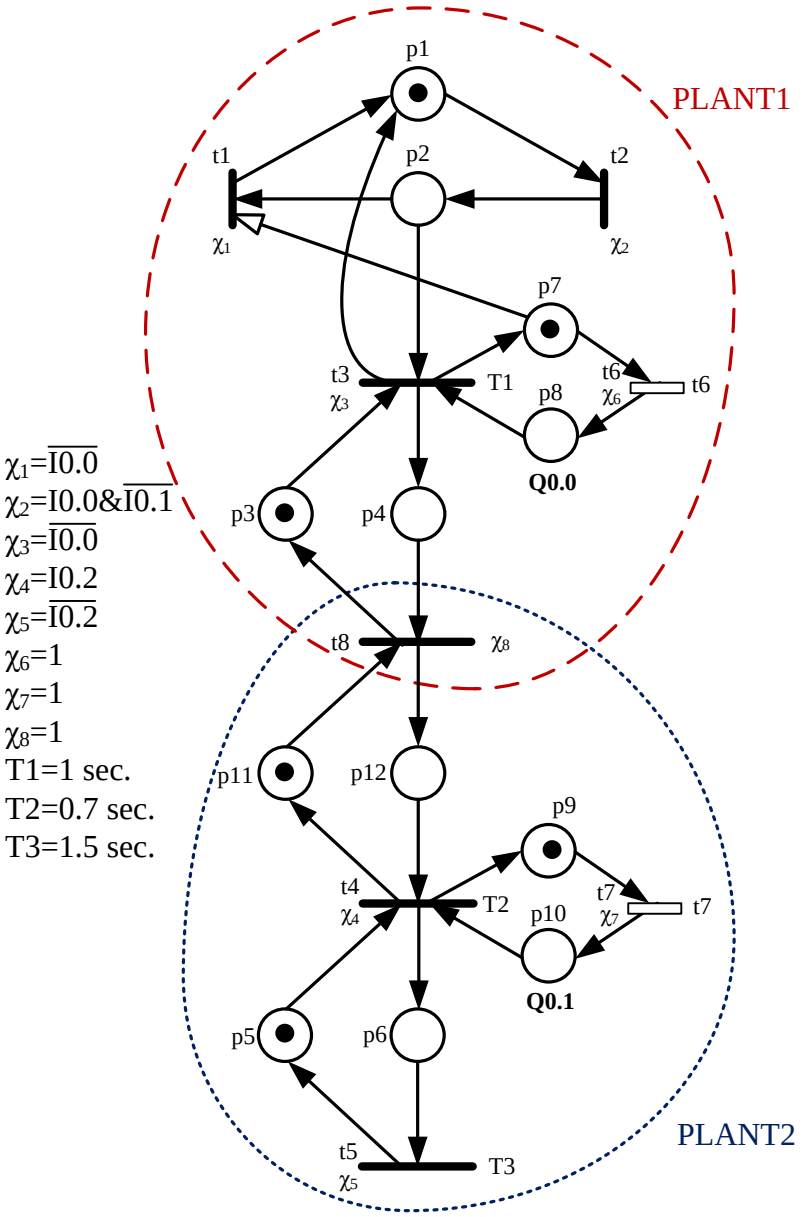
TCT olay etiketleri

APN geçişi :	t1	t2	t3	t4	t5	t6	t7	t8
TCT etiketi :	12	22	32	42	52	61	71	82

5. Adım: Sisteme ait spesifikasyonlar aşağıdaki gibidir,

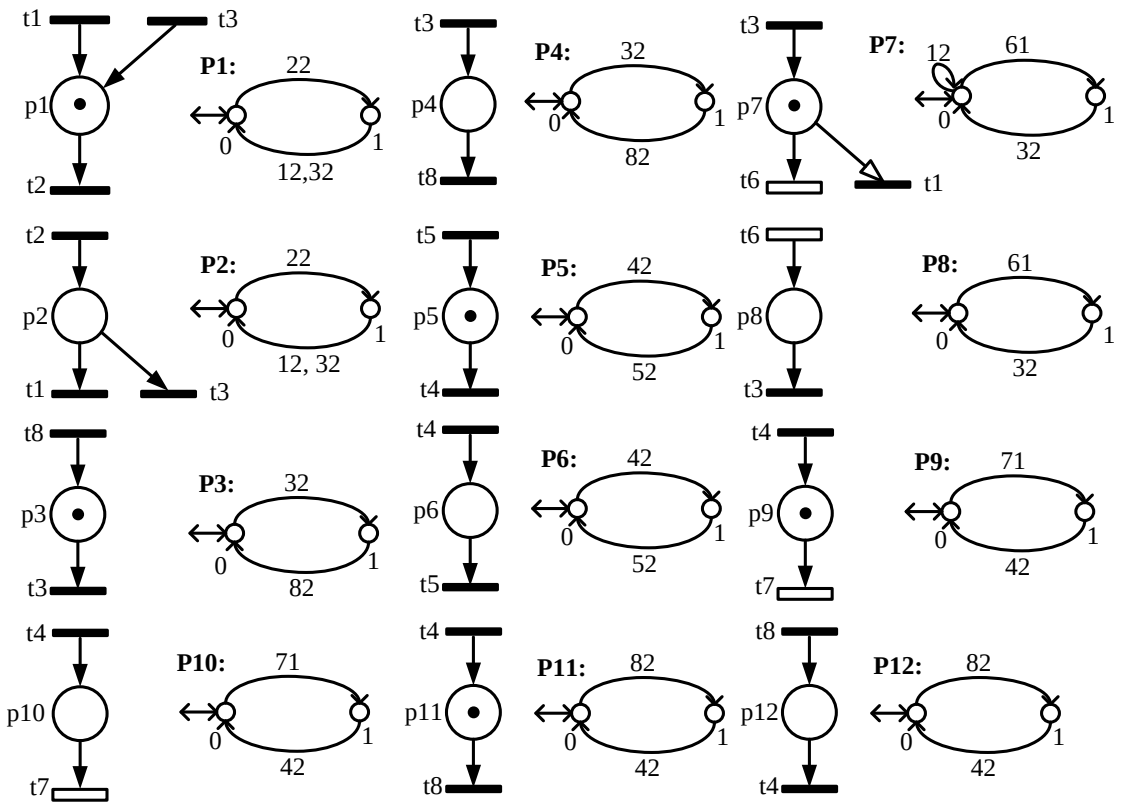
1. Seçme selenoidi (itici selenoid) seçme alanında bir halka olduğunda ve halka toplama kolunda en az bir boş yer olduğunda çalışacaktır (SP1).

2. Birleştirme selenoidi (döner selenoid) halka toplama kolunda en az bir halka olduğunda ve birleştirme noktası boş olduğunda çalışacaktır (SP2).

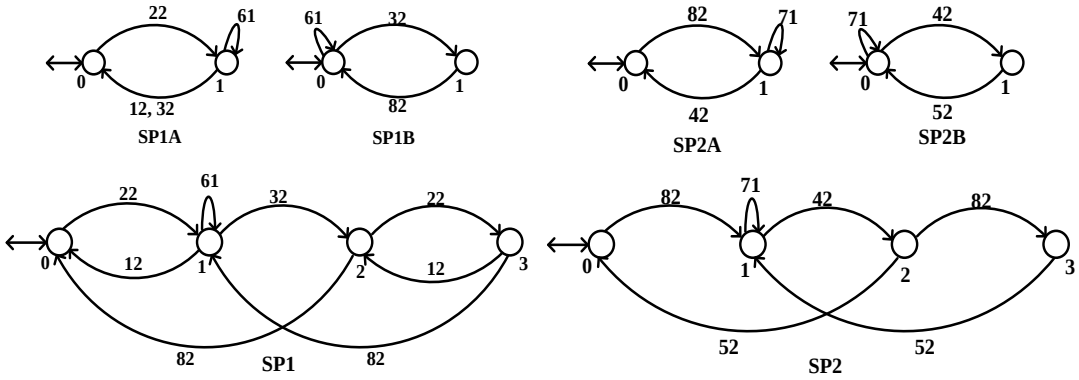


Şekil 6.16 Halka toplama kolu kapasitesinin 2 olduğu durum için APN modeli

Spesifikasyonlara ait otomata modelleri Şekil 6.18'de görülmektedir. Bir önceki örnekte olduğu gibi spesifikasyonlara ait otomata modelleri, her bir spesifikasyon parça parça (SP1A, SP1B, SP2A ve SP2B) modellenerek bu parçaların birleştirilmesi ile elde edilmiştir. Her bir spesifikasyonla ilgili olan sistem parçası Şekil 6.16'da görülmektedir. SP1 ile ilgili olan sistem parçası PLANT1, SP2 ile ilgili olan sistem parçası ise PLANT2 olarak adlandırılmıştır.



Şekil 6.17 Şekil 6.16'da görülen APN modelindeki mevkilerin otomata karşılıkları



Şekil 6.18 Spesifikasyonların otomata modelleri

6. Adım: Modüler gözeticiler aşağıdaki TCT işlemleri ile elde edilmiştir.

P1 = Create(P1,[mark 0],[tran [0,22,1],[1,12,0],[1,32,0]]) (2,3)
P2 = Create(P2,[mark 0],[tran [0,22,1],[1,12,0],[1,32,0]]) (2,3)
P3 = Create(P3,[mark 0],[tran [0,32,1],[1,82,0]]) (2,2)
P4 = Create(P4,[mark 0],[tran [0,32,1],[1,82,0]]) (2,2)
P5 = Create(P5,[mark 0],[tran [0,42,1],[1,52,0]]) (2,2)
P6 = Create(P6,[mark 0],[tran [0,42,1],[1,52,0]]) (2,2)

P7 = Create(P7,[mark 0],[tran [0,12,0],[0,61,1],[1,32,0]]) (2,3)
P8 = Create(P8,[mark 0],[tran [0,61,1],[1,32,0]]) (2,2)
P9 = Create(P9,[mark 0],[tran [0,71,1],[1,42,0]]) (2,2)
P10 = Create(P10,[mark 0],[tran [0,71,1],[1,42,0]]) (2,2)
p11 = Create(p11,[mark 0],[tran [0,82,1],[1,42,0]]) (2,2)
p12 = Create(p12,[mark 0],[tran [0,82,1],[1,42,0]]) (2,2)

SP1A = Create(SP1A,[mark 0],[tran [0,22,1],[1,12,0],[1,32,0],[1,61,1]]) (2,4)
SP1B = Create(SP1B,[mark 0],[tran [0,32,1],[0,61,0],[1,82,0]]) (2,3)
SP2A = Create(SP2A,[mark 0],[tran [0,82,1],[1,42,0],[1,71,1]]) (2,3)
SP2B = Create(SP2B,[mark 0],[tran [0,42,1],[0,71,0],[1,52,0]]) (2,3)
SP1 = Sync(SP1A,SP1B) (4,8) Blocked_events = None
SP2 = Sync(SP2A,SP2B) (4,6) Blocked_events = None

PLANT1 = Sync(P1,P2) (2,3) Blocked_events = None
PLANT1 = Sync(PLANT1,P3) (4,7) Blocked_events = None
PLANT1 = Sync(PLANT1,P4) (4,7) Blocked_events = None
PLANT1 = Sync(PLANT1,P7) (8,15) Blocked_events = None
PLANT1 = Sync(PLANT1,P8) (8,15) Blocked_events = None
PLANT2 = Sync(P11,P12) (2,2) Blocked_events = None
PLANT2 = Sync(PLANT2,P5) (4,5) Blocked_events = None
PLANT2 = Sync(PLANT2,P6) (4,5) Blocked_events = None
PLANT2 = Sync(PLANT2,P9) (8,13) Blocked_events = None
PLANT2 = Sync(PLANT2,P10) (8,13) Blocked_events = None

MODSUP1 = Supcon(PLANT1,SP1) (5,8)
CDAT1 = Condat(PLANT1,MODSUP1) Controllable.
MODSUP2 = Supcon(PLANT2,SP2) (5,6)
CDAT2 = Condat(PLANT2,MODSUP2) Controllable.

Hesaplamalar sonucunda bulunan modüler gözetici MODSUP1 ve kontrol verisi CDAT1 aşağıda görülen liste şeklindedir. Modüler gözetici MODSUP1'in otomata modeli Şekil 6.19.(a)'da görülmektedir.

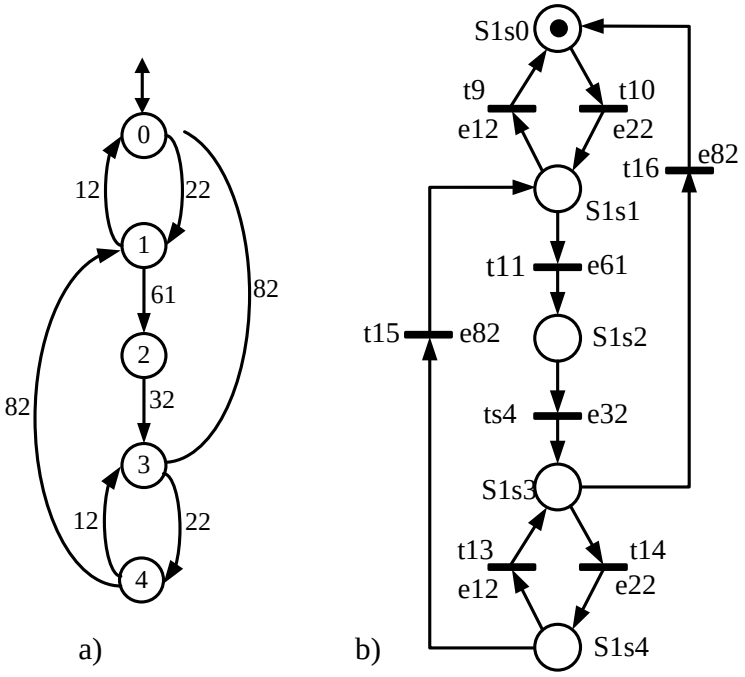
MODSUP1 # states: 5 state set: 0 ... 4 initial state: 0
marker states: 0 vocal states: none # transitions: 8
transitions: [0, 22, 1] [1, 12, 0] [1, 61, 2] [2, 32, 3][3, 22, 4] [3, 82, 0] [4, 12, 3] [4, 82, 1]

CDAT1:
Control data: 0: 61 3: 61 4: 61

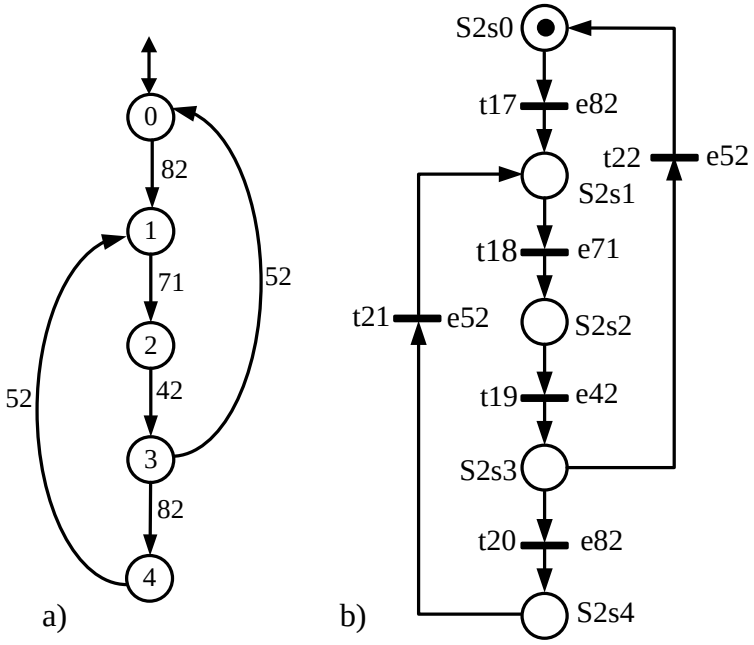
Hesaplamalar sonucunda bulunan modüler gözetici MODSUP2 ve kontrol verisi CDAT2 aşağıda görülen liste şeklindedir. Modüler gözetici MODSUP2'nin otomata modeli Şekil 6.20.(a)'da görülmektedir.

MODSUP2 # states: 5 state set: 0 ... 4 initial state: 0
marker states:0 vocal states: none # transitions: 8
transitions: [0, 82, 1] [1, 71, 2] [2, 42, 3] [3, 52,0] [3, 82,4] [4, 52, 1]

CDAT2:
Control data: 0: 71 3: 71 4: 71



Şekil 6.19 a) MODSUP1'in otomata modeli, b) MODSUP1'in auto-net modeli



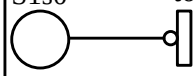
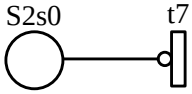
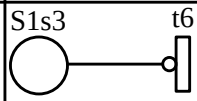
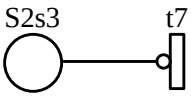
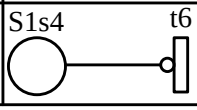
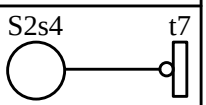
Şekil 6.20 a) MODSUP2'nin otomata modeli, b) MODSUP2'nin auto-net modeli

7. Adım: Bu adımda elde edilen modüler gözeticilerin herhangi bir ikileme veya bloklanmaya sebep olmadığı TCT yardımıyla aşağıdaki gibi test edilmiştir.

true = Nonconflict(MODSUP1,MODSUP2)
PLANT = Sync(PLANT1,PLANT2) (64,176) Blocked_events = None
SPEC = Sync(SP1,SP2) (16,44) Blocked_events = None
SUP = Supcon(PLANT,SPEC) (25,54)
TEST = Sync(MODSUP1,MODSUP2) (25,54) Blocked_events = None
TEST = Sync(TEST,PLANT) (25,54) Blocked_events = None
true = Isomorph(TEST,SUP;identity)

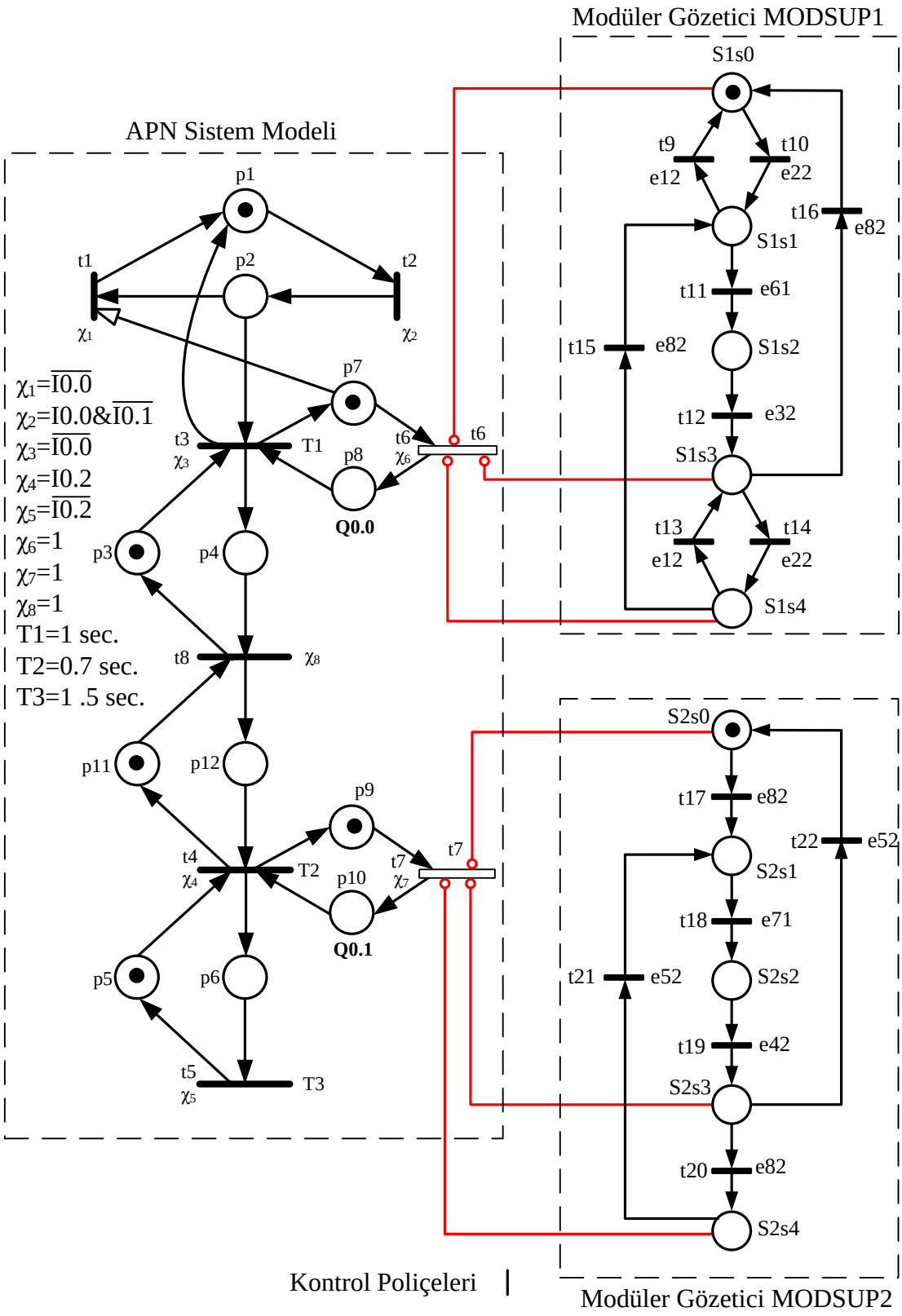
8. Adım: Elde edilen modüler gözeticiler MODSUP1 ve MODSUP2'ye ait otomata modellerinin auto-net karşılıkları sırası ile Şekil 6.19.(b) ve 6.20.(b)'de görülmektedir.

9. Adım: Bu adımda, elde edilen iki modüler gözetici MODSUP1 ve MODSUP2 kontrol verileri (CDAT1 ve CDAT2) Şekil 6.21'de görüldüğü gibi yasaklama oklarına dönüştürülmektedir. Hesaplanan modüler gözeticiler bu yasaklama okları kullanılarak Şekil 6.22'de görüldüğü gibi kontrol edilmemiş sisteme ait APN modeline bağlanmıştır. Bu işlem sonucunda, deneysel endüstriyel üretim sisteminin halka toplama kolu kapasitesinin 2 olduğu durum için denetleyici elde edilmiş olur.

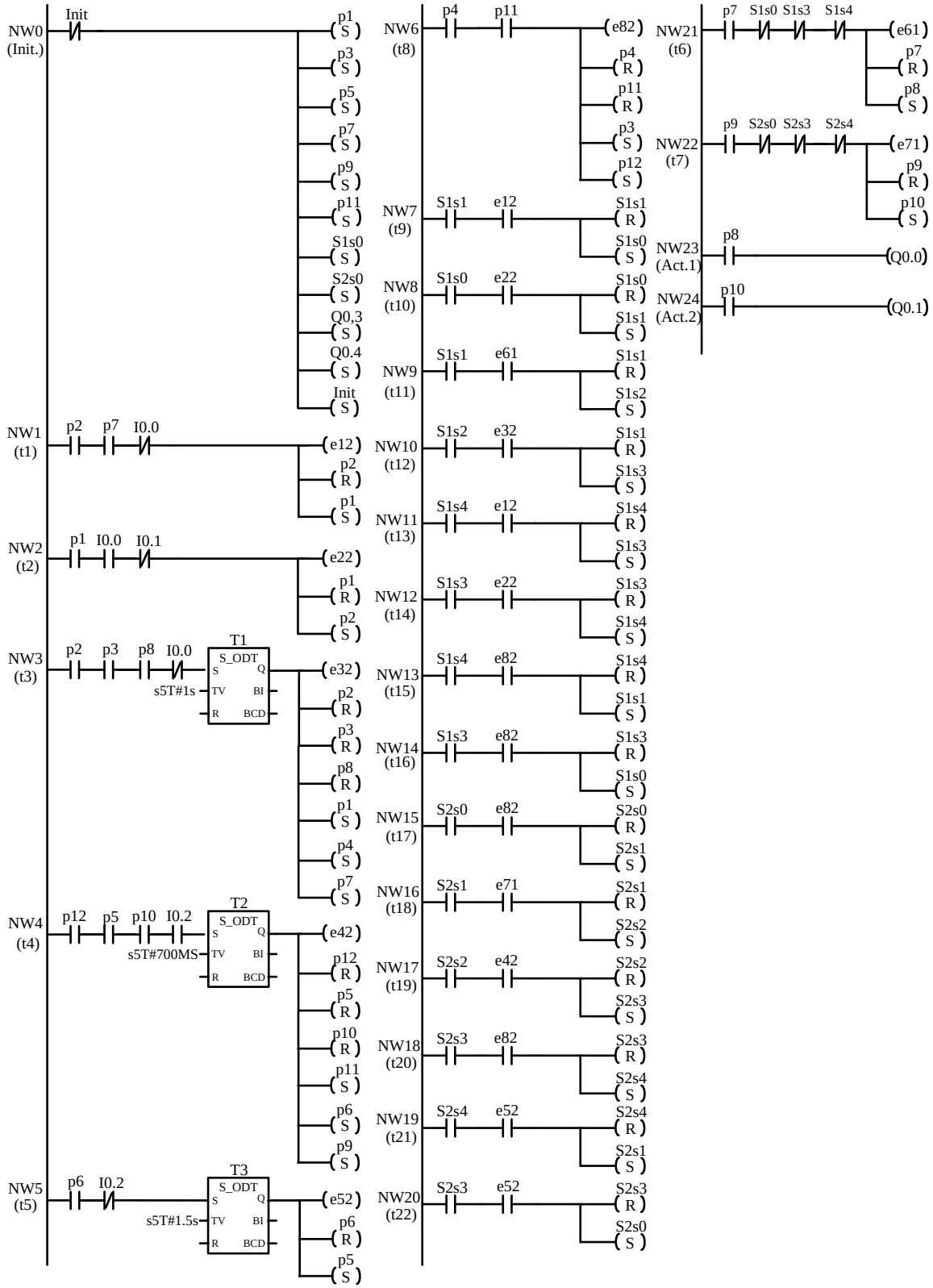
CDAT1	Yasaklama Oku	CDAT2	Yasaklama Oku
0: 61		0: 71	
3: 61		3: 71	
4: 61		4: 71	

Şekil 6.21 Kontrol verilerinin (CDAT1 ve CDAT2) yasaklama oku karşılıkları

10. Adım: Şekil 6.22'de görülen halka toplama kolu kapasitesinin 2 olduğu durum için kapalı çevrim modüler melez denetleyici modelini gerçekleştiren PLC merdiven diyagramı kodu Şekil 6.23'te görülmektedir.



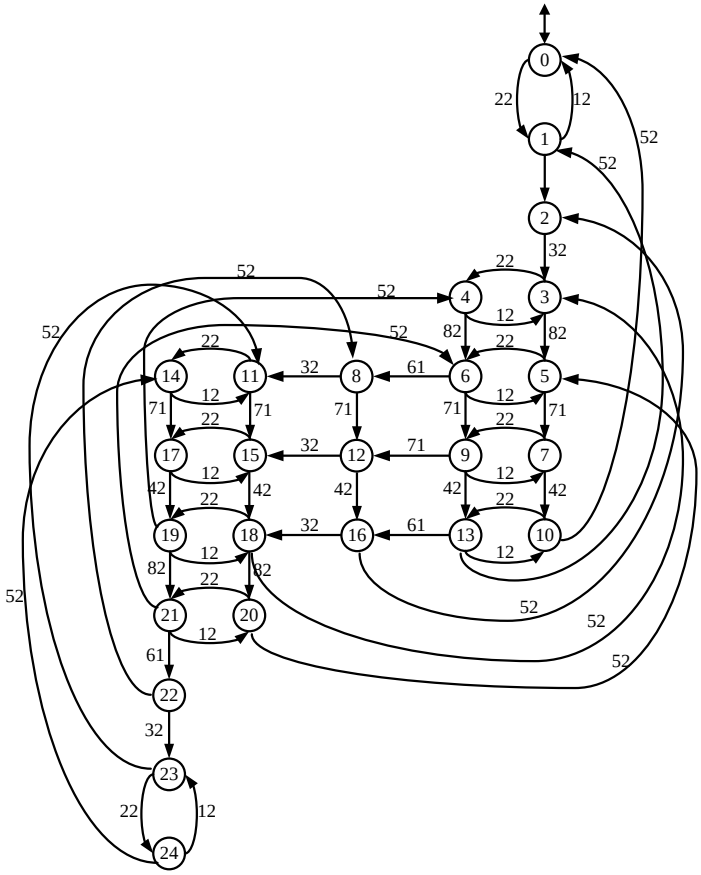
Şekil 6.22 Halka toplama kolu kapasitesinin 2 olduğu durum için kapalı çevrim modüler melez denetleyici modeli



Şekil 6.23 Halka toplama kolu kapasitesinin 2 olduğu durum için
kapalı çevrim modüler melez denetleyici modeline ait PLC merdiven diyagramı kodu

Halka toplama kolu kapasitesinin 2 olduğu durum için yekpare gözetici *SUP = Supcon(PLANT,SPEC) (25,54)* TCT işlemi ile elde edilmektedir. Bu işlemden elde

edilen yekpare gözetici 24 durum ve 54 geçiş'ten oluşmaktadır. Bu gözeticinin otomata modeli Şekil 6.24'te görülmektedir. Bu otomata modeli auto-net yapısına dönüştürülerek kontrol verisinin belirlediği yasaklama okları ile sisteme ait APN modeline bağlanarak kapalı çevrim yekpare denetleyici modeli elde edilecektir. Yekpare yaklaşımla elde edilecek kapalı çevrim denetleyici modeli için **A** (modeldeki tüm mevki ve geçişlerin toplam sayısı), APN modelinde 12 mevki ve 8 geçiş, auto-net modelinde ise 24 mevki ve 54 geçiş olduğundan toplam ***99*** olarak hesaplanmıştır. Aynı kontrol işlemini yerine getiren kapalı çevrim modüler melez denetleyici modeli için **A**, APN modelinde 12 mevki ve 8 geçiş, auto-net modellerinde ise 10 mevki ve 14 geçiş olduğundan toplam ***44*** olarak hesaplanmıştır. Buradan da görülebileceği gibi modüler yaklaşımla aynı kontrol işlemi daha küçük bir denetleyici yapısı ile yerine getirilebilmektedir. PLC kodunun hafızada kapladığı alan açısından bakıldığında modüler denetleyici için yazılan PLC kodu (Şekil 6.23) hafızada ***308*** Byte'lık alan kaplamaktadır. Bu uygulama için yekpare melez denetleyiciyi gerçekleştiren kod hafızada ***674*** Byte'lık alan kaplamaktadır. Hafızada kaplanan alan yönünden de modüler yaklaşımın kullanımı avantajlı olmaktadır.



Şekil 6.24 Halka toplama kolu kapasitesinin 2 olduğu durum için yekpare gözeticinin otomata modeli

6.2.3 Halka toplama kolu kapasitesinin 5 olduğu durum

Halka toplama kolu kapasitesinin 5 olduğu durum için modüler gözeticiler aşağıdaki adımlar takip edilerek elde edilmiştir.

1. Adım: Deneysel endüstriyel üretim sisteminin kontrol edilmemiş APN modeli Şekil 6.25'te görülmektedir. Halka toplama kolu kapasitesinin 2 olduğu durumu modelleyen APN modeline eklenen p13, p14, p15, p16, p17 ve p18 mevkileri ile halka toplama kolu kapasitesi beşe yükseltilmiştir.

2. Adım: Şekil 6.25'te görülen APN modelinin 1 sınırlı olduğu ve mevkilerinde en fazla bir jeton bulanabileceği kolaylıkla görülmektedir.

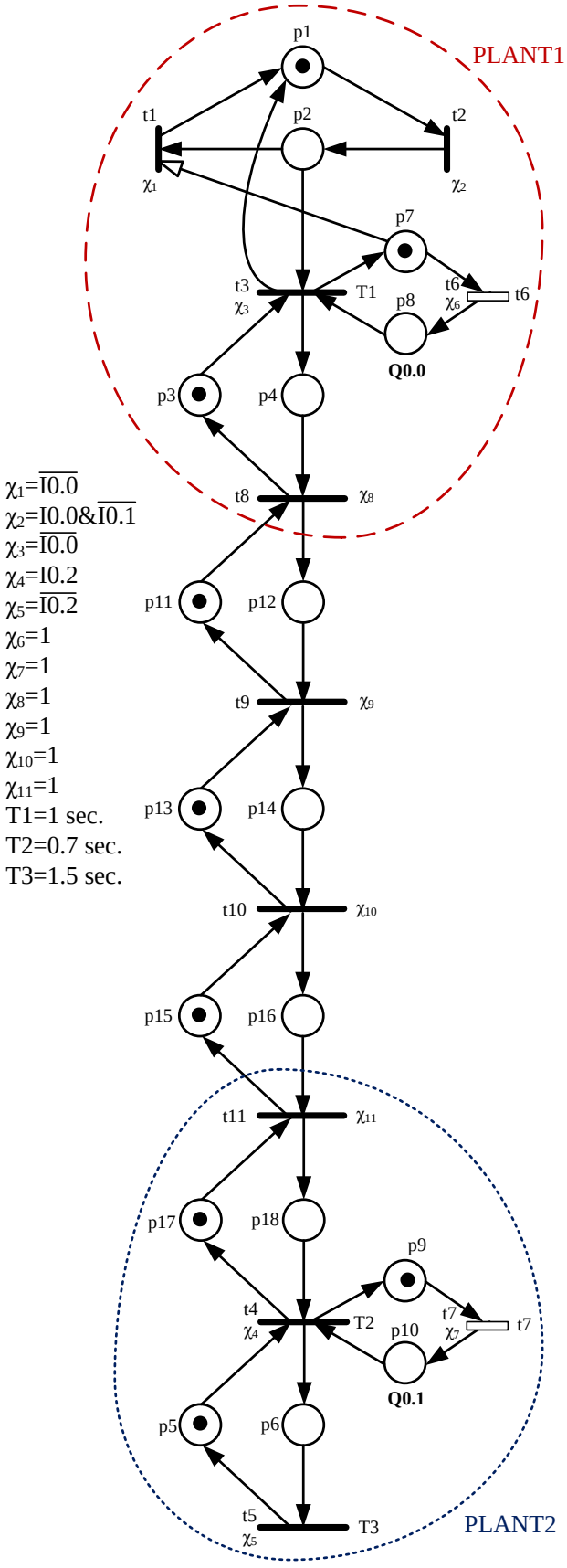
3. Adım: Şekil 6.25'te görülen APN modeli basit bir yapıya sahip olduğu için bir sadeleştirme tekniği uygulanmasına gerek görülmemiştir.

4. Adım: Şekil 6.25'te görülen sistemin kontrol edilmemiş APN modelinin otomataya çevrilmesi gerekmektedir. Bu APN modelindeki her bir mevkinin aşağıda verilen TCT olay kodlaması göz önünde bulundurularak elde edilen tampon otomata modelleri Şekil 6.26'da görülmektedir. TCT olay kodlaması yapılırken, TCT yazılımının gereksinimi olarak kontrol edilebilir olaylar için tek sayılar, kontrol edilemeyen olaylar için ise çift sayılar etiket olarak atanmıştır.

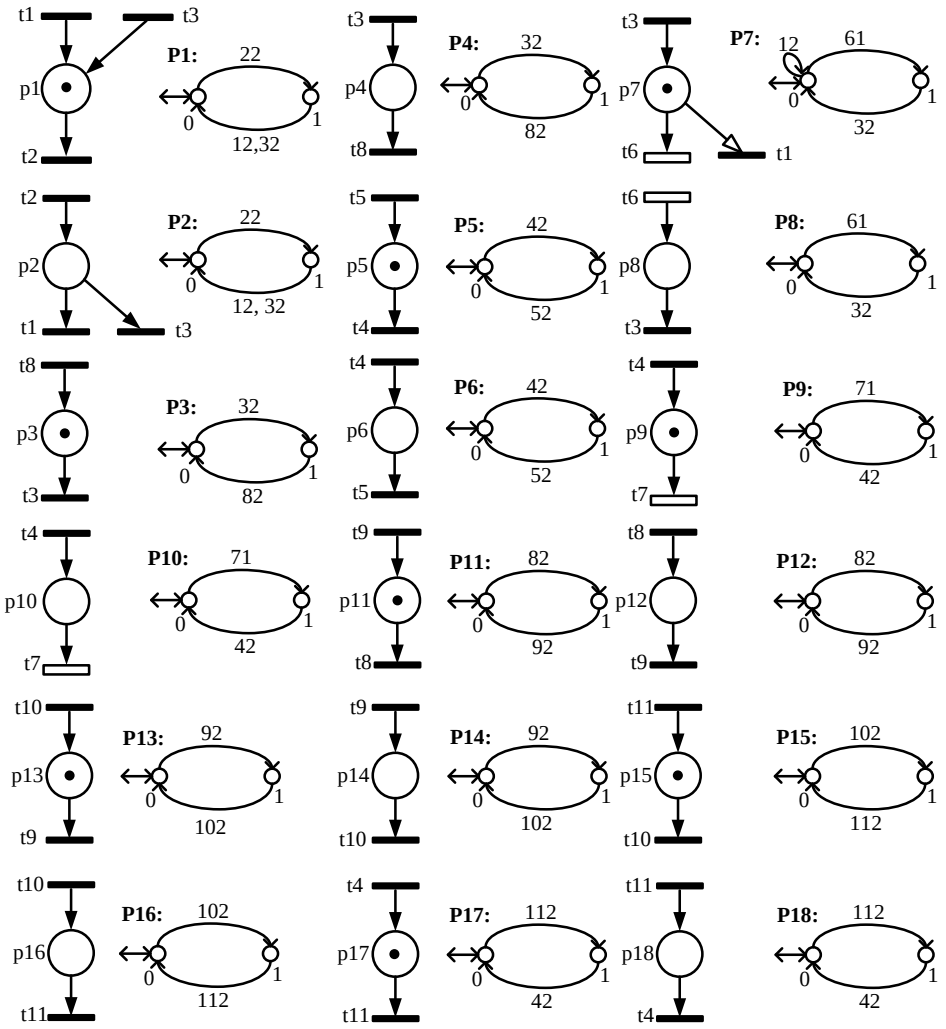
TCT olay etiketleri

APN geçişi: t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11
TCT etiketi:12	22	32	42	52	61	71	82	92	102	112

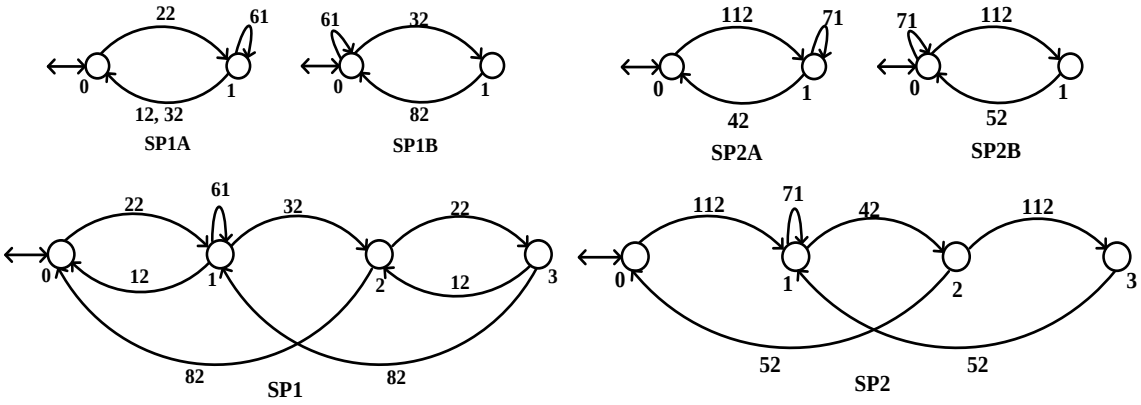
5. Adım: Sisteme ait spesifikasyonlar bir önceki örnekle aynı olup, spesifikasyonlara ait otomata modelleri Şekil 6.27'de görülmektedir. Önceki örnekte olduğu gibi spesifikasyonlara ait otomata modelleri, her bir spesifikasyon parça parça (SP1A, SP1B, SP2A ve SP2B) modellenerek bu parçaların birleştirilmesi ile elde edilmiştir. Her bir spesifikasyonla ilgili olan sistem parçası (PLANT1, PLANT2) Şekil 6.25'te görülmektedir. İlk spesifikasyon SP1 ile ilgili olan sistem parçası PLANT1, ikinci spesifikasyon SP2 ile ilgili olan sistem parçası ise PLANT2 olarak adlandırılmıştır.



Şekil 6.25 Halka toplama kolu kapasitesinin 5 olduğu durum için APN modeli



Şekil 6.26 Şekil 6.25'te görülen APN modelindeki mevkilerin otomata karşılıkları



Şekil 6.27 Spesifikasyonların otomata modelleri

6. Adım: Modüler gözeticiler aşağıdaki TCT işlemleri ile elde edilmiştir.

P1 = Create(P1,[mark 0],[tran [0,22,1],[1,12,0],[1,32,0]]) (2,3)
P2 = Create(P2,[mark 0],[tran [0,22,1],[1,12,0],[1,32,0]]) (2,3)
P3 = Create(P3,[mark 0],[tran [0,32,1],[1,82,0]]) (2,2)
P4 = Create(P4,[mark 0],[tran [0,32,1],[1,82,0]]) (2,2)
P5 = Create(P5,[mark 0],[tran [0,42,1],[1,52,0]]) (2,2)
P6 = Create(P6,[mark 0],[tran [0,42,1],[1,52,0]]) (2,2)
P7 = Create(P7,[mark 0],[tran [0,12,0],[0,61,1],[1,32,0]]) (2,3
P8 = Create(P8,[mark 0],[tran [0,61,1],[1,32,0]]) (2,2)
P9 = Create(P9,[mark 0],[tran [0,71,1],[1,42,0]]) (2,2)
P10 = Create(P10,[mark 0],[tran [0,71,1],[1,42,0]]) (2,2)
P11 = Create(P11,[mark 0],[tran [0,82,1],[1,92,0]]) (2,2)
P12 = Create(P12,[mark 0],[tran [0,82,1],[1,92,0]]) (2,2)
p13 = Create(p13,[mark 0],[tran [0,92,1],[1,102,0]]) (2,2)
p14 = Create(p14,[mark 0],[tran [0,92,1],[1,102,0]]) (2,2)
p15 = Create(p15,[mark 0],[tran [0,102,1],[1,112,0]]) (2,2)
p16 = Create(p16,[mark 0],[tran [0,102,1],[1,112,0]]) (2,2)
p17 = Create(p17,[mark 0],[tran [0,112,1],[1,42,0]]) (2,2)
p18 = Create(p18,[mark 0],[tran [0,112,1],[1,42,0]]) (2,2)

SP1A = Create(SP1A,[mark 0],[tran [0,22,1],[1,12,0],[1,32,0],[1,61,1]]) (2,4)
SP1B = Create(SP1B,[mark 0],[tran [0,32,1],[0,61,0],[1,82,0]]) (2,3)
SP2A = Create(SP2A,[mark 0],[tran [0,112,1],[1,42,0],[1,71,1]]) (2,3)
SP2B = Create(SP2B,[mark 0],[tran [0,42,1],[0,71,0],[1,52,0]]) (2,3)

PLANT1 = Sync(P1,P2) (2,3) Blocked_events = None
PLANT1 = Sync(PLANT1,P3) (4,7) Blocked_events = None
PLANT1 = Sync(PLANT1,P4) (4,7) Blocked_events = None
PLANT1 = Sync(PLANT1,P7) (8,15) Blocked_events = None
PLANT1 = Sync(PLANT1,P8) (8,15) Blocked_events = None
PLANT2 = Sync(P17,P18) (2,2) Blocked_events = None
PLANT2 = Sync(PLANT2,P5) (4,5) Blocked_events = None
PLANT2 = Sync(PLANT2,P6) (4,5) Blocked_events = None
PLANT2 = Sync(PLANT2,P9) (8,13) Blocked_events = None
PLANT2 = Sync(PLANT2,P10) (8,13) Blocked_events = None

SP1 = Sync(SP1A,SP1B) (4,8) Blocked_events = None
SP2 = Sync(SP2A,SP2B) (4,6) Blocked_events = None
SPEC = Sync(SP1,SP2) (16,56) Blocked_events = None

Modüler gözeticiler aşağıdaki gibi hesaplanmıştır.

MODSUP1 = Supcon(PLANT1,SP1) (5,8)
CDAT1 = Condat(PLANT1,MODSUP1) Controllable.
MODSUP2 = Supcon(PLANT2,SP2) (5,6)
CDAT2 = Condat(PLANT2,MODSUP2) Controllable.

Hesaplamalar sonucunda bulunan modüler gözetici MODSUP1 ve kontrol verisi CDAT1 aşağıda verilmiştir. Modüler gözetici MODSUP1'in otomata modeli Şekil 6.28.(a)'da görülmektedir.

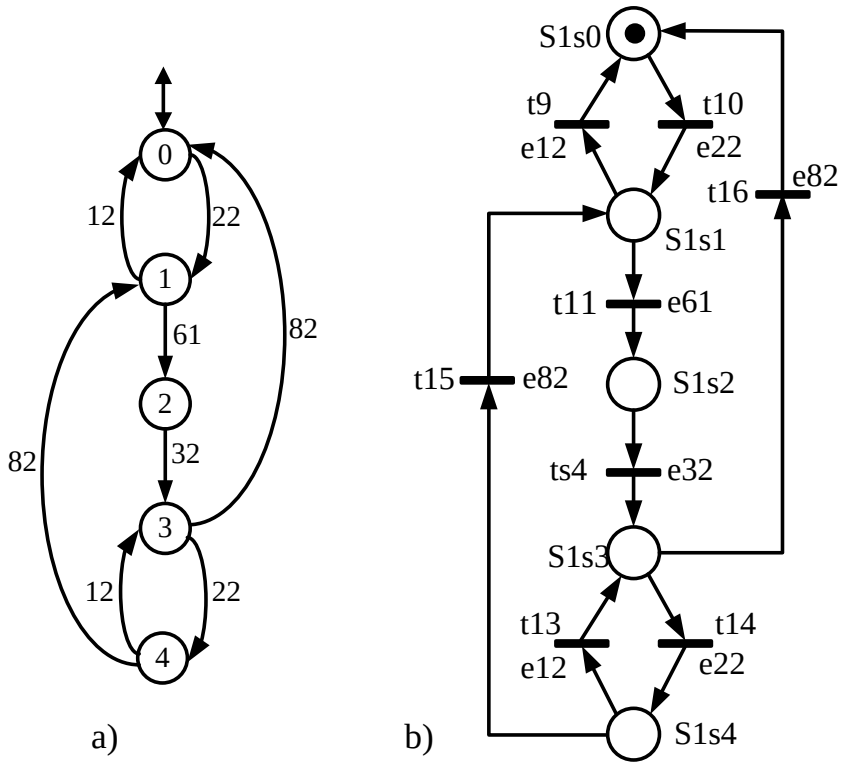
MODSUP1 # states: 5 state set: 0 ... 4 initial state: 0
marker states: 0 vocal states: none # transitions: 8
transitions: [0, 22, 1] [1, 12, 0] [1, 61, 2] [2, 32, 3][3, 22, 4] [3, 82, 0] [4, 12, 3] [4, 82, 1]

CDAT1:
Control data: 0: 61 3: 61 4: 61

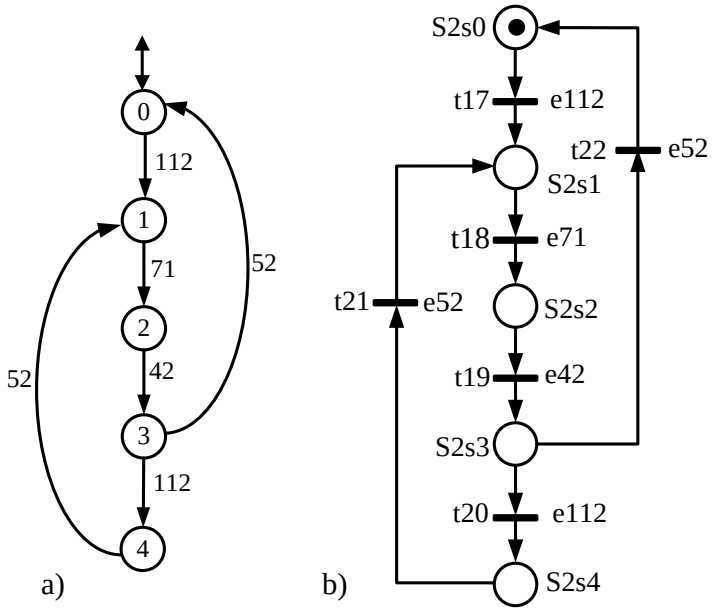
Hesaplamalar sonucunda bulunan modüler gözetici MODSUP2 ve kontrol verisi CDAT2 aşağıda verilmiştir. Modüler gözetici MODSUP2'nin otomata modeli Şekil 6.29.(a)'da görülmektedir.

MODSUP2 # states: 5 state set: 0 ... 4 initial state: 0
marker states:0 vocal states: none # transitions: 8
transitions: [0, 112, 1] [1, 71, 2] [2, 42, 3] [3, 112, 4] [3, 52, 0] [4, 52, 1]

CDAT2:
Control data: 0: 71 3: 71 4: 71



Şekil 6.28 a) MODSUP1'in otomata modeli, b) MODSUP1'in auto-net modeli



Şekil 6.29 a) MODSUP2'nin otomata modeli, b) MODSUP2'nin auto-net modeli

7. Adım: Bu adımda elde edilen modüler gözeticilerin herhangi bir ikileme veya bloklanmaya sebep olmadığı TCT yardımıyla aşağıdaki gibi test edilmiştir.

true = Nonconflict(MODSUP1,MODSUP2)

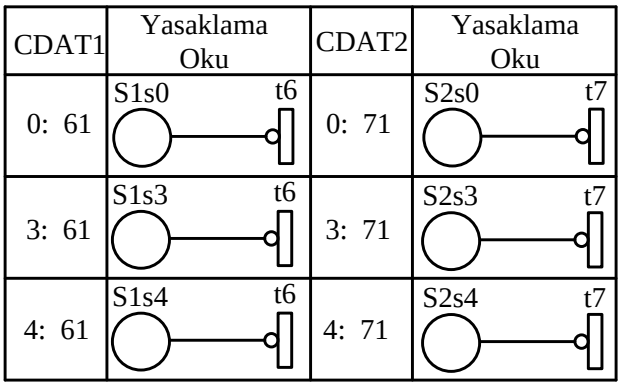
PLANT = Sync(PLANT1,PLANT2) (64,224) Blocked_events = None
PLANT = Sync(PLANT,P11) (128,480) Blocked_events = None
PLANT = Sync(PLANT,P12) (128,480) Blocked_events = None
PLANT = Sync(PLANT,P13) (256,1024) Blocked_events = None
PLANT = Sync(PLANT,P14) (256,1024) Blocked_events = None
PLANT = Sync(PLANT,P15) (512,1792) Blocked_events = None
PLANT = Sync(PLANT,P16) (512,1792) Blocked_events = None

SSPEC = Selfloop(SPEC,[92,102]) (16,88)
SUP = Supcon(PLANT,SSPEC) (200,580)
TEST = Sync(MODSUP1,MODSUP2) (25,70) Blocked_events = None
TEST = Sync(TEST,PLANT) (200,580) Blocked_events = None
true = Isomorph(SUP,TEST;identity)

8. Adım: Elde edilen modüler gözeticiler MODSUP1 ve MODSUP2'ye ait otomata modellerinin auto-net karşılıkları sırası ile Şekil 6.28.(b) ve Şekil 6.29.(b)'de görülmektedir.

9. Adım: Bu adımda, elde edilen iki modüler gözetici MODSUP1 ve MODSUP2 kontrol verileri (CDAT1 ve CDAT2) Şekil 6.30'da görüldüğü gibi yasaklama oklarına

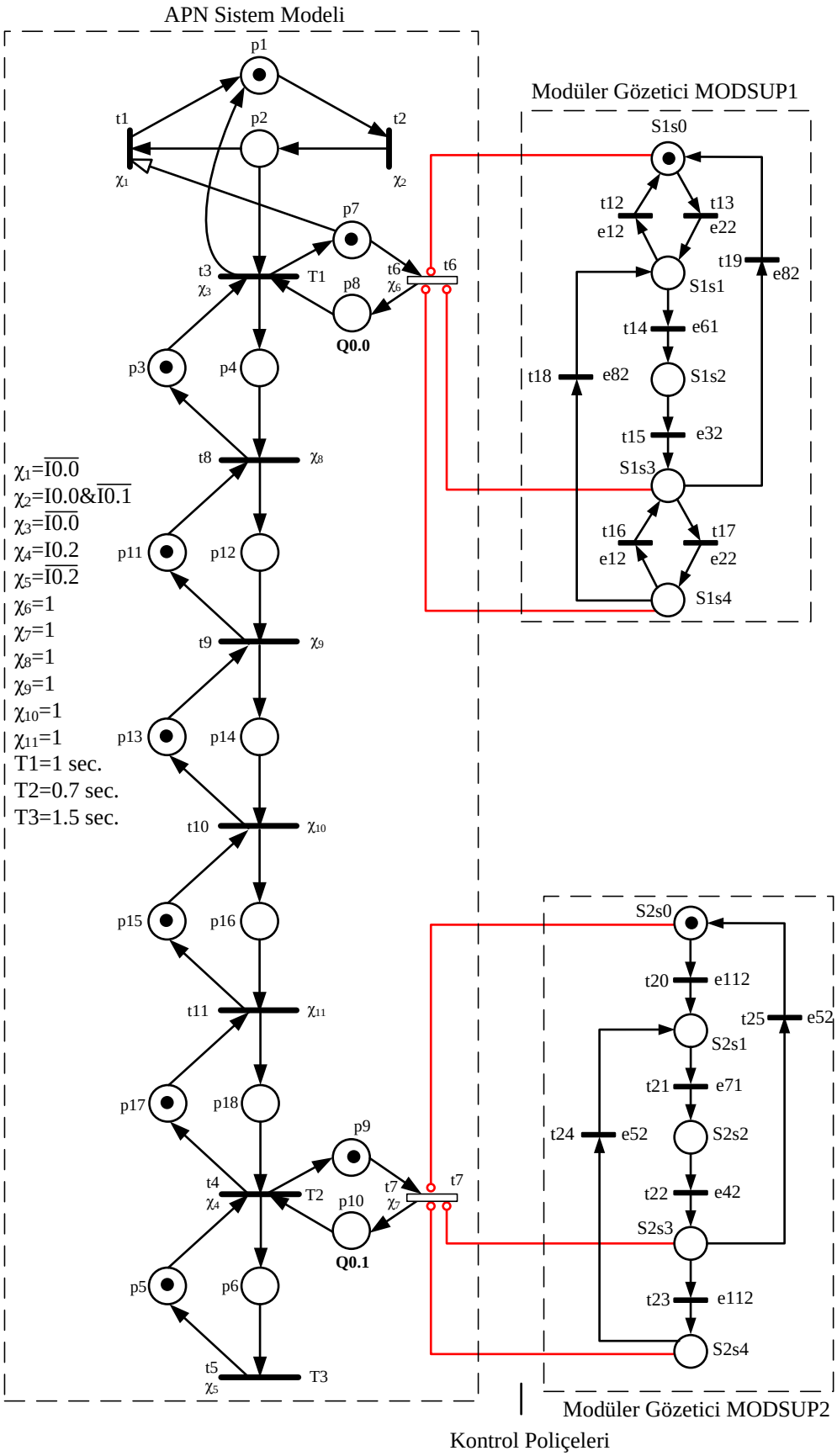
dönüştürülmektedir. Hesaplanan modüler gözeticiler bu yasaklama okları kullanılarak Şekil 6.31'de görüldüğü gibi kontrol edilmemiş sisteme ait APN modeline bağlanmıştır. Bu işlem sonucunda, deneysel endüstriyel üretim sisteminin halka toplama kolu kapasitesinin 5 olduğu durum için denetleyici elde edilmiş olur.



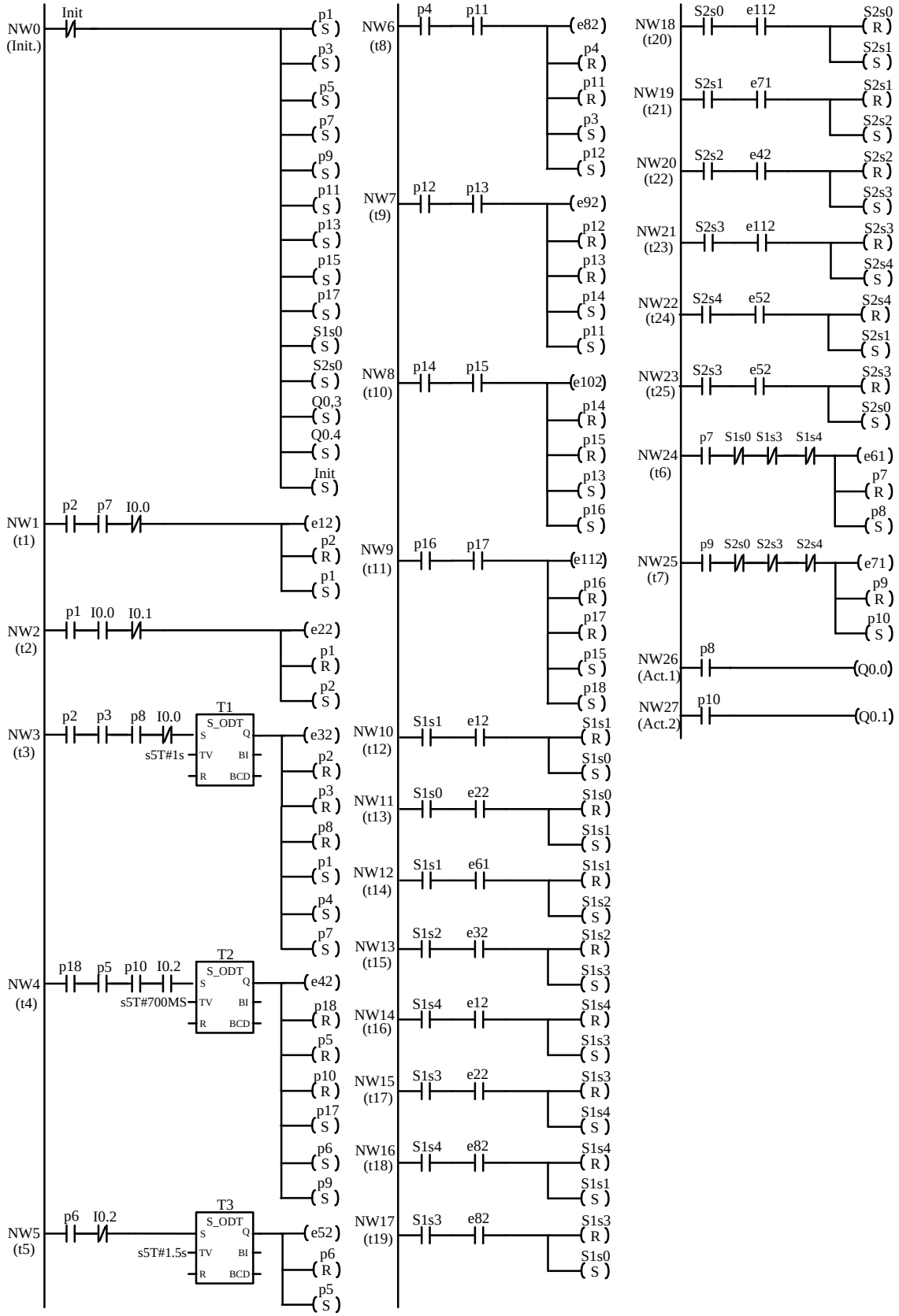
Şekil 6.30 Kontrol verilerinin (CDAT1 ve CDAT2) yasaklama oku karşılıkları

10. Adım: Şekil 6.31'de görülen halka toplama kolu kapasitesinin 5 halka olduğu durum için kapalı çevrim modüler melez denetleyici modelini gerçekleştiren PLC merdiven diyagramı kodu Şekil 6.32'de görülmektedir.

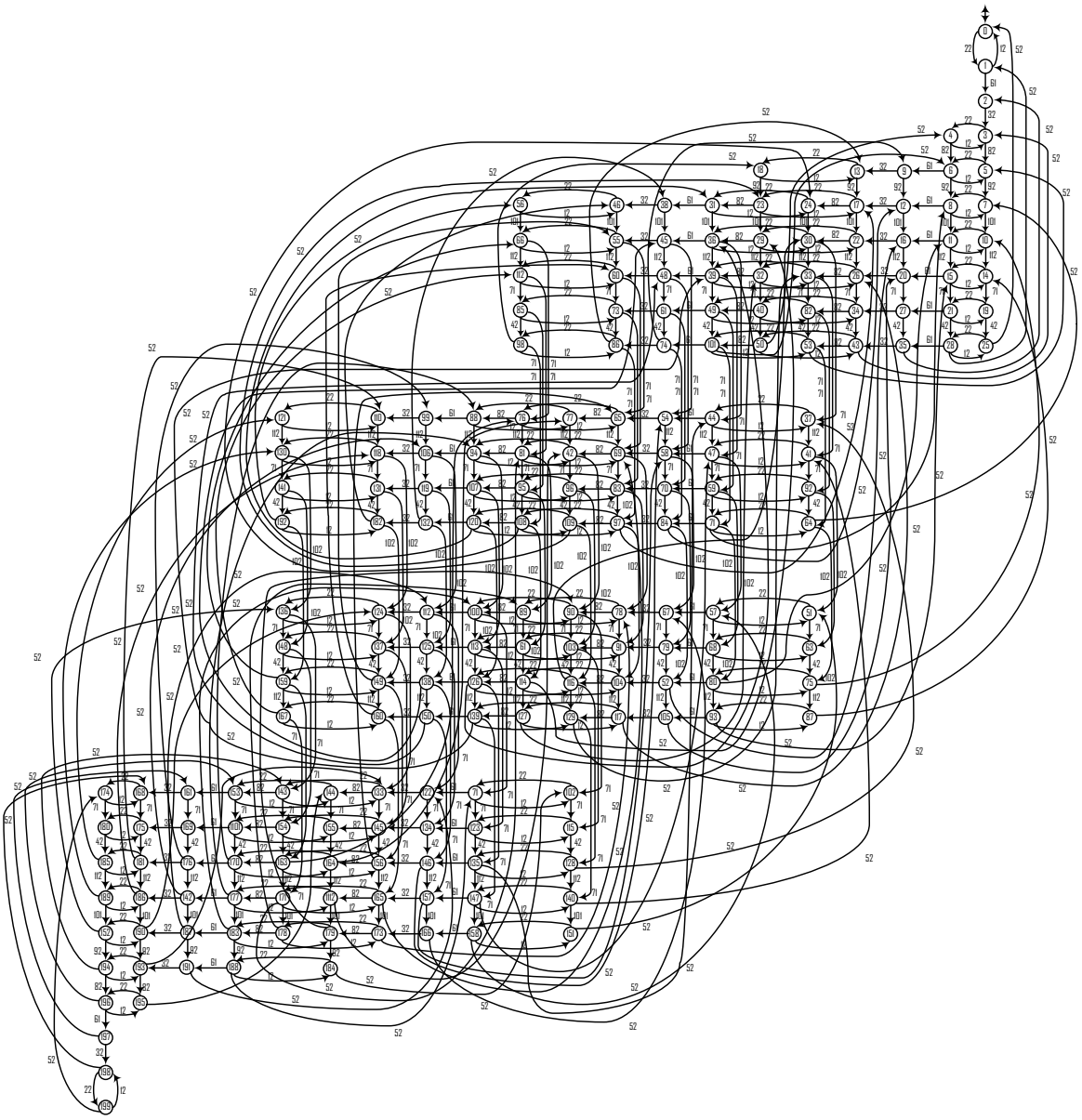
Halka toplama kolu kapasitesinin 5 olduğu durum için yekpare gözetici *SUP = Supcon(PLANT,SPEC) (200,580)* TCT işlemi ile elde edilmiştir. Bu işlemden elde edilen yekpare gözetici 200 durum ve 580 geçişten oluşmaktadır. Bu gözeticinin otomata modeli Şekil 6.33'te görülmektedir. Yekpare yaklaşımla elde edilen kapalı çevrim denetleyici modeli için **A** (modeldeki tüm mevki ve geçişlerin toplam sayısı), APN modelinde 18 mevki ve 11 geçiş, auto-net modelinde ise 200 mevki ve 580 geçiş olduğundan toplam ***809*** olarak hesaplanır. Aynı kontrol işlemini yerine getiren kapalı çevrim modüler melez denetleyici modeli için **A**, APN modelinde 18 mevki ve 11 geçiş, auto-net modellerinde ise 10 mevki ve 14 geçiş olduğundan toplam ***53*** olarak hesaplanır. Buradan da görülebileceği gibi modüler yaklaşımla aynı kontrol işlemi daha küçük bir denetleyici yapısı ile yerine getirilebilmektedir. PLC kodunun hafızada kapladığı alan açısından bakıldığında modüler denetleyici için yazılan PLC kodu (Şekil 6.32) hafızada ***356*** Byte'lık alan kaplamaktadır. Bu uygulama için yekpare melez denetleyiciyi gerçekleştiren kod hafızada ***5406*** Byte'lık alan kaplamaktadır. Hafızada kaplanan alan yönünden de modüler yaklaşımın kullanımı avantajlı olmaktadır.



Şekil 6.31 Halka toplama kolu kapasitesinin 5 olduğu durum için kapalı çevrim modüler melez denetleyici modeli



Şekil 6.32 Halka toplama kolu kapasitesinin 5 olduğu durum için kapalı çevrim modüler melez denetleyici modeline ait PLC kodu



Şekil 6.33 Halka toplama kolu kapasitesinin 5 olduğu durum için yekpare gözeticinin otomata modeli

6.3 Sonuç

Tezin bu kısmında önerilen modüler melez yapının avantajlarını gösterebilmek amacı ile deneysel endüstriyel üretim sistemi halka toplama kolu kapasitesinin 1, 2 ve 5 olduğu durumları için hem yekpare hem de modüler melez denetleyici yapıları hesaplanmıştır. Elde edilen denetleyici yapıları PLC ile gerçekleştirilmiş ve PLC merdiven diyagramı kodunun hafızada kapladığı alan karşılaştırılmıştır. Halka toplama

kolu kapasitesinin farklı durumları için elde edilen yekpare ve modüler denetleyicilerin toplam mevki ve geçiş sayıları ile bu denetleyicileri gerçekleştirmek için yazılan PLC merdiven diyagramı kodlarının boyutları Çizelge 6.2'de görülmektedir. Çizelgeden de görülebileceği gibi, aynı işlemi yerine getirmek üzere hesaplanan modüler melez denetleyicinin boyutu ve PLC kodunun hafızada kapladığı alan, yekpare melez denetleyicinin boyutundan çok daha küçüktür. Modüler melez denetleyici yapısı ile yekpare melez denetleyici arasındaki boyut farkı, sistem modelinin büyümesiyle çok fazla artmaktadır. Dolayısı ile sistem modelinin büyük olduğu durumlarda, yekpare melez denetleyici yapısı yerine modüler melez denetleyici yapısının kullanılmasının daha uygun olacağı sonucuna varılmıştır.

Çizelge 6.2 Deney setindeki halka toplama kolu kapasitesinin farklı durumları için hesaplanan denetleyici boyutları

Halka Kapasitesi	Yekpare Denetleyici		Modüler Denetleyici	
	Mevki ve Geçişlerin Toplam Sayısı (A)	PLC Kod Boyutu (Byte)	Mevki ve Geçişlerin Toplam Sayısı (A)	PLC Kod Boyutu (Byte)
1	52	382	41	292
2	99	674	44	308
5	809	5406	53	356

BÖLÜM VII

AYRIK OLAY SİSTEMLERİNİN KONTROLLÜ PETRİ AĞLARI VE RAMADGE-WONHAM YÖNTEMİ KULLANILARAK MELEZ KONTROLÜ

AOS'lerin kontrolünde yaygın olarak kullanılan Petri ağı temelli denetleyicilerin literatürde iki farklı şekli mevcuttur [29]. Birinci gruptaki denetleyiciler sistem tarafından üretilen her bir yeni olay (event) için hesaplanan bir fonksiyon şeklinde ifade edilen gönderim denetleyicileri olarak isimlendirilirler. Derlenmiş denetleyiciler olarak isimlendirilen ikinci gruptaki denetletleyicilerde ise, kontrol mekanizması bir ağ yapısı şeklinde gösterilmektedir. Denetleyicilerin bir ağ şeklinde ifade edilmesinin pek çok faydası vardır [29]: Öncelikle, çevrim-içi bir hesaplama gerektirmediğinden kontrol işlemi için gereken hesaplama daha hızlıdır. İkinci olarak, hem kontrol mekanizması hem de sistem modeli ağ yapısı şeklinde ifade edildiğinden aynı Petri ağı sistem yürütme algoritmaları her iki ağ yapısını gerçekleştirmek için kullanılabilir. Son olarak, kontrol edilecek sisteme ait kapalı çevrim kontrol modeli standart ağ elde etme yapıları kullanılarak gerçekleştirilebilir. Bu açıklanan üstünlüklerinden dolayı *derlenmiş denetleyiciler*, *gönderim denetleyicileri*ne göre daha fazla tercih edilmektedir.

Petri ağlarının genişletilmiş bir sınıfı olan Kontrollü Petri ağları ilk olarak Krogh [74] ile Ichikawa ve Hiraishi [75] tarafından önerilmiştir. CtlPN'ler ile AOS'lerin kontrolünde hesaplanan denetleyiciler genellikle gönderim denetleyicileridir. CtlPN modelindeki her farklı jeton dağılımı için yeni bir kontrol poliçesi hesaplanmaktadır. CtlPN için önerilen gönderim denetleyicileri, ağ yapısından farklı olarak genellikle bir algoritmadır. CtlPN için birleştirilmiş denetleyici (Combined Supervisor) yapıları da önerilmiştir [107]. Bu yapılarda ilk olarak Petri ağı formunda gözlemci (observer) sentezlenmiştir. Sentezlenen bu gözlemcinin jeton dağılımına göre kontrol kuralı algoritma olarak elde edilmiştir. Birleştirilmiş yapıların gönderim denetleyicilerine göre bazı avantajları mevcuttur. Bunlardan bazıları, birleştirilmiş denetleyicilerde elde edilen gözlemci yapısı daha küçük olduğundan hesaplama daha hızlıdır. Kontrol poliçesinin fonksiyon olarak hesaplanabildiği optimum birleştirilmiş denetleyici her zaman mevcuttur [108]. Birleştirilmiş denetleyicilerde de kontrol kuralı için bir algoritma işletileceğinden gerçek zamanlı kontrolde kullanımı zor olacaktır. CtlPN'ler hakkında detaylı bilgi [76, 109]'da bulanabilir. Literatürde, CtlPN'ler kullanılarak elde edilen

denetleyiciler *gönderim denetleyicileri (mapping supervisor)* şeklindedir ve şu ana kadar CtlPN'ler kullanılarak elde edilen *derlenmiş denetleyicilerin (compiled supevisor)* tasarımı ve gerçekleştirilmesi konusunda mevcut çalışma bulunmamaktadır.

CtlPN ile modellenmiş AOS'lerin kontrolünde derlenmiş denetleyicilerin kullanılmasını sağlamak üzere gerçekleştirilen çalışmalar bu bölümde açıklanmaktadır. Derlenmiş denetleyicilerin hesaplanması için önerilen yöntemler [52]'de ortaya konan yaklaşımlar temel alınarak oluşturulmuştur.

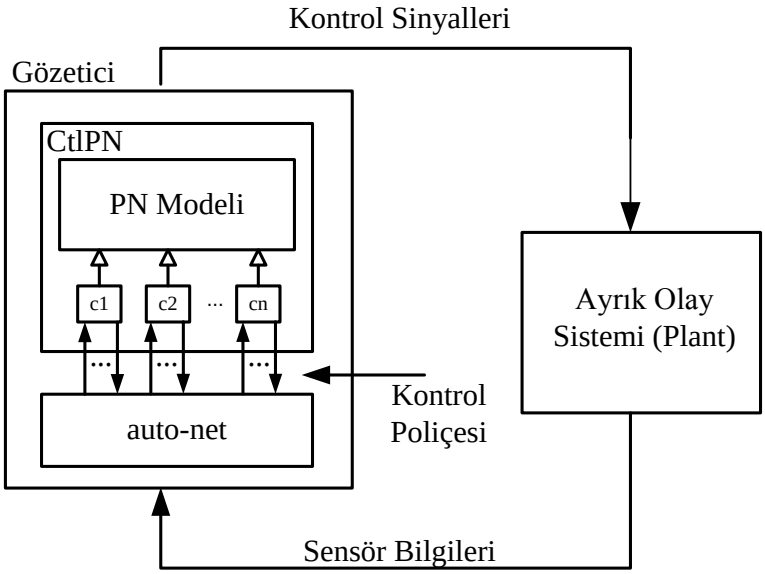
Tezin bu bölümünde ilk olarak, CtlPN ile modellenmiş AOS'lerinde kullanılmak üzere derlenmiş denetleyicilerin hesaplanabilmesi için önerilen yöntem açıklanmaktadır. Önerilen yöntemin uygulanabilirliği literatürde yaygın olarak kullanılan örnekler üzerinde gösterilmiştir. İkinci olarak, önerilen yöntemin modüler versiyonu yine literatürde yaygın olarak kullanılan bir örnek ile açıklanmıştır. Son olarak, önerilen yöntemlerin gerçek sistemlerin kontrolüne uygulanabilirliği deneysel endüstriyel üretim sistemi kullanılarak gösterilmiştir.

7.1 Kontrollü Petri Ağları için Derlenmiş Yekpare Denetleyicinin Sentezlenmesi

Petri ağlarının ve Gözetimli Kontrol Teorisinin avantajlarından aynı anda yaralanan melez bir yöntem [52]'de önerilmiştir. Bu çalışmada elde edilen gözetici yapılarında sisteme ait PN modeline elde edilen auto-net gözetici yasaklama okları kullanılarak bağlanmış ve kontrol sistemi elde edilmiştir.

CtlPN kullanılarak AOS'lerin kontrolü için önerilen yöntemin blok diyagramı Şekil 7.1'de görülmektedir. CtlPN'ler sıradan Petri ağlarında bulunan kontrol edilebilir geçişleri kontrol etmek üzere c1, c2, ..., cn harici kontrol mevkileri eklenmesi ile elde edilmektedirler. CtlPN modelindeki kontrol mevkilerini ve bu mevkileri kontrol edilebilir geçişlere bağlayan okları göz ardı ederek, [52]'de önerilen yöntem ile auto-net formunda bir gözetici ve kontrol poliçesi (kontrol verisi) elde edilmektedir. Elde edilen gözetici ve kontrol poliçesi kullanılarak gözeticinin bazı durumlarında kontrol mevkilerine jeton aktarılabilir yani ilgili geçiş yetkilendirilebilir veya bazı durumlarında kontrol mevkilerinden jeton alınabilir yani ilgili geçiş yasaklanabilir. Kontrol mevkilerine jeton aktarma ve alma işlemi PN modeli ile eş zamanlı çalıştığı kabul

edilen auto-net yapısında bulunan geçişler yardımıyla yapılmaktadır. Sonuç olarak CtlPN modeli sistemi istenilen doğru çalışmaya zorlamaktadır.

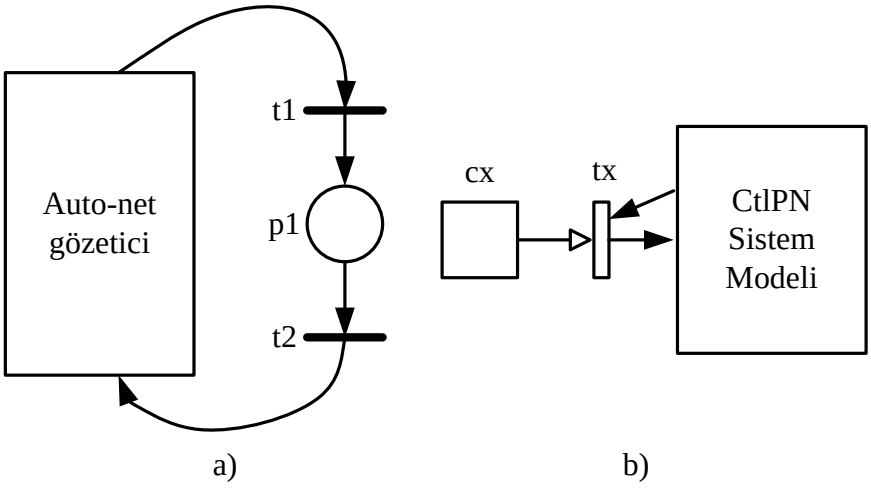


Şekil 7.1 Önerilen melez yöntemin blok şeması

CtlPN'ler için derlenmiş denetleyici sentezi iki aşamadan oluşmaktadır. İlk aşama istenilen kontrol spesifikasyonunu yerine getirecek auto-net ve kontrol poliçesinin (kontrol verisinin) Gözetimli Kontrol Teorisi ve [52]'de önerilen yöntem ile sentezlenmesidir. İkinci aşama ise, auto-net yapısının geçişlerinden CtlPN'nin kontrol mevkilerine bağlanacak olan sıradan giriş/çıkış oklarının (yani kontrol poliçesinin) hesaplanmasıdır. İlk aşamada, auto-net gözetici, CtlPN olarak modellenen sistemdeki kontrol mevkileri ve bu mevkileri kontrol edilebilir geçişlere bağlayan oklar göz ardı edilerek [52]'de önerilen yöntemle bulunabilir. İkinci aşama olan kontrol poliçesinin nasıl hesaplanabileceği ise aşağıda açıklanmaktadır.

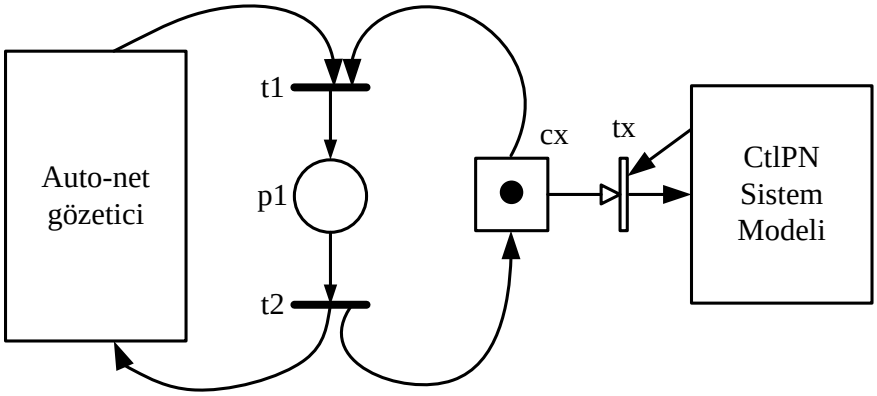
[52]'de önerilen metot ile auto-net yapısının yanında bir de kontrol verisi hesaplanır. Bu veri, auto-netin hangi mevkisinde sistem (plant) modelindeki hangi geçişin yasaklanacağını içermektedir. Örneğin, Şekil 7.2.(a)'da görülen auto-net yapısı, bir sistem için hesaplanan gözeticinin bir parçası olsun. Buna göre Şekil 7.2.(b)'de görülen modeldeki tx geçişinin kontrol edilmesi söz konusudur. Bu modelde CtlPN kullanıldığından tx kontrol edilebilir geçişine cx kontrol mevkisi bağlıdır. Kontrol edilebilir geçişin tetiklenebilmesi ancak bu kontrol mevkisinde jeton bulunması durumunda mümkün olmaktadır. [52]'de önerilen yöntem ile elde edilen örnek bir

kontrol verisinin ***p1 mevkisinde jeton varken tx geçişinin (olayının) yasaklanması*** olduğunu kabul edelim.



Şekil 7.2 a) Bir auto-net modelinin bir mevki iki geçişten oluşan parçası,
b) CtlPN modelindeki kontrol edilebilir bir geçiş

Bu örnek için CtlPN modelinde bulunan tx geçişini, auto-net modelinin p1 mevkisinde jeton varken yasaklayacak kontrol poliçesi Şekil 7.3'te görüldüğü gibi gerçekleştirilir. Şekil 7.3'ten görülebileceği gibi, t1 geçişinin tetiklenmesiyle cx kontrol mevkisindeki jeton alınmakta, t2 geçişi ile de bu mevkiye jeton yerleştirilmektedir. t1 geçişinin tetiklenmesi ile auto-net bu mevkiye ulaştığında cx mevkisindeki jeton alınarak tx geçişinin tetiklenmesi engellenmektedir.



Şekil 7.3 CtlPN modeli için tx geçişini auto-net gözeticinin
p1 mevkisinde jeton varken yasaklayacak kontrol poliçesinin gerçekleştirilmesi

Yukarıdaki örnekten de görülebileceği gibi, p1 mevkisinde jeton varken tx geçişini yasaklayabilmek için auto-netin p1 mevkisine ulaştığı geçişte yani t1 geçişinde cx

mevkisindeki jetonun alınması gerekmektedir. Eğer auto-net p1 mevkisinde değil ise tx geçişinin tetiklenebilir konumda olması gerekir. Bu işlem ise, auto-netin p1 geçişinden ayrıldığı geçişte yani t2 geçişinde cx mevkisine jeton aktarılarak yapılabilir. Bu işlemi formal olarak ifade edecek olursak, p1 mevkisi için Pre(p1) = {t1} ve Post(p1) = {t2} dir. Cx mevkisi için p1 mevkisinin Pre kümesi Post, Post kümesi ise Pre kümesi olmalıdır. Yani Pre(cx) = Post(p1) = {t2} ve Post(cx) = Pre(p1) = {t1}.

Şekil 7.2'de görülen auto-net modelindeki p1 mevkisinde başlangıçta jeton bulunmamaktadır. Yani başlangıçta auto-net bu mevkide değildir. Kontrol verisine göre auto-net bu mevkiye ulaştığında tx olayının yasaklanması gerekmektedir. Dolayısıyla auto-net bu mevkide olmadığından, başlangıçta CtlPN modelindeki tx olayı yetkilendirilmelidir. Yani, başlangıçta cx mevkisine jeton yüklenmelidir. Bu işlemi formal olarak ifade edecek olursak, μ(p1)=0 ise μ(cx)=1 olacaktır.

Yukarıda görülen örnek sadece bir mevki ve ona bağlı olan bir giriş ve bir çıkış geçişi için dikkate alınmıştır. Bu durum tüm ağ için genellenebilmektedir. Önerilen yöntemin genel formu aşağıdaki gibidir.

Sistemin kontrol edilmemiş davranışını modelleyen CtlPN yapısı aşağıda tanımlanmıştır.

$$CtlPN=(P_p,T_p, E_p,C, B,\mu_{p0}, \mu_{c0}) \quad (7.1)$$

Burada;

P_p: Sistem (plant) modelindeki sonlu mevkiler (durum mevkileri) kümesi,

T_p: Sistem (plant) modelindeki sonlu geçişler kümesi,

E_p: $E_p \subseteq (P_p \times T_p) \cup (T_p \times P_p)$ mevkiler arası geçişleri tanımlayan bağıntıdır (incidence relation), $P \cap T \neq \varnothing$ ve $P \cup T \neq \varnothing$,

C: Sonlu kontrol mevkileri kümesi,

B: $B \subseteq (C \times T_p)$ kontrol mevkilerini geçişlere bağlayan yönlenmiş oklar,

μ_{p0}: durum mevkilerinin başlangıç jeton işaretlemesi,

μ_{c0}: kontrol mevkilerinin başlangıç jeton işaretlemesidir.

CtlPN modelinde bulunan geçişler kontrol edilemeyen (uncontrolled) ve kontrol edilebilir (controlled) geçişler olmak üzere iki kümeye ayrılabilir. $T_p = T_{puc} \cup T_{pc}$

CtlPN ile modellenmiş bir AOS için derlenmiş denetleyici hesaplama yönteminde kullanılacak olan gözetici [52]'de önerilen metot yardımıyla hesaplanmaktadır. Hesaplama sonucunda elde edilen auto-net yapısı saf'tır (pure). Yani özdöngü (selfloop) içermez. Auto-net (AN) yapısı aşağıdaki gibi tanımlanabilir,

$$AN=(P_s,T_s,Pre_s,Post_s,\mu_{s0}) \tag{7.2}$$

Burada;

P_s: sonlu mevkiler kümesi,

T_s: sonlu geçişler kümesi,

Pre_s, P_s'den T_s'e yönlenmiş oklar kümesine karşılık gelen $P_s \times T_s \rightarrow \{0, 1\}$ giriş kodlamasıdır.

$Post_s$, T_s'den P_s'ye yönlenmiş oklar kümesine karşılık gelen $T_s \times P_s \rightarrow \{0, 1\}$ çıkış kodlamasıdır.

$\mu_{s0:}$ auto-netin başlangıç jeton işaretlemesidir.

Auto-net yapısı için $W_s : P_s \times T_s \rightarrow \{0, 1, -1\}$ olarak tanımlanan bağlantı matrisi (incidence matrix)

$$W_s(p,t) = \begin{cases} 1 & \text{ğer } (t,p) \in Pre_s \\ -1 & \text{ğer } (p,t) \in Post_s \\ 0 & \text{ğer di} \end{cases} \tag{7.3}$$

şeklinde tanımlanabilir. Auto-net, RW yönteminden elde edilen otomata formundaki yapının Petri ağı olarak ifade edilmiş şeklidir. Auto-net yapısında tüm mevkilerin jeton kapasitesi 1 dir ve ağda sadece bir adet jeton bulunur.

TCT yazılımı ile yapılan işlemler sonucunda, elde edilen gözeticinin hangi durumunda, sistemdeki hangi kontrol edilebilir olayın yasaklanacağını içeren kontrol verisi

hesaplanmaktadır. Bu kontrol verisini $CD: P_s x T_c \rightarrow \{0, 1\}$ olmak üzere aşağıdaki biçimde bir matris ile tanımlayabiliriz:

$$CD(p_s, t_c) = \begin{cases} 1 & \text{eğer } p_s \text{ mevkisinde } t_c \text{ geçişi yasaklanıyorsa} \\ 0 & \text{eğer durumlarda di} \end{cases} \quad (7.4)$$

Auto-netin hangi mevkisinde, CtlPN modelinin hangi kontrol edilebilir geçişinin yasaklanacağı, CD matrisi ile yukarıdaki gibi belirlenmektedir. CtlPN modelindeki bir kontrol edilebilir geçişin yasaklanabilmesi ancak onun kontrol mevkisinde jeton bulunmaması ile mümkün olacağı yukarıdaki örnekte incelendi. Eğer her bir kontrol edilebilir geçişe bir kontrol mevkisi bağlı ise $CD: P_s \times T_c \rightarrow \{0, 1\}$ tanımını $CD: P_s \times C \rightarrow \{0, 1\}$ şeklinde değiştirmek mümkündür. Örneğin, kontrol verisi auto-netin p1 mevkiside tx geçişinin yasaklanması ise CD(p1,cx)=1 olarak alınabilir.

Yukarıda açıklanan örnekten de görüldüğü gibi, auto-net mevkisinin Pre kümesi kontrol mevkisi için Post kümesi, Post kümesi ise Pre kümesi olmalıdır. Post ve Pre kümeleri bilgisi incidence matrisinde (W) toplu olarak bulunmaktadır. Bu matrisin (-1) ile çarpımı yani tüm elemanlarının işaret değişimi Pre ve Post'ların yerinin değişmesini sağlayacaktır. Bu elde edilen yeni matrisin, kontrol verisinin matris biçimi olan CD matrisi ile çarpımından bulunan yeni matris, kontrol mevkileri ile auto-netin geçişleri arasındaki kontrol poliçesini tanımlayan matris olacaktır. $CP: C \times T_s \rightarrow \{0, 1, -1\}$ şeklinde tanımlanmak üzere aşağıdaki gibi belirlenebilir.

$$CP = -CD^T W_s \quad (7.5)$$

CP kontrol poliçesinin yanı sıra bir de kontrol mevkileri için başlangıç jeton dağılımının hesaplanması gerekir. Bu hesaplama için auto-net yapısını tekrar göz önüne alalım. Bu yapı RW yöntemi sonucunda elde edilen otomatanın PN olarak ifade edilmiş biçimidir. Auto-net yapısı, otomatanın durumlarının kapasitesi bir olan birer mevki ve her bir geçişi için ise bir PN geçişi atanmak suretiyle elde edilmektedir. Bu yapıdaki ağda sadece bir adet jeton bulunabilir. RW metodu ile hesaplanan gözetici TCT hesaplamaları sonucunda bulunmakta ve her zaman ilk durumu başlangıç durumu

olmaktadır. Bu nedenle bu ilk durumu simgeleyen PN mevkisi her zaman başlangıçta jeton yüklenmekte geriye kalan mevkilere ise jeton yüklenmemektedir. Yukarıda anlatılan örnekten de görülebileceği gibi auto-net mevkisi başlangıçta boş ise kontrol mevkisi başlangıçta jeton yüklü olacak, dolu ise kontrol mevkisinde başlangıçta jeton bulunmayacaktır. Tüm kontrol mevkileri ve auto-netin başlangıç jeton dağılımını göz önüne alarak, kontrol mevkileri için başlangıç jeton işaretlemesi aşağıdaki şekilde tanımlanabilir.

$$\mu_{c0} = \overline{CD^T \mu_{s0}} \tag{7.6}$$

Bu denklemdeki $\overline{(...)}$ ifadesi ile hesaplama sonucunda bulunacak vektörün içeriğinin lojik değile (not) benzer şekilde "0" ise "1", "1" ise "0" olarak değiştirilmesi ifade edilmektedir. μ_{s0} yani auto-net'in başlangıç jeton dağılımını gösteren vektör denklem (7.7)'de görülmektedir. Bu vektör ile CD^T'nin çarpımı sonucu elde edilecek vektör CD^T'nin ilk sütunu olacaktır. Dolayısı ile kontrol mevkilerinin başlangıç jeton dağılımı için yapılacak hesaplamayı gösteren denklem (7.6) sadeleşerek (7.8)'deki gibi hesaplanabilir.

$$\mu_{s0=} \begin{bmatrix} 1 \\ 0 \\ . \\ . \\ 0 \end{bmatrix} \tag{7.7}$$

$$\mu_{c0} = \overline{(CD[1 \times n])^T} \tag{7.8}$$

Burada, n kontrol mevkilerinin sayısıdır.

CtlPN modelleri kullanılan AOS'lerde derlenmiş denetleyicilerin hesaplanması için önerilen yöntem aşağıdaki gibi önerilmektedir.

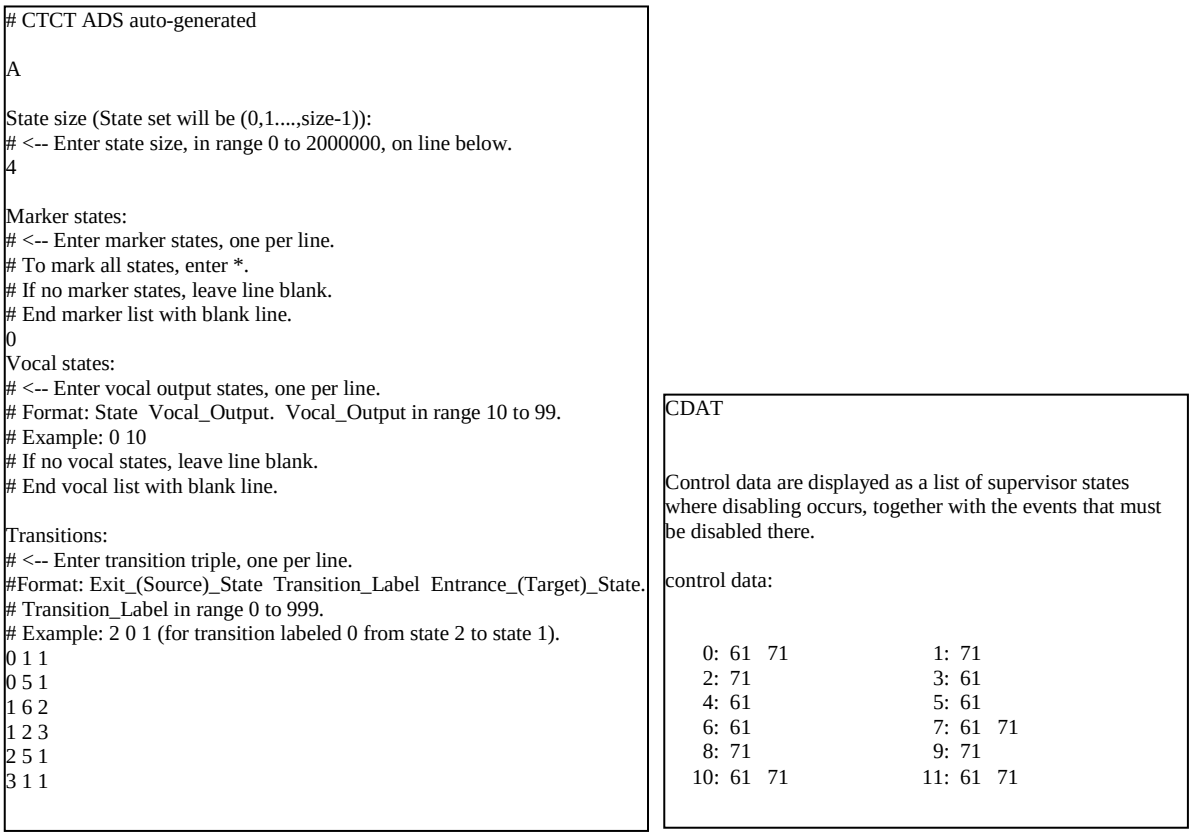
1. Sistemin kontrol edilmemiş davranışı CtlPN olarak modellenir.

2. CtlPN modelindeki kontrol mevkileri ve bu mevkileri geçişlere bağlayan oklar göz ardı edilerek geriye kalan PN modeli bir önceki kısımda ortaya konan yöntemler kullanılarak eşdeğer tampon (buffer) otomata modellerine dönüştürülür.
3. Spesifikasyonlar otomata olarak modellenir.
4. Gözetimli Kontrol Teorisi uygulanarak RW gözeticisi (SUPER) ve kontrol verisi (CDAT) elde edilir.
5. İndirgenmiş gözetici (SIMSUP) ve indirgenmiş kontrol verisi (SIMDAT) elde edilir.
6. Elde edilen gözetici (SIMSUP) auto-net yapısına dönüştürülür.
7. SIMSUP auto-neti için Incidence matrisi (W) (7.3) nolu eşitlik kullanılarak oluşturulur. SIMDAT kontrol verisi (7.4) nolu formül kullanılarak matris formuna dönüştürülür.
8. (7.5) ve (7.8) nolu formüller kullanılarak kontrol poliçesi (CP) ve kontrol mevkilerinin başlangıç jeton işaretlemesi (μ_{c0}) hesaplanır.
9. CP matrisi ile hesaplanan oklar auto-netin geçişlerinden kontrol mevkilerine bağlanarak kapalı çevrim derlenmiş denetleyici elde edilir.

RW metodunun uygulanması için gereken hesaplamalarda TCT yazılımı kullanılmaktadır. Bu yazılımdan elde edilen gözetici .ADS formatında bir dosyada ve kontrol verisi de .PDT formatında bir dosyada saklanmaktadır. Yukarıda anlatılan matris hesaplamalarının hızlandırılması ve kolaylaştırılması yazılacak bir bilgisayar programı ile mümkündür. Bu amaçla bir C kodu yazılmış ve bu kod CodeBlocks programı ile derlenerek bir "cp.exe" programı oluşturulmuştur. Bu programın kaynak kodu Ek-C'de verilmiştir. Bu program hakkında aşağıda bilgi mevcuttur.

Kontrol poliçesinin ve kontrol mevkilerinin başlangıç jeton dağılımını hesaplayan kodun temel algoritması aşağıda görülmektedir. Kod giriş olarak .ADS formatındaki gözetici ve .PDT formatındaki kontrol verisini almaktadır. Bu iki dosyanın formatları Şekil 7.4'te görülmektedir. Şekil 7.4.(a)'da örnek bir .ADS dosyası için dosya formatı görülmektedir. Bu dosyanın son kısmında gözeticinin geçişleri alt alta [*kaynak durumu olay etiketi hedef durumu*] formatında tanımlanmaktadır. [1 6 2] gösterimi, gözeticinin "1" etiketli durumundan "6" etiketli olayın gerçekleşmesi ile "2" etiketli duruma geçeceğini tanımlamaktadır. Şekil 7.4.(b)'de ise örnek bir .PDT dosya formatı

görülmektedir. Bu dosya kontrol verisini tanımlamaktadır. Dosyanın alt kısmında görülen “10: 61 71” gösterimi gözeticinin “10” etiketli durumunda “61” ve “71” etiketli olayların yasaklanacağı anlamına gelmektedir. Kontrol verisinin dosya yapısı incelendiğinde bir satırda bulunan iki durumun her birisi için maksimim 6 adet olay etiketi yazılabilmekte olduğu görülmektedir.



a) b)

Şekil 7.4 a) Örnek bir .ADS dosya formatı, b) Örnek bir .PDT dosya formatı

TCT yazılımında bir gözetici *Supcon(.)* komutu kullanılarak elde edilir. Örneğin *SUPER = Supcon(PLANT,SPEC)* şeklinde yazılan TCT komutunda PLANT sistem modelinin, SPEC spesifikasyon modelinin otomata ifadesi olmak üzere SUPER, elde edilen RW gözeticisidir. Benzer şekilde TCT yazılımında, kontrol verisi *Condat(.)* komutu kullanılarak elde edilir. Örneğin *CDAT = Condat(PLANT,SUPER)* ifadesinde CDAT isimli kontrol verisini elde etmek için ilgili sistem modeli PLANT ve RW gözeticisi SUPER kullanılmalıdır. Bu çalışmada önerilen yöntemde sistem modelinde bulunan kontrol edilebilir geçişlerin her birine bir adet kontrol mevkisinin yetkileme oku ile bağlı olduğu varsayılmaktadır. Bu nedenle yazılan kod ile hesaplamalarda

sistem modelinde bulunan kontrol edilebilir geçişlerin etiketleri, kontrol mevkilerinin isimleri olarak kullanılmaktadır. Sisteme ait otomata modeli ve TCT yazılımda bulunan *Allevents(.)* komutu kullanılarak sistem modelinde kullanılan tüm olay etiketleri (yani CtlPN modelinde bulunan tüm geçiş etiketleri) bir duruma ("0" etiketli durum) öz döngü olacak şekilde yeni bir otomata elde edilebilir. Elde edilen bu otomataya ait .ADS dosyası kullanılarak, sistem modelinde bulunan olay etiketleri, olay sayısı ve kontrol edilebilir olay etiketleri elde edilebilir. Aşağıda görülen algoritmalar kullanılarak yazılan C koduyla, ilgili matris hesaplamaları çok kolaylaştırılmıştır. Bu kod exe program haline getirilerek "cp.exe" olarak adlandırılmıştır. Hesap sonuçları ekrana ve ayrıca log.txt adlı bir log dosyasına yazdırılmıştır.

Kontrol Poliçesi Hesaplama Algoritması:

Girdi: *.ADS formatında gözetici ve .PDT formatında kontrol verisi.*

1. *[tanımlama]: gerekli tanımlamalar alt algoritmalarda yapılmıştır.*
2. *ALLevents_okuma algoritması ile kontrol edilebilir olaylar listesi elde edilir.*
3. *ADS_okuma algoritması ile gözeticinin geçişleri okunarak Ws matrisi hesaplanır.*
4. *PDT_okuma algoritması ile kontrol verisi okunarak CD matrisi hesaplanır.*
5. *CP_hesaplama algoritması ile (5.5) nolu denklem kullanılarak Kontrol poliçesi; (5.8) nolu denklem kullanılarak ta kontrol mevkilerinin başlangıç jeton sayısı hesaplanır.*

Çıktı: *Matris formda kontrol poliçesi ve informal formda kontrol mevkilerinin başlangıç jeton dağılımı.*

Kontrol Poliçesi Hesaplama Algoritmasının sonu.

ALLevents_okuma Algoritması:

Girdi: *.ADS formatında, tüm olayların listesi; Bu format [kaynak durumu olay etiketi hedef durumu] biçiminde oluşturulmuştur. Bu formatın her bir satırındaki olay etiketi sistem modelinde bulunan bir geçiş etiketidir. Bu formattaki her bir satır geçiş olarak isimlendirilmektedir. Geçişlerin tamamı göz önünde bulundurulduğunda [n_evnt x 3]'lük bir dizi oluşmaktadır. Bu dizinin ilk ve son sütunu "0" dır. Yani sistemde bulunan tüm olaylar "0" etiketli duruma özdöngülüdür (Selfloop).*

1. *[tanımlama]: Crr_e: güncel olay bilgisi, EvlstTrans[n_evnt x 3]: geçişler dizisi, Con_ev: kontrol edilebilir olaylar listesi, n_cev: kontrol edilebilir olayların sayısı, Ev_list*
2. *[başlangıç]: i=0, j=0;*
3. *Crr_e=Ev_list[i,2]*
4. *Eğer Crr_ev tek ise Con_ev[j]= Crr_e*
   j=j+1
   değilse
   i=i+1
5. *eğer (i<n_trans) 3.adıma geri dön*
6. *n_cev=j-1;*

Çıktı: *Kontrol edilebilir olay sayısı n_cev, kontrol edilebilir olaylar listesi Con_ev,*

ALLevents_okuma Algoritmasının sonu.

ADS_okuma Algoritması:

Girdi: *.ADS formatında RW gözeticisi, [kaynak durumu olay etiketi hedef durumu] biçiminde oluşturulmuştur. Bu formatın her bir satırı bir geçiş olarak isimlendirilmektedir. Geçişlerin tamamı göz önünde bulundurulduğunda [n_trans x 3]'lük bir dizi oluşmaktadır.*

1. *[tanımlama]: Ws: gözeticinin incidence matrisi, n_trans: geçişlerin toplam sayısı (her bir geçiş bir satırla ifade edildiğinden satır sayısı), Trans[n_trans x 3]: geçişler dizisi Crr_s_state: güncel kaynak durumunu belirten değişken, Crr_t_state: güncel hedef durumunu belirten değişken, Crr_e: güncel olay bilgisi, n_sts: durumların sayısı, Ev_list: olaylar listesi.*
2. *[başlangıç]: i=0;*
3. *n_sts=7. satır. (.ADS dosyasının 7. satırında auto_net'in kaç durumdan oluştuğu bilgisi mevcuttur.*
4. *Ws[n_sts x n_trans]=0*
5. *Crr_s_state=Trans[i,1], Crr_t_state= Trans[i,3], Crr_e= Trans[i,2] geçişler dizisinden bir geçiş al ve ilgili bileşenlere ayır.*
6. *Ws[Crr_s_state,i]= -1*
7. *Ws[Crr_t_state,i]=1*
8. *i=i+1*
9. *eğer (i<n_trans) ise 5. adıma geri dön.*

Çıktı: *gözeticinin incidence matrisi Ws, toplam durum sayısı n_sts.*

ADS_okuma Algoritmasının sonu.

PDT_okuma Algoritması:

Girdi: *.PDT formatında RW gözeticisi için kontrol verisi. PDT formatındaki dosyadaki bilgiler [states]:[events][states]:[events] biçimindedir. Burada, [states] bir durum etiketi [events] ise maksimum 6 olay etiketi içermektedir. Bu dört parçalı bilgi control[n_csts x 7] halinde bir diziye dönüştürülmüştür. Burada n_csts PDT dosyasında bulunan durumların sayısıdır. Bu dizinin control[n_csts x 1] sütunu mevki bilgisini geriye kalanlar ise yasaklanacak olay etiketlerini içermektedir. n_sts, toplam durum sayısı ADS_oku algoritmasından, olaylar listesi Ev_list, kontrol edilebilir olaylar listesi Con_ev, kontrol edilebilir olayların sayısı n_cev, ALLevents_okuma algoritmasından elde edilir.*

1. *[tanımlama]: C_ev: güncel durumu bildiren değişken, Con_ev: kontrol edilebilir olaylar listesi, C_state: güncel durumu bildiren değişken, C_events: güncel olayı bildiren değişken,*
2. *[başlangıç]: u=0, v=0, i=0, j=0, k=0,*
3. *CD[n_sts x n_cev]=0*
4. *C_ev= Ev_list[u]*
5. *C_state=control [i, 1]*
6. *C_events= control [i, j]*
7. *Eğer C_events=C_ev[k]*
   CD[C_state][k]=1
   k=k+1
8. *Eğer k<n_cev ise 7. adıma geri dön*

k=0, j=j+1

9. *Eğer j<7 ise 6. adıma geri dön*
   j=0, i=i+1
10. *Eğer i<n_csts ise 5. adıma geri dön*

Çıktı: *matris formda (CD) kontrol verisi.*

PDT_okuma Algoritmasının sonu.

CP_hesaplama Algoritması:

Girdi: *Incidence matrisi Ws[n_sts][n_trans] ve kontrol verisi matrisi CD[n_sts][n_cev].*

1. *[tanımlama]: CP: kontrol poliçesinin incidence matrisi, ms: gözeticinin başlangıç jeton işaretlemesi, mc: kontrol mevkilerinin başlangıç jeton işaretlemesi*
2. *[başlangıç]: i=0, j=0; k=0, u=0, v=0, x=0, y=0, w=0, CP[n_cev][n_trans]=0, mc[n_cev]=0, ms[n_sts]=0*
3. *CDT[u][v]=CD[v][u]*
4. *v=v+1, eğer (v<n_sts) ise 3. adıma geri dön.*
5. *u=u+1, eğer (v<n_cev) ise 3. adıma geri dön.*
6. *CP[i][j]=CP[i][j]+CD[i][k]*Ws[k][j]*
7. *k=k+1, eğer (k<n_sts) ise 6. adıma geri dön.*
8. *j=j+1, eğer (j<n_trans) ise 6. adıma geri dön.*
9. *i=i+1, eğer (i<n_cev) ise 6. adıma geri dön.*
10. *CPs[x][y]=(-1)*CP[x][y]*
11. *x=x+1, eğer (x<n_cev) ise 10. adıma geri dön.*
12. *y=y+1, eğer (y<n_trans) ise 10. adıma geri dön.*
13. *Eğer CDT[w,1]=1 ise mc[w]=0*
14. *Eğer CDT[w,1]=0 ise mc[w]=1*
15. *w=w+1, eğer (w<n_cev) ise 13. adıma geri dön*

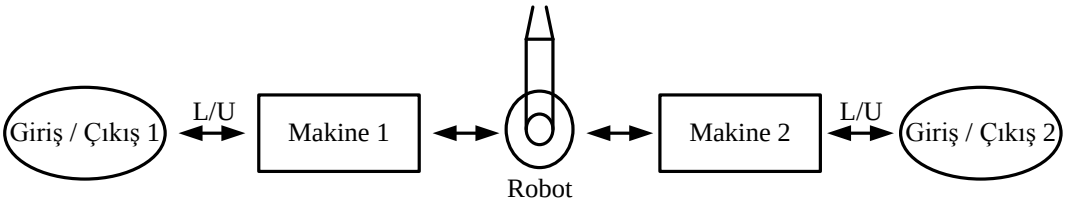
Çıktı: *matris formda kontrol poliçesi CP ve kontrol mevkilerin başlangıç jeton işaretlemesini gösteren mc vektörü*

CP_hesaplama Algoritmasının sonu.

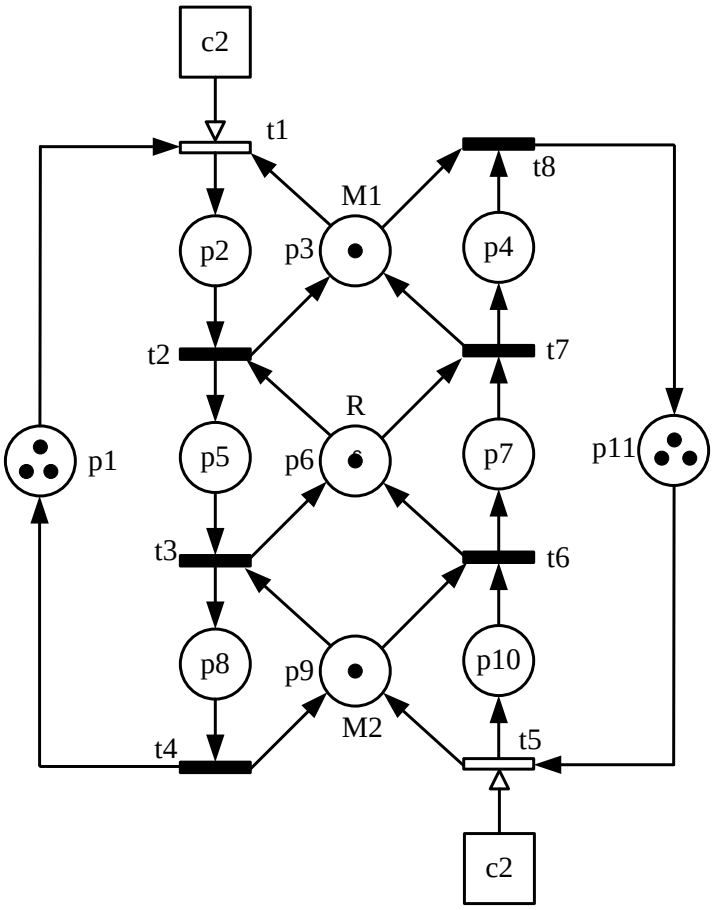
Aşağıdaki örnekle önerilen yöntemin ve yazılan programın kullanımı gösterilmektedir.

İncelenecek olan sistem Şekil 7.5'de görülen esnek üretim sistemidir (Flexible Manufacturing System - FMS). Bu sistemde iki makine ve bir robot bulunmaktadır. Makineler (M1 ve M2) aynı anda sadece bir parçayı işleyebilir ve benzer şekilde robot (R) sadece bir parçayı bir makineden diğerine aktarabilir. P1 (▉) ve P2 (▒) adında iki farklı parça bu sistem tarafından işlenmektedir. Parçalar makinelere I/O1 ve I/O2 giriş/çıkış tamponları tarafından yüklenip çıkarılmaktadır. Buna göre bu iki farklı parçanın makineler ve robot tarafından işlenmesini ifade eden iki farklı üretim akışı söz konusudur: P1 için üretim akışı P1: M1 $\Rightarrow$ R $\Rightarrow$ M2 şeklinde ve benzer şekilde P2 için üretim akışı P2: M2 $\Rightarrow$ R $\Rightarrow$ M1 şeklindedir. Bu sistemde spesifikasyon olarak herhangi

bir tıkanma (deadlock) olmaksızın istenilen iki farklı üretim akışının sağlanması istenmektedir. Şekil 7.5'te görülen FMS sisteminin CtlPN modeli Şekil 7.6'da görülmektedir.

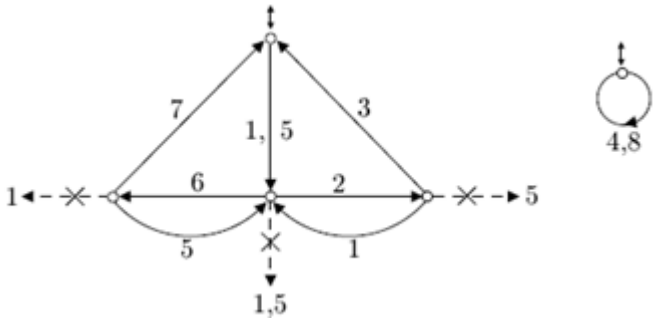


Şekil 7.5 Örnek bir esnek üretim sistemi (Flexible Manufacturing System - FMS)



Şekil 7.6 FMS'in CtlPN modeli

Bu problemin önerilen yöntem ile çözümü için ilk olarak CtlPN modelindeki kontrol mevkileri göz ardı edildikten sonra [52]'de önerilen yol takip edilerek bu sistem için auto-net formunda gözetici sentezlenmelidir. Sentezlenecek gözetici otomata formunda [48]'de hesaplanmıştır. Bu model Şekil 7.7'de görülmektedir. Hesaplanan bu gözeticinin auto-net biçimi Şekil 7.8'de verilmiştir.



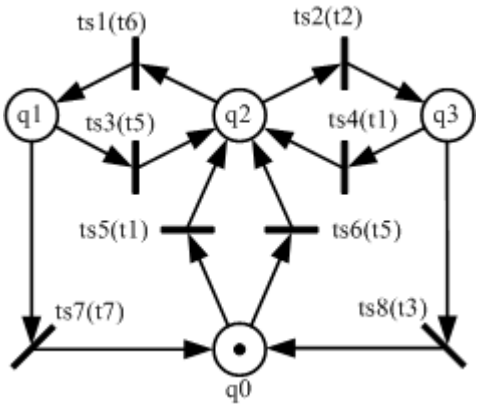
Şekil 7.7 Sistemin kontrolü için elde edilen indirgenmiş gözetici SIMSUP [48]

Bu problem için kullanılan CtlPN modelindeki geçişleri simgeleyen TCT olay etiketleri aşağıda görülmektedir. Kontrol edilebilir geçişler için tek sayılar, kontrol edilemeyen geçişler için ise çift sayılar etiket olarak kullanılmaktadır.

CtlPN modelindeki geçiş:	t1	t2	t3	t4	t5	t6	t7	t8
TCT olay etiketi:	1	2	30	40	5	6	70	8

Şekil 7.8'de görülen auto-net geçişleri yeniden adlandırılmıştır. Bu auto-nette bulunan her bir geçiş FMS için önerilen CtlPN modelindeki aynı olayı karakterize eden geçiş ile senkron çalışmaktadır. Şekil 7.8'de görülen auto-net modelindeki geçişlerin senkron olarak çalıştığı CtlPN geçiş isimleri aşağıda görülmektedir.

Auto-net modelindeki geçiş:	ts1	ts2	ts3	ts4	ts5	ts6	ts7	ts8
CtlPN modelindeki geçiş:	t6	t2	t5	t1	t1	t5	t7	t3



Şekil 7.8 Şekil 7.7'de görülen indirgenmiş gözeticinin auto-net karşılığı

Şekil 7.8'de görülen auto-net için Ws Incidence matrisi aşağıdaki gibidir.

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

Şekil 7.8'de görülen auto-net için kontrol verisi aşağıda görülmektedir:

1: 1 2: 1 5 3: 5

Bu verinin matris biçimine getirilmiş şekli aşağıda görülmektedir.

$$CD = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix}$$

Denklem (7.5) yardımıyla kontrol poliçesi aşağıdaki gibi hesaplanır:

$$CP = (-1)\begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}\begin{bmatrix} 0 & 0 & 0 & 0 & -1 & -1 & 1 & 1 \\ 1 & 0 & -1 & 0 & 0 & 0 & -1 & 0 \\ -1 & -1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & -1 & -1 & -1 & 1 & 0 \\ 1 & 0 & -1 & 0 & -1 & -1 & 0 & 1 \end{bmatrix}$$

Denklem (7.6) yardımıyla başlangıç jeton sayısı aşağıdaki gibi hesaplanır:

$$\mu_{c0} = \overline{(CD[1\times n])^T} = \overline{\left(\begin{bmatrix} 0 & 0 \end{bmatrix}\right)^T} = \overline{\begin{bmatrix} 0 \\ 0 \end{bmatrix}} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

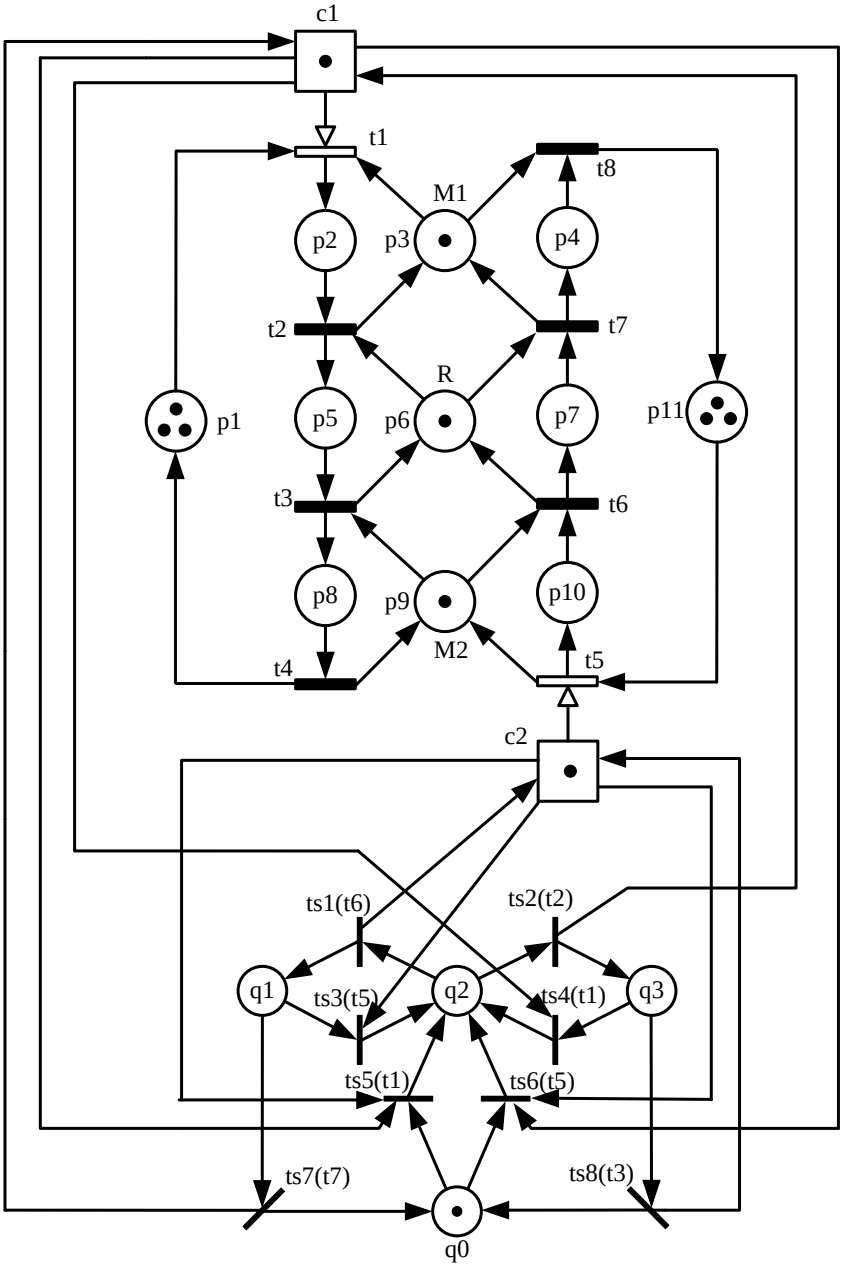
Bu hesaplamadan da görüleceği gibi başlangıçta kontrol mevkilerinin her ikisine de jeton konulmalıdır.

Yukarıda görülen CP matrisi kontrol mevkileri ile auto-netin geçişleri arasındaki bağıntıyı tanımlamaktadır. Bu matrise göre, c1 ve c2 kontrol mevkilerinin Pre ve Post geçişleri Çizelge 7.1'de görülmektedir.

Çizelge 7.1 Örnek FMS için hesaplanan kontrol poliçesi

Kontrol mevkisi	Pre	Post	Başlangıç Jetonu
c1	ts2, ts7	ts4, ts5, ts6	1
c2	ts1, ts8	ts3, ts5, ts6	1

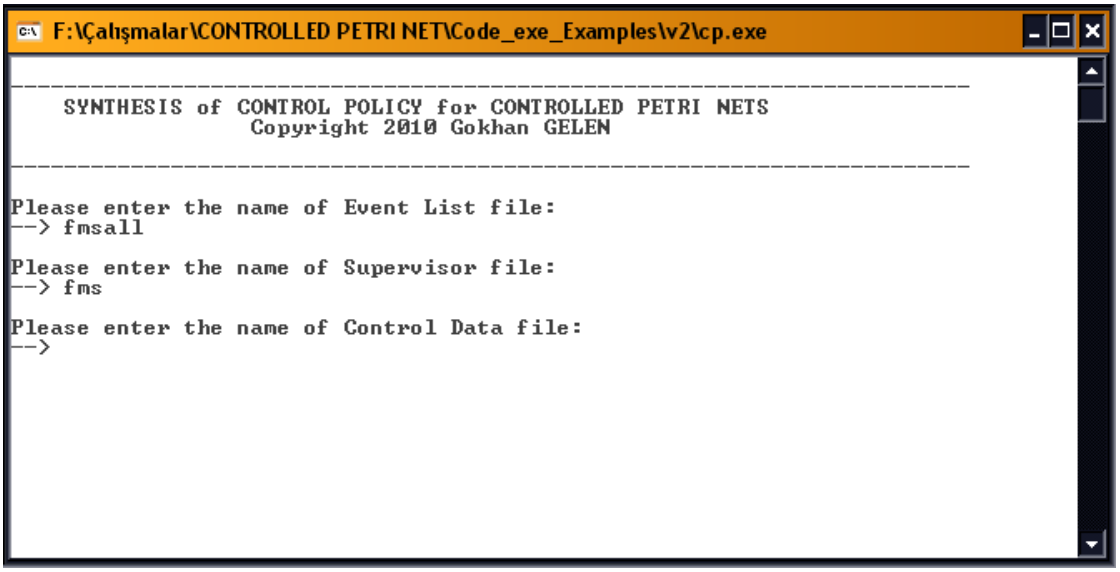
Bu verilere göre elde edilen derlenmiş denetleyici Şekil 7.9'da görüldüğü gibi CtlPN modeline bağlanarak, kapalı çevrim kontrol sistemi elde edilmiş olur.



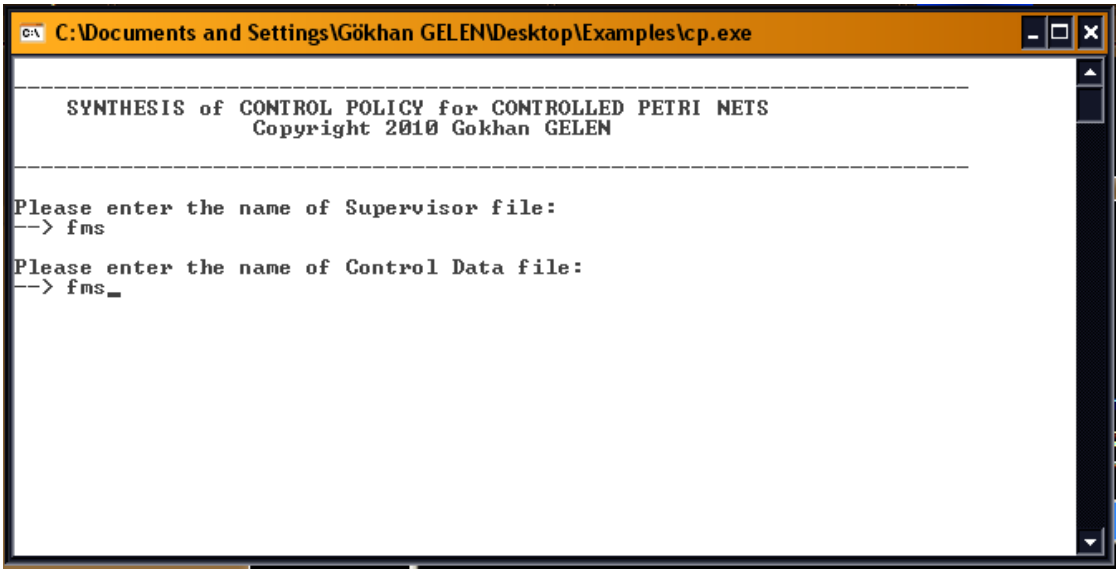
Şekil 7.9 FMS için kapalı çevrim kontrol sistemi

Yazılan kod ile FMS için derlenmiş denetleyici aşağıda görüldüğü gibi hesaplanmıştır. İlk olarak yukarıda açıklanan algoritmalar kullanılarak yazılan C kodundan elde edilen

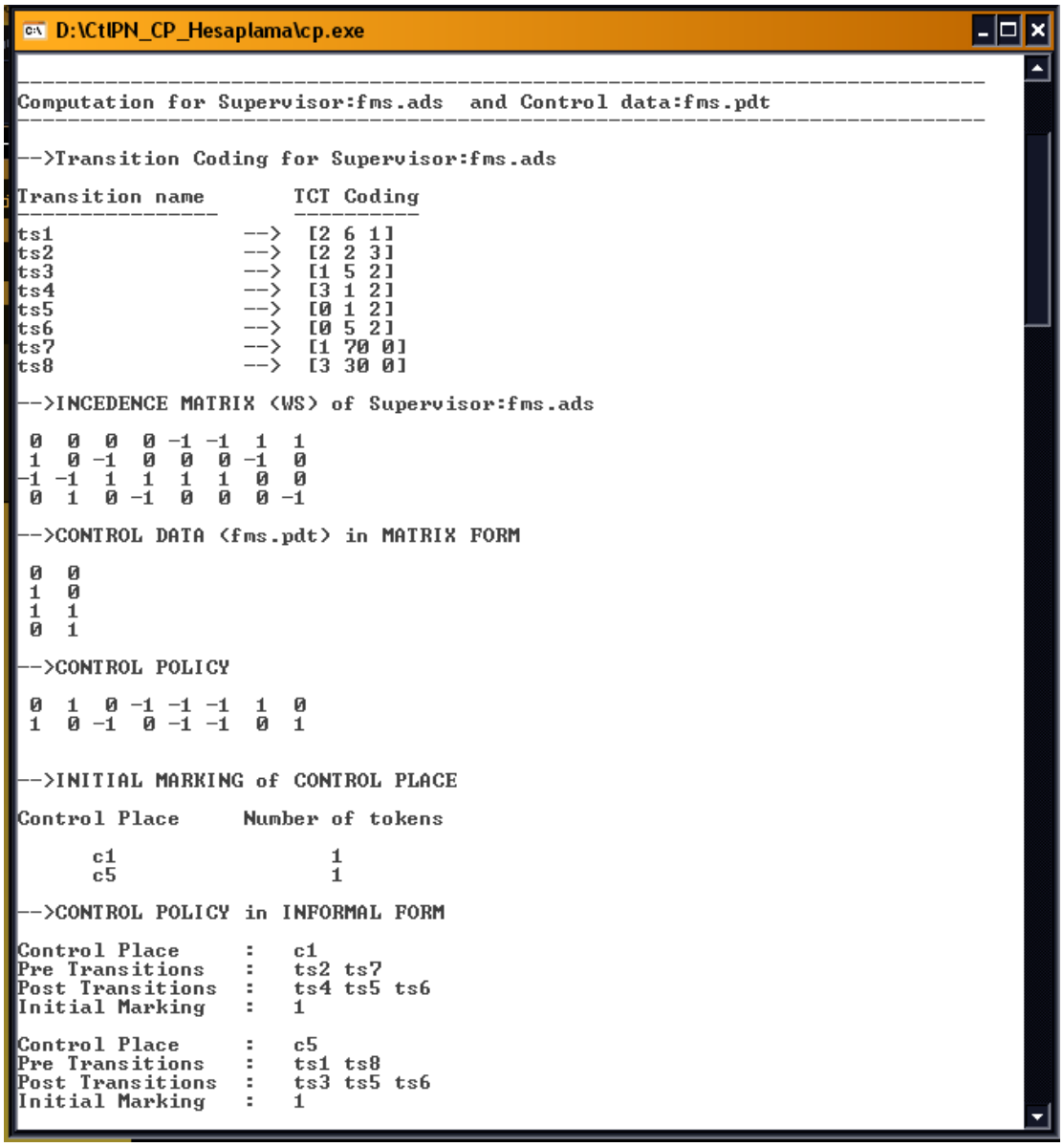
“cp.exe” programı çalıştırılır. Bu program ilk olarak auto-net modelinde bulunan tüm olay etiketlerini içeren ADS dosyasının ismini istemektedir. Bu girdinin ardından programa auto-nete ait ADS dosyasının ismi girilir. Bu girdi işlemlerine ait program ekran çıktısı Şekil 7.10’da görülmektedir. Kontrol verisine ait PDT dosyasının isminin programa girilmesinin ardından yazılan program kontrol poliçesini ve başlangıç jeton dağılımını aşağıda görüldüğü gibi hesaplayarak hem ekrana hem de log dosyasına kaydeder. Şekil 7.11’de PDT dosya isminin programa girilmesi görülmektedir. Programın ürettiği sonuçların ekran çıktısı ise Şekil 7.12’de görülmektedir.



Şekil 7.10 Programa ADS dosya isminin girilmesi



Şekil 7.11 Programa PDT dosya isminin girilmesi



Şekil 7.12 Programın ürettiği ekran çıktılarının görüntüsü

Program her hesaplama sonunda bulunan sonuçları ayrıca, ekrana yazılan formatın benzer şekliyle bir log dosyasına yazarak kayıt altına almaktadır. Yapılacak yeni hesaplamalar bu log dosyasının alt kısmına yazılır. Önceki hesaplamalara ait kayıtlar silinmeden bu dosyada saklanmaktadır.

7.2 Örnekler

7.2.1 Küçük imalat sisteminin CtlPN ile melez kontrolü

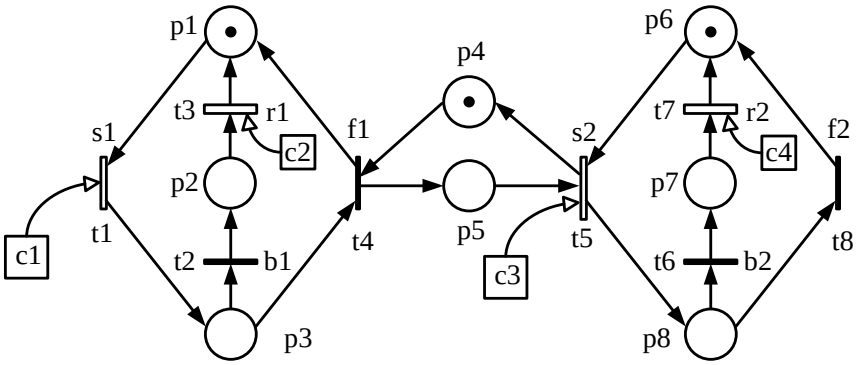
Bu örnek AOS literatüründe kullanılan iki makine ve bir tampondan oluşan imalat sistemidir [46, 71]. Bu sistemde Makine 1 ve Makine 2, kapasitesi 1(bir) olan tampon ile Şekil 7.13 görüldüğü gibi birbirine bağlanmıştır. Sistemdeki her bir makinenin; bir işlem yapmadığı durum *BOŞTA*, ilgili parçayı işlediği durum *ÇALIŞIYOR* ve arıza oluştuğu durum *ARIZALI* olmak üzere üç durumu mevcuttur. Tamponun ise dolu olduğunu belirten *DOLU* ve boş olduğunu belirten *BOŞ* olmak üzere iki durumu mevcuttur. Başlangıçta tamponun içerisinde bir parça olmadığı ve makinelerin BOŞTA olduğu kabul edilmektedir.

Şekil 7.13 Örnek bir imalat sistemi

Bu imalat sisteminde bir ürünün üretimi şu şekilde belirlenmektedir. İşlenecek parça Makine 1 tarafından girişten alınmaktadır. Bu örnekte girişte sonsuz kapasiteli bir tampon olduğu ve giriş parçalarının buradan alındığı kabul edilmektedir. Makine 1 girişten aldığı parça üzerindeki işini (örneğin parçayı delmesi, kesmesi, vb) bitirmesi ile parça tampona geçmektedir. Makine çalışırken arızaya geçmesi durumunda tamir edilerek tekrar çalışma durumuna dönmektedir. Tampon'da bulunan parça Makine 2 tarafından alınarak üzerinde ilgili işlemler gerçekleştirilir ve çıktı ürünü oluşturulur. Makine 1'e benzer olarak Makine 2 de çalışma konumundayken arızaya geçerse tamir edilerek tekrar çalışır duruma getirilmektedir. Bu sistemin kontrolü için bir önceki kısımda anlatılan adımlarla denetleyici aşağıdaki gibi hesaplanır:

1. Adım: Sistemin kontrol edilmemiş davranışının CtlPN olarak modellenmesi

Sistemin CtlPN modeli Şekil 7.14'te görülmektedir. Bu modelde görülen geçişlerin simgelediği olaylar, bu olayların anlamları ve bu olaylar için TCT programında kullanılan etiketler Çizelge 7.2'de görülmektedir.



Şekil 7.14 Örnek imalat sisteminin CtlPN modeli

Çizelge 7.2 Örnek imalat sisteminin CtlPN modelindeki olay etiketleri

Olay	Anlamı	TCT Etiketi
s1	M1'in çalışmaya başlaması	1
b1	M1'in arızalanması	2
r1	M1'in tamir edilmesi	3
f1	M1'in işini bitirmesi	4
s2	M2'in çalışmaya başlaması	5
b2	M2'in arızalanması	6
r2	M2'in tamir edilmesi	7
f2	M2'in işini bitirmesi	8

Şekil 7.14'ten görülebileceği gibi kontrol edilebilir geçişler ancak kontrol mevkilerinde jeton bulunması durumunda tetiklenebilmektedirler. Makinelerin çalışmalarını tekrar hatırlayacak olursak; başlangıçta makinelerin boşta olduğu varsayılmaktadır. Örneğin, Şekil 7.14'te görüldüğü gibi başlangıç jeton dağılımında sadece p1, p4 ve p6 mevkilerinde jeton bulunmaktadır. Bu, başlangıçta makinelerin boşta ve tamponun boş olduğunu göstermektedir.

Makine 1 için s1 olayının gerçekleşmesi (makinenin çalışmaya başlaması) ve c1 kontrol mevkisinde jeton bulunması ile bu geçiş tetiklenir ve jeton p3 (makine çalışıyor) mevkisine geçer. Bu durumda t2 ve t4 geçişleri açıktır. Makine 1'in arızalanması (b1 olayının gerçekleşmesi) ile t2 geçişi tetiklenir ve jeton p2 mevkisine geçer. Bu, makinenin arızada olması demektir. Bu durumda, sadece t3 geçişi açıktır ve bu geçişin tetiklenebilmesi için makinenin tamir edilmesi yani r1 olayının gerçekleşmesi ve c2 kontrol mevkisinde jeton bulunması gerekir. Bu olayın gerçekleşerek t3 geçişinin tetiklenmesi ile jeton tekrar p2 mevkisine dönmektedir. Bu da, makinenin tekrar

çalışıyor olması anlamına gelmektedir. p3 mevkisinde jeton varken ve makinenin işini bitirmesi ile t4 geçişi tetiklenmektedir. Bu durumda p3 ve p4 mevkilerinden jeton alınarak, p1 ve p5 mevkilerine aktarılmaktadır. Yani makine 1 boşta konumuna geri dönmekte ve tampon dolmaktadır. Makine 2'nin çalışması Makine 1 ile aynıdır.

2. Adım: CtlPN modelindeki kontrol mevkileri ve bu mevkileri geçişlere bağlayan oklar göz ardı edilerek geriye kalan PN'in eşdeğer tampon (buffer) modellerine dönüştürülmesi

Şekil 7.14'te görülen CtlPN modelinde bulunan kontrol mevkileri (c1, c2, c3, c4) ve bu mevkileri geçişlere bağlayan yetkileme okları göz ardı edilerek, bu ağdaki tüm mevkilerin otomata karşılıklarına Şekil 7.15'te görüldüğü gibi dönüştürülür.

3. Adım: Spesifikasyonların otomata olarak modellenmesi

Bu imalat sisteminde spesifikasyonlar aşağıdaki gibidir.

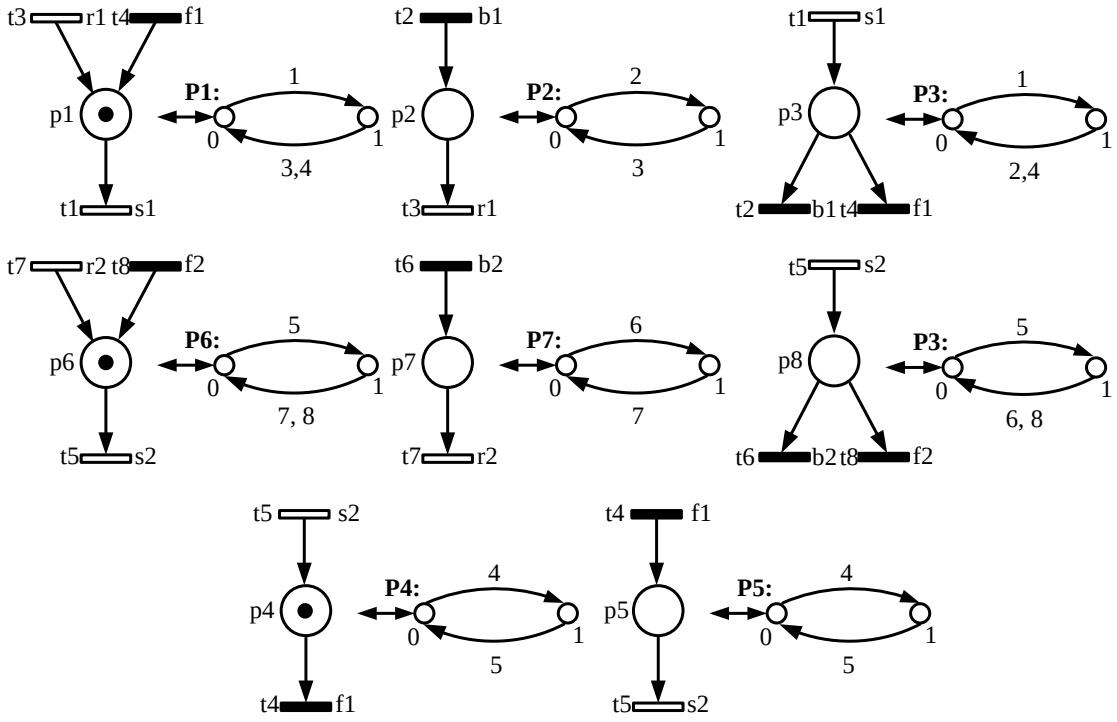
1. Sistemde bulunan tampon ne taşacak ne de tamamen boşalacak. Yani tampon dolu iken makine 1, tampon boş iken de makine 2 çalışmayacaktır.
2. Her iki makinenin bozulması durumunda ilk önce Makine 2 tamir edilecektir.

Bu spesifikasyonlardan ilkine ait otomata modeli (SP1) Şekil 7.16'da görülmektedir. Bu modele göre her zaman *f1* (Makine1'in yapacağı işlemi bitirmesi) *s2*'den (Makine 2'nin tampondan parça almasından) önce oluşmaktadır. Ayrıca bu modele göre, Makine 1 çalışmasını bir kez bitirdikten (*f1*) sonra Makine 1'in çalışmasını bir kez daha bitirebilmesi için öncelikle Makine 2'nin çalışmaya başlaması (*s2*) gerekmektedir. İkinci spesifikasyona ait otomata modeli (SP2) de Şekil 7.16'da görülmektedir. Bu modele göre Makine 1'in tamir edilebilmesi için (r1) Makine 2'nin arızalanmamış olması (b2) gerekir. Eğer, Makine 2 arızalanmışsa (b2), bu durumda Makine 1'in tamir edilebilmesi (r1) için öncelikli olarak Makine 2'in tamir edilmesi (r2) gerekir.

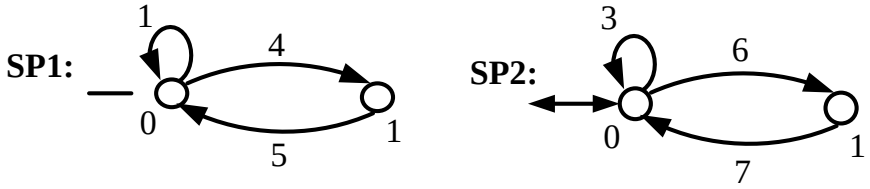
4.Adım: Gözetimli Kontrol Teorisi uygulanarak RW gözeticisi (SUPER) ve kontrol verisi (CDAT) elde edilmesi

RW metodu ile bu örnek için gözetici aşağıda görülen TCT işlemleri yardımıyla hesaplanmıştır. Hesaplamalar sonucunda bulunan gözeticinin otomata modeli ve kontrol verisi aşağıda görülmektedir.

P1 = Create(P1,[mark 0],[tran [0,1,1],[1,3,0],[1,4,0]]) (2,3)
P2 = Create(P2,[mark 0],[tran [0,2,1],[1,3,0]]) (2,2)
P3 = Create(P3,[mark 0],[tran [0,1,1],[1,2,0],[1,4,0]]) (2,3)
P4 = Create(P4,[mark 0],[tran [0,4,1],[1,5,0]]) (2,2)
P5 = Create(P5,[mark 0],[tran [0,4,1],[1,5,0]]) (2,2)
P6 = Create(P6,[mark 0],[tran [0,5,1],[1,7,0],[1,8,0]]) (2,3)
P7 = Create(P7,[mark 0],[tran [0,6,1],[1,7,0]]) (2,2)
P8 = Create(P8,[mark 0],[tran [0,5,1],[1,6,0],[1,8,0]]) (2,3)
SP1 = Create(SP1,[mark 0],[tran [0,1,0],[0,4,1],[1,5,0]]) (2,3)
SP2 = Create(SP2,[mark 0],[tran [0,3,0],[0,6,1],[1,7,0]]) (2,3)



Şekil 7.15 CtlPN modelindeki mevkilerin otomata karşılıkları



Şekil 7.16 Spesifikasyonlara ait otomata modelleri

M1 = Sync(P1,P2) (4,7) Blocked_events = None
M1 = Sync(M1,P3) (3,4) Blocked_events = None
BUF = Sync(P4,P5) (2,2) Blocked_events = None
M2 = Sync(P6,P7) (4,7) Blocked_events = None
M2 = Sync(M2,P8) (3,4) Blocked_events = None
PLANT = Sync(M1,BUF) (6,10) Blocked_events = None
PLANT = Sync(PLANT,M2) (18,42) Blocked_events = None

SSP1 = Selfloop(SP1,[2,3,6,7,8]) (2,13)
SSP2 = Selfloop(SP2,[1,2,4,5,8]) (2,13)
SPEC = Sync(SSP1,SSP2) (4,20) Blocked_events = None

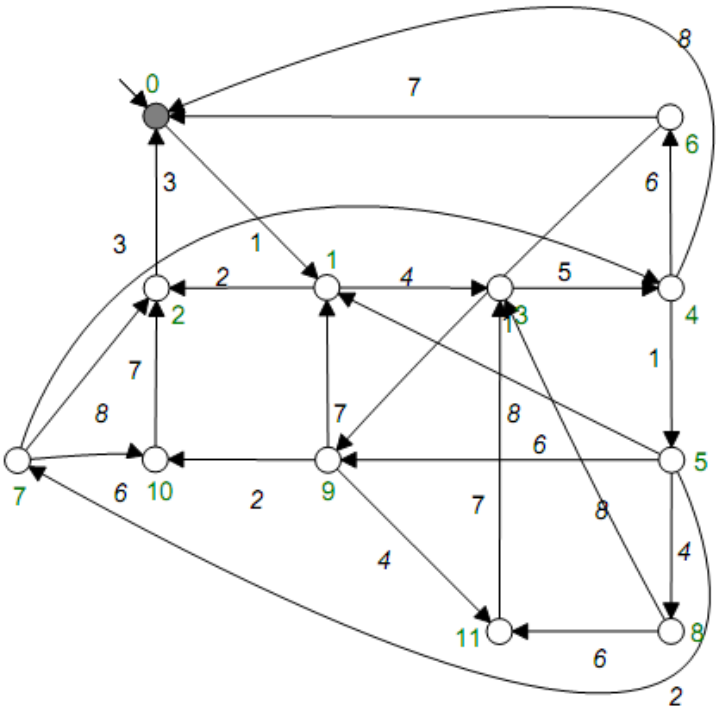
SUPER = Supcon(PLANT,SPEC) (12,24)
CDAT = Condat(PLANT,SUPER) Controllable.

SMALL = Allevents(PLANT) (1,14)

<u>Kontrol verisi:</u>

3: 1	8: 1
10: 3	11: 1

Elde edilen gözetici modeli Şekil 7.17'de görülmektedir.



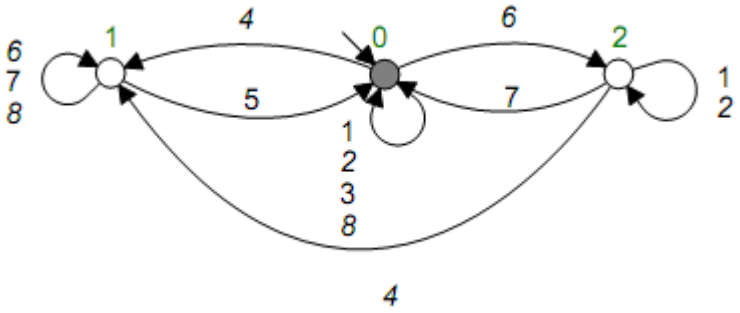
Şekil 7.17 Küçük imalat sistemi için elde edilen gözetici (SUPER)

5. Adım: SIMSUP ve indirgenmiş kontrol verisinin (SIMDAT) elde edilmesi

4. adımda elde edilen gözetici (SUPER) ve kontrol verisi (CDAT) kullanılarak indirgenmiş gözetici SIMSUP ve bu gözeticiye ait kontrol verisi SIMDAT aşağıda görüldüğü gibi hesaplanır.

SIMSUP = Supreduce(PLANT,SUPER,CDAT) (3,14;slb=3)
SIMDAT= Condat(PLANT,SIMSUP) Controllable.

İndirgenmiş gözetici SIMSUP Şekil 7.18'de görülmektedir.



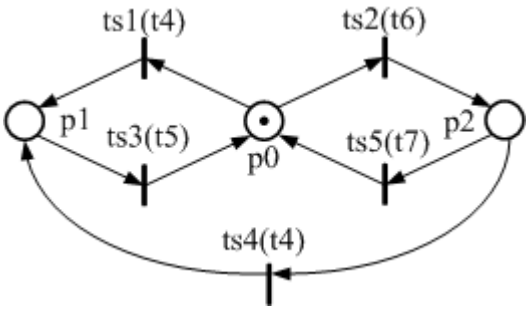
Şekil 7.18 Küçük imalat sistemi için elde edilen indirgenmiş gözetici (SIMSUP)

Kontrol verisi:

1: 1 2: 3

6. Adım: Elde edilen gözeticinin (SIMSUP) auto-net yapısına dönüştürülmesi

5. adımda elde edilen indirgenmiş gözeticinin (SIMSUP) auto-nete dönüştürülmüş biçimi Şekil 7.19'da görülmektedir. Bu modelde 3 mevki ve 5 adet geçiş bulunmaktadır. Başlangıç durumu, p0 mevkisine bir adet jeton eklenerek sağlanmaktadır.



Şekil 7.19 Şekil 7.18'de görülen indirgenmiş gözeticinin auto-net karşılığı

7. Adım: Kontrol poliçesinin yazılan kod ile hesaplanması

TCT hesaplamaları sonucunda elde edilen indirgenmiş gözetici SIMSUP ve kontrol verisi SIMDAT birer dosyada saklanmaktadır. Bir önceki kısımda anlatılan "cp.exe" programı çalıştırılarak bu iki dosyanın ismi programa girilir. Program bu iki dosyadan gerekli bilgileri okuyarak bir önceki bölümde anlatılan matrisleri elde eder ve gerekli hesaplamaları yaparak kontrol poliçesini oluştur. Programın bu iki dosya için ürettiği çıktı aşağıda görülmektedir.

Program Çıktısı:

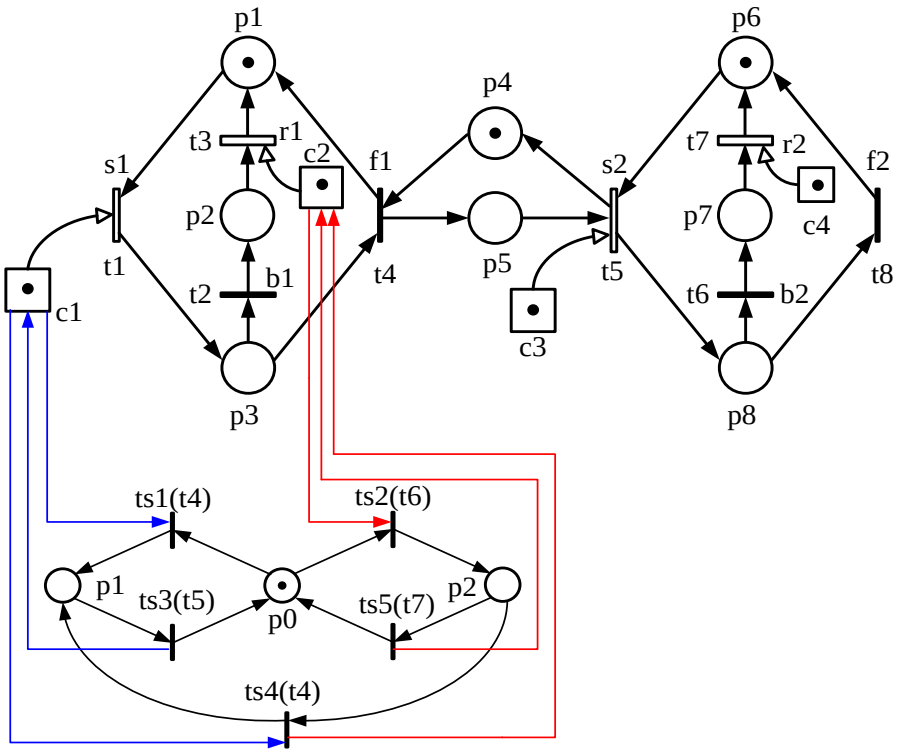
```
-------------------------------------------------------------------------------
Computation for Supervisor:simsup.ads  and Control data:simdat.pdt
-------------------------------------------------------------------------------
-->Transition Coding for Supervisor:sfact.ads
Transition name      TCT Coding
---------------      ----------
ts1            -->  [0 4 1]
ts2            -->  [0 6 2]
ts3            -->  [1 5 0]
ts4            -->  [2 4 1]
ts5            -->  [2 7 0]

-->Incedence Matrix (WS) of Supervisor:sfact.ads
-1 -1  1  0  1
 1  0 -1  1  0
 0  1  0 -1 -1

-->Control Data (sfact.pdt) in Matrix Form
 0  0  0  0
 1  0  0  0
 0  1  0  0

-->Control Policy
-1  0  1 -1  0
 0 -1  0  1  1
 0  0  0  0  0
 0  0  0  0  0

-->Initial Marking Of Control Places
Control Place     Number of tokens
     c1              1
     c3              1
     c5              1
     c7              1

-->Control Policy in Informal Form
Control Place    :  c1
Pre Transitions  :  ts3
Post Transitions:   ts1 ts4
Initial Marking  :  1
```

```
Control Place    :  c3
Pre Transitions  :  ts4, ts5
Post Transitions :  ts2
Initial Marking  :  1

Control Place    :  c5
Pre Transitions  :
Post Transitions :
Initial Marking  :  1

Control Place    :  c7
Pre Transitions  :
Post Transitions :
Initial Marking  :  1
```

Yukarıdaki sonuçlardan görüldüğü gibi, kontrol mevkilerin Pre ve Post oklarının hangi geçişlere bağlanacağı ve başlangıçta kontrol mevkilerinin jeton işaretlemesinin ne olacağı hesaplanmaktadır.

8. Adım: Kapalı çevrim derlenmiş denetleyicinin elde edilmesi

Şekil 7.14'teki CtlPN modeli, Şekil 7.19'daki auto-net ve 7. adımda hesaplanan bilgiler kullanılarak elde edilen kapalı çevrim kontrol sistemi Şekil 7.20'de görülmektedir. Şekilden ve yukarıdaki program çıktılarından da görülebileceği gibi c3 ve c5 isimli kontrol mevkileri için hesaplama sonucunda Pre veya Post oku bulunmamış ancak başlangıçta jeton bulunduracağı hesaplanmıştır. Kontrol mevkileri, kontrol edilebilir geçişlere yetkileme oku ile bağlandığından bu geçiş tetiklense bile içerisindeki jeton eksiltilmemektedir. Herhangi bir Pre veya Post okuna sahip olmayan ancak içerisinde jeton bulunan bir kontrol mevkisine sahip kontrol edilebilir geçiş her zaman tetiklenebilir demektir.



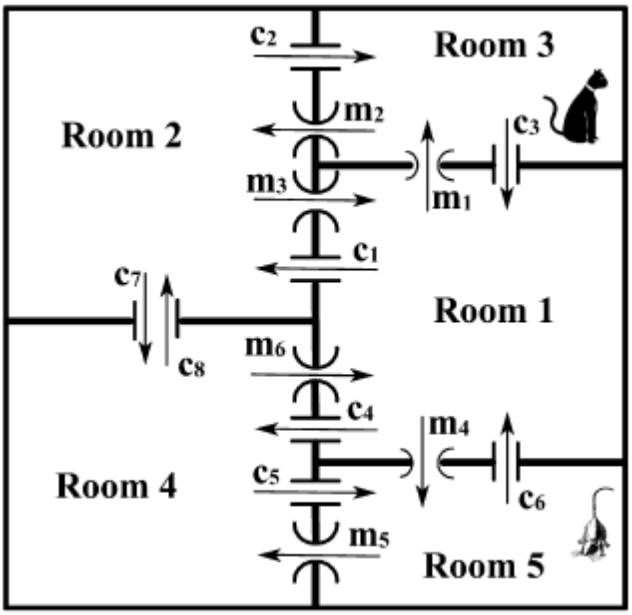
Şekil 7.20 Küçük İmalat sistemi için elde edilen kapalı çevrim kontrol sistemi

7.2.2 Kedi-fare probleminin önerilen yöntem ile çözümü

Bu örnekte kullanılan kedi-fare problemi, AOS'ler konusunda çalışan birçok araştırmacı tarafından kullanılan popüler bir örnektir. Kedi-fare probleminin çözümü ilk kez Gözetimli Kontrol Teorisi kullanılarak Ramadge ve Wonham tarafından sunulmuştur [5]. Aynı problem için farklı bir çözüm ise Uzam tarafından PN'ler ve Bölgeler Teorisi (Theory of Regions) kullanılarak önerilmiştir [42]. Kedi-fare problemi için CtlPN yapısı kullanılarak gözetici hesabı Boel ve diğerleri tarafından çalışılmıştır [110]. Bu problemde bir kedi ve bir farenin Şekil 7.21'de görülen bir labirentte bulundukları varsayılmaktadır. Bu problemdeki kedi ve fare, iki mobil robot veya bir esnek üretim sisteminde bulunan iki AGV(Automatic Guided Vehicle) olarak ta düşünülebilir. Labirent beş odadan oluşmaktadır (Room1, Room2, …, Room5). Başlangıçta kedinin 3. odada farenin ise 5. odada bulunduğu varsayılmaktadır. Odalar arasında kedi için özelleştirilmiş c1, c2, …, c8 olmak üzere sekiz kapı, fare için özelleştirilmiş m1, m2, …, m6 olmak üzere altı adet kapı bulunmaktadır. Kedi ve fare kendisi için özelleştirilmiş kapıdan şekilde görülen yönlerde geçebilmektedirler. Labirentte bulunan

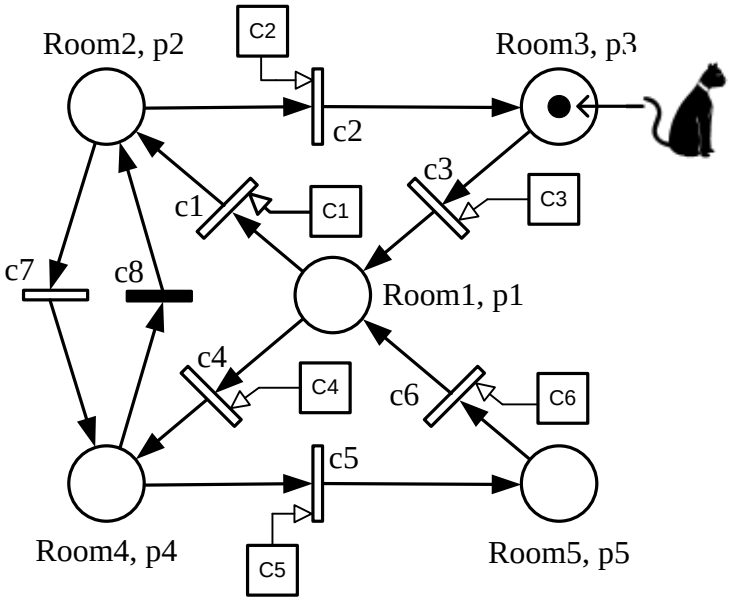
c7 ve c8 kapıları dışındaki tüm kapılar kedi ve farenin hareketini kontrol etmek üzere açılıp kapatılabilmektedir. Bu örnek için kontrol spesifikasyonları aşağıdaki gibidir:

1. Kedi ve fare aynı anda aynı odada bulunmamalıdır.
2. Kedi ve farenin her ikisi de başlangıç odalarına geri dönebilmelidir.

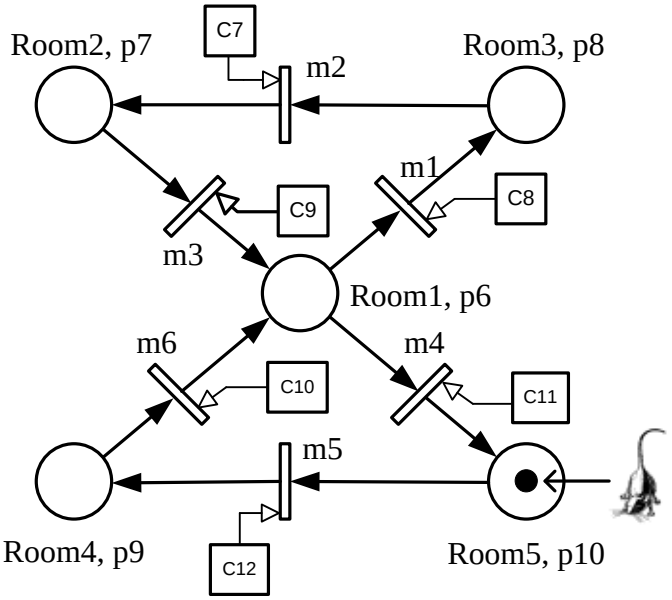


Şekil 7.21 Kedi ve farenin bulunduğu labirent

Kedi ve farenin bu labirentteki hareketleri sırası ile Şekil 7.22 ve Şekil 7.23'te görülen CtlPN modelleri kullanılarak ifade edilebilmektedir. Bu modellerdeki mevkiler kedi ve farenin bulundukları odaları, geçişler ise kapıları simgelemektedir. Bu modellerde bulunan geçişler için tanımlanan TCT etiketleri Çizelge 7.3'te görülmektedir. Önerilen yöntem ile bu problem için gözetici hesaplanabilmesi için CtlPN modelinde bulunan kontrol mevkileri ihmal edilerek bu modellerin otomata karşılıklarının elde edilmesi gerekir. Ayrıca yukarıda görülen kontrol spesifikasyonlarının da otomata olarak modellenmesi gerekmektedir. Kedi ve fare için oluşturulan CtlPN modellerindeki her bir mevkinin otomata karşılığı Şekil 7.24'te görülmektedir.



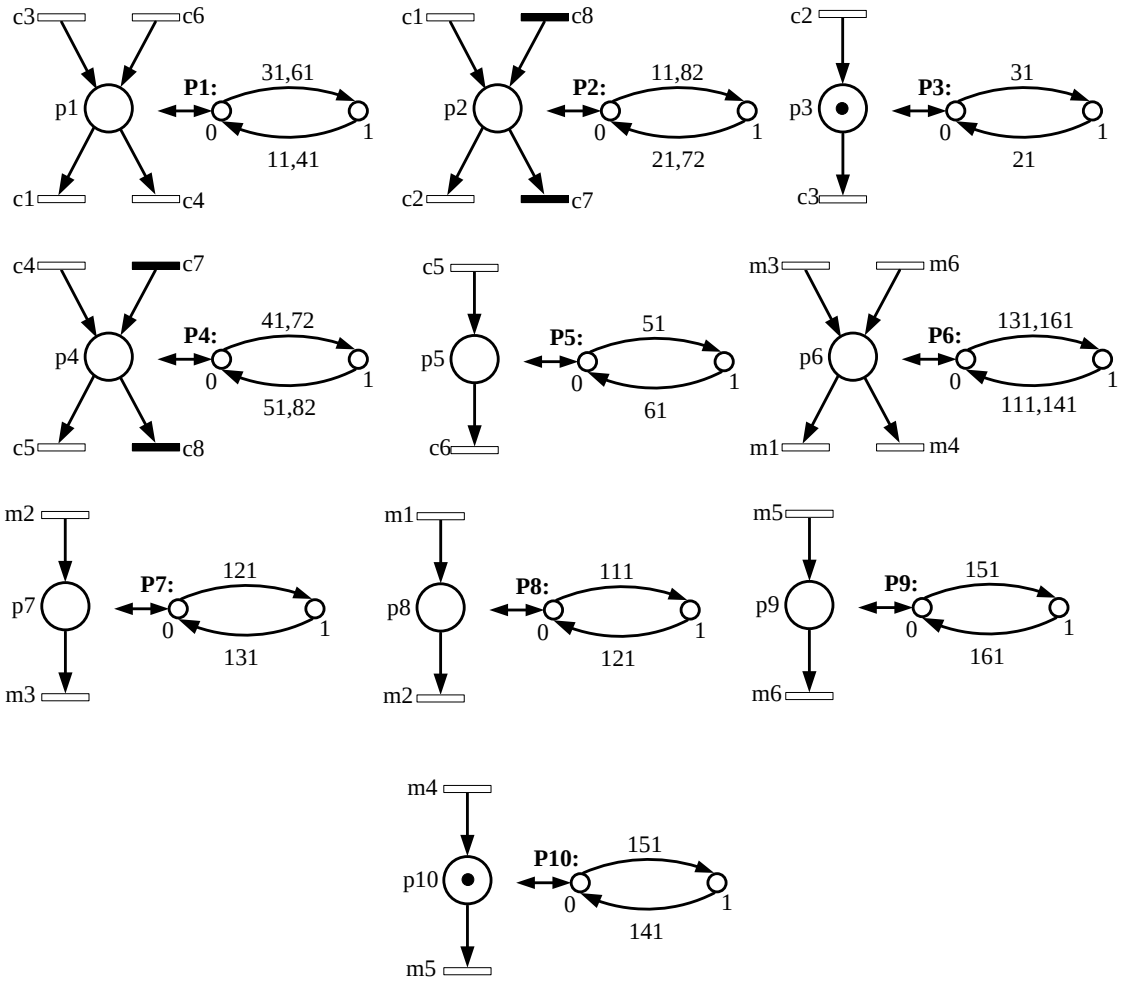
Şekil 7.22 Kedinin labirentteki hareketine ait CtlPN modeli



Şekil 7.23 Farenin labirentteki hareketine ait CtlPN modeli

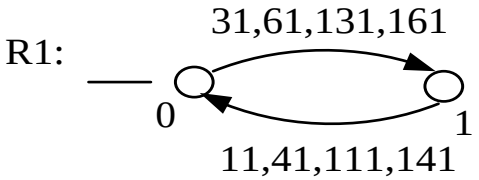
Çizelge 7.3 Kedi ve Fare için tanımlananTCT etiketleri

Geçiş adı	c1	c2	c3	c4	c5	c6	c7	c8
TCT etiketi	11	21	31	41	51	61	72	82
Geçiş adı	m1	m2	m3	m4	m5	m6		
TCT etiketi	111	121	131	141	151	161		

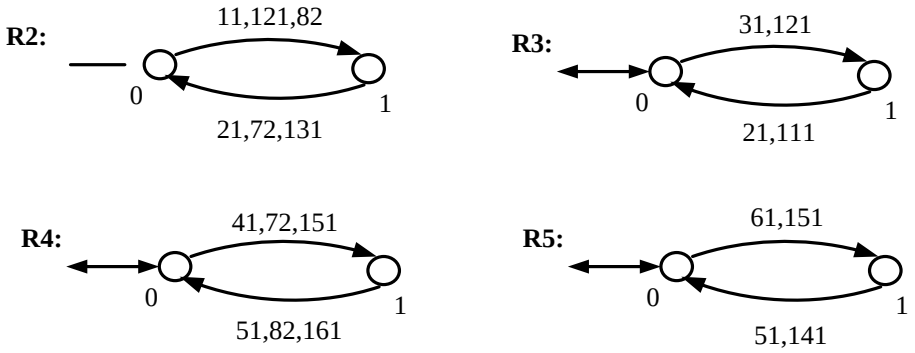


Şekil 7.24 CtlPN modelindeki mevkilerin otomata karşılıkları

Bu örnek için yukarıda belirtilen kontrol spesifikasyonlarının otomata olarak modellenmesi gerekmektedir. Kedi ve farenin aynı anda aynı odayı paylaşmaması için odalar tek tek göz önünde bulundurulmalıdır. Örneğin 1. oda için kedi p1 mevkisinde iken farenin p6 mevkisinde bulunmaması gerekmektedir. Bu spesifikasyon Şekil 7.25'te görüldüğü gibi modellenebilir. Modelden görülebileceği gibi p1 ve p6 mevkilerinin herhangi bir giriş geçişi gerçekleştikten sonra, bir çıkış geçişi gerçekleşmeden yeni bir giriş geçişi gerçekleşememektedir. Tüm odalar için ayrı ayrı oluşturulan spesifikasyonlar Şekil 7.26'da görülmektedir.

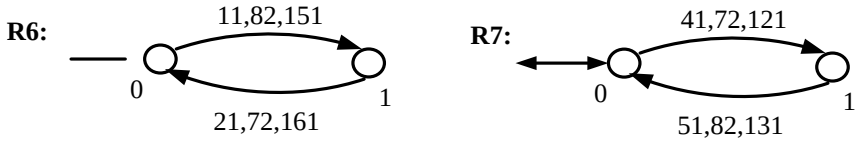


Şekil 7.25 İlk oda için gereken kontrol spesifikasyonunun otomata modeli



Şekil 7.26 Diğer odalar için gerekli kontrol spesifikasyonlarının otomata modelleri

Kedi için oluşturulan CtlPN modelinde bulunan c7 ve c8 geçişleri kontrol edilemez geçişlerdir. Bu geçişlerden dolayı, eğer fare 4. odada ise kedi 2. odada bulunmamalıdır. Benzer olarak, eğer fare 2. odada ise kedi 4. odada olmamalıdır. Bu iki durumu ifade etmek için de iki spesifikasyon modeli kullanılmaktadır. Bu modeller Şekil 7.27'de görülmektedir.



Şekil 7.27 Kontrol edilemez c7 ve c8 geçişleri sebebiyle gereken spesifikasyonlarının otomata modelleri

Yukarıda belirlenen otomata modelleri ve spesifikasyonlar kullanılarak kedi-fare probleminin çözümü için gerçekleştirilen TCT hesaplamaları aşağıda görülmektedir.

```
P1 = Create(P1,[mark 0],[tran [0,31,1],[0,61,1],[1,11,0],[1,41,0]])  (2,4)
P2 = Create(P2,[mark 0],[tran [0,11,1],[0,82,1],[1,21,0],[1,72,0]])  (2,4)
P3 = Create(P3,[mark 0],[tran [0,31,1],[1,21,0]])  (2,2)
P4 = Create(P4,[mark 0],[tran [0,41,1],[0,72,1],[1,51,0],[1,82,0]])  (2,4)
P5 = Create(P5,[mark 0],[tran [0,51,1],[1,61,0]])  (2,2)
P6 = Create(P6,[mark 0],[tran [0,131,1],[0,161,1],[1,111,0],[1,141,0]])( 2,4)
P7 = Create(P7,[mark 0],[tran [0,121,1],[1,131,0]])  (2,2)
P8 = Create(P8,[mark 0],[tran [0,111,1],[1,121,0]])  (2,2)
P9 = Create(P9,[mark 0],[tran [0,151,1],[1,161,0]])  (2,2)
P10 = Create(P10,[mark 0],[tran [0,151,1],[1,141,0]])  (2,2)

R1 = Create(R1,[mark 0],[tran [0,31,1],[0,61,1],[0,131,1],[0,161,1],[1,11,
0],[1,41,0],[1,111,0],[1,141,0]])  (2,8)
R2 = Create(R2,[mark 0],[tran [0,11,1],[0,82,1],[0,121,1],[1,21,0],[1,72,0],[1,131,0]])
(2,6)
```

```
R3 = Create(R3,[mark 0],[tran [0,31,1],[0,121,1],[1,21,0],[1,111,0]])  (2,4)
R4 = Create(R4,[mark 0],[tran [0,41,1],[0,72,1],[0,151,1],[1,51,0],[1,82,0],[1,161,0]])
(2,6)
R5 = Create(R5,[mark 0],[tran [0,61,1],[0,151,1],[1,51,0],[1,141,0]])  (2,4)
R6 = Create(R6,[mark 0],[tran [0,11,1],[0,82,1],[0,151,1],[1,21,0],[1,72,0],[1,161,0]])
(2,6)
R7 = Create(R7,[mark 0],[tran [0,41,1],[0,72,1],[0,121,1],[1,51,0],[1,82,0],[1,131,0]])
(2,6)
PLANT = Sync(P1,P2)  (4,13)  Blocked_events = None
PLANT = Sync(PLANT,P3)  (8,22)  Blocked_events = None
PLANT = Sync(PLANT,P4)  (16,40)  Blocked_events = None
PLANT = Sync(PLANT,P5)  (5,8)  Blocked_events = None
PLANT = Sync(PLANT,P6)  (10,36)  Blocked_events = None
PLANT = Sync(PLANT,P7)  (20,77)  Blocked_events = None
PLANT = Sync(PLANT,P8)  (40,134)  Blocked_events = None
PLANT = Sync(PLANT,P9)  (80,288)  Blocked_events = None
PLANT = Sync(PLANT,P10)  (25,70)  Blocked_events = None

SPEC = Sync(R1,R2)  (4,22)  Blocked_events = None
SPEC = Sync(SPEC,R3)  (8,36)  Blocked_events = None
SPEC = Sync(SPEC,R4)  (16,72)  Blocked_events = None
SPEC = Sync(SPEC,R5)  (10,42)  Blocked_events = None
SPEC = Sync(SPEC,R6)  (20,66)  Blocked_events = None
SPEC = Sync(SPEC,R7)  (16,50)  Blocked_events = None

SUPER = Supcon(PLANT,SPEC)  (6,9)
CDAT = Condat(PLANT,SUPER)  Controllable.

CMSUP = Supreduce(PLANT,SUPER,CDAT)  (2,9;slb=2)
CMDAT = Condat(PLANT,RSUP)  Controllable.

CMALL = Allevents(PLANT)  (1,14)
```

TCT hesaplamaları sonucunda elde edilen gözetici CMSUP, kontrol verisi CMDAT ve CMALL birer dosyada saklanmaktadır. Bir önceki kısımda anlatılan "cp.exe" programı çalıştırılarak bu dosyaların isimleri programa girilir. Program bu dosyalardan gerekli bilgileri okuyarak önceki bölümde anlatılan matrisleri elde eder ve gerekli hesaplamaları yaparak kontrol poliçesini oluşturur. Programın bu iki dosya için ürettiği sonuçlar aşağıda görülmektedir.

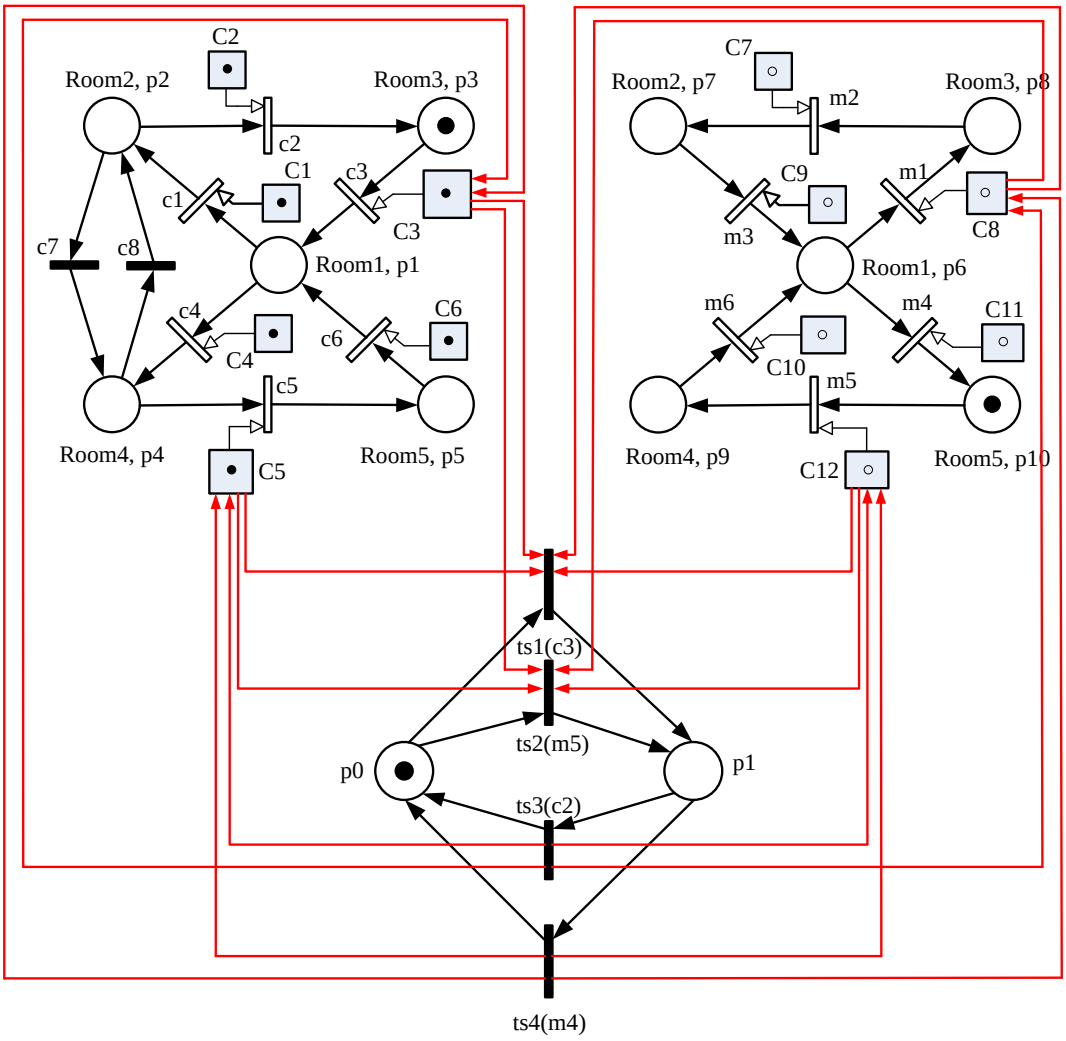
Program Çıktısı:

```
------------------------------------------------------------------------------------------
Computation for Supervisor:cmsup.ads  and Control data:cmdat.pdt
------------------------------------------------------------------------------------------
-->Transition Coding for Supervisor:cmsup.ads
```

```
Transition name      TCT Coding
----------------     ----------
ts1            -->  [0 31 1]
ts2            -->  [0 151 1]
ts3            -->  [1 21 0]
ts4            -->  [1 141 0]

-->Incedence Matrix (WS) of Supervisor:cmsup.ads
-1 -1  1  1
 1  1 -1 -1

-->Control Data (cmdat.pdt) in Matrix Form
 0  0  0  0  0  0  0  0  0  0  0  0
 0  0  1  0  1  0  1  0  0  0  1  0

-->Control Policy
 0  0  0  0
 0  0  0  0
-1 -1  1  1
 0  0  0  0
-1 -1  1  1
 0  0  0  0
-1 -1  1  1
 0  0  0  0
 0  0  0  0
 0  0  0  0
-1 -1  1  1
 0  0  0  0

-->Initial Marking of Control Place
Control Place     Number of tokens
    c11               1
    c21               1
    c31               1
    c41               1
    c51               1
    c61               1
    c111              1
    c121              1
    c131              1
    c141              1
    c151              1
    c161              1

-->Control Policy in Informal Form
Control Place    :  c11
Pre Transitions  :
Post Transitions :
Initial Marking  :  1

Control Place    :  c21
Pre Transitions  :
Post Transitions :
Initial Marking  :  1

Control Place    :  c31
Pre Transitions  :  ts3 ts4
```

```
Post Transitions :  ts1 ts2
Initial Marking  :  1

Control Place    :  c41
Pre Transitions  :
Post Transitions :
Initial Marking  :  1

Control Place    :  c51
Pre Transitions  :  ts3 ts4
Post Transitions:  ts1 ts2
Initial Marking  :  1

Control Place    :  c61
Pre Transitions  :
Post Transitions :
Initial Marking  :  1

Control Place    :  c111
Pre Transitions  :  ts3 ts4
Post Transitions :  ts1 ts2
Initial Marking  :  1

Control Place    :  c121
Pre Transitions  :
Post Transitions :
Initial Marking  :  1

Control Place    :  c131
Post Transitions :
Pre Transitions  :
Initial Marking  :  1

Control Place    :  c141
Pre Transitions  :
Post Transitions :
Initial Marking  :  1

Control Place    :  c151
Pre Transitions  :  ts3 ts4
Post Transitions :  ts1 ts2
Initial Marking  :  1

Control Place    :  c161
Pre Transitions  :
Post Transitions :
Initial Marking  :  1
--------------------------------------------------------------------------------
```

Yukarıdaki sonuçlardan görüldüğü gibi, kontrol mevkilerinin Pre ve Post oklarının hangi geçişlere bağlanacağı ve başlangıçta kontrol mevkilerin jeton işaretlemesinin ne olacağı hesaplanmıştır. Hesaplanan bu bilgiler ışığında elde edilen auto-net formundaki gözetici CtlPN modeline bağlanarak kapalı çevrim kontrol sistemi Şekil 7.28'de görüldüğü gibi elde edilmiştir.

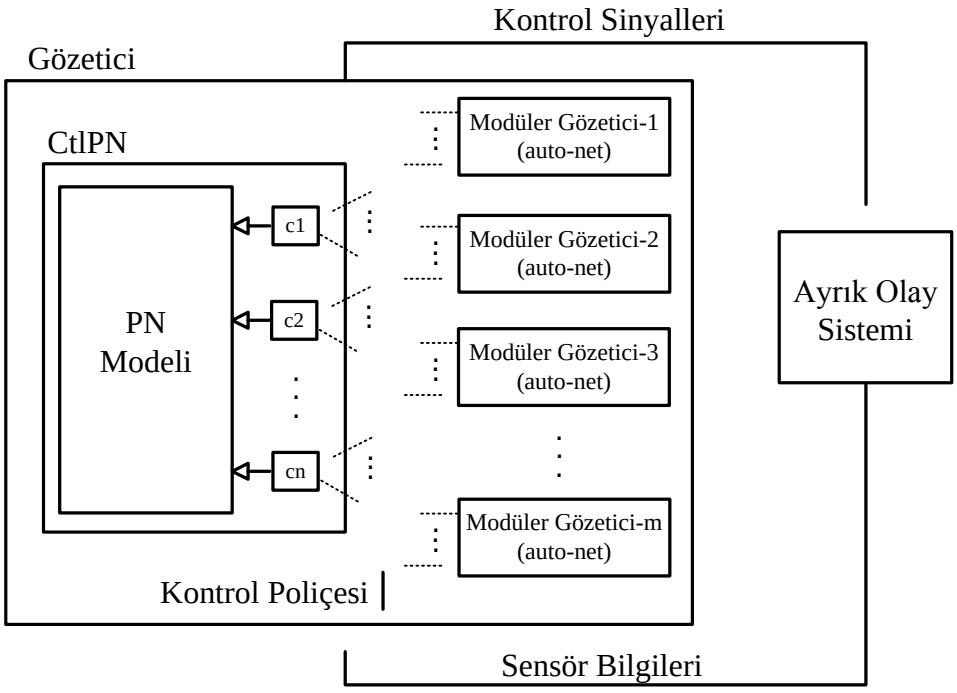


Şekil 7.28 Kedi fare problemi için elde edilen kapalı çevrim kontrol sistemi

7.3 Kontrollü Petri Ağları için Derlenmiş Modüler Denetleyici Sentezlenmesi

Önceki kısımda Kontrollü Petri Ağları (CtlPN) kullanılarak modellenmiş bir AOS için derlenmiş yekpare denetleyicilerin nasıl hesaplanabileceği detaylı bir biçimde açıklanmıştır. Bu kısmda, bir önceki kısımda açıklanan yöntemin modüler olarak nasıl uygulanabileceği anlatılmaktadır. CtlPN'ler için önerilen modüler melez yöntemin blok şeması Şekil 7.29'da görülmektedir. Modüler yöntemde sistemin kontrol edilmiş davranışını belirleyen spesifikasyonların her biri için ayrı bir gözetici hesaplanmaktadır. Şemadan da görülebileceği gibi, sistemi kontrol eden gözetici yapısı AOS'nin CtlPN modeli ve modüler gözeticilerden oluşmaktadır. CtlPN modelinde *n:* kontrol edilebilir geçiş sayısı olmak üzere kare ile gösterilen *n* tane kontrol mevkisi (*cn*) bulunmaktadır. Bu kontrol mevkileri ile modüler gözeticiler arasındaki bağlantılar sıradan oklar

vasıtasıyla sağlanmaktadır. Bu bağlantıların tamamı kontrol poliçesi olarak adlandırılmakta ve yekpare yöntemde önerilen hesaplama tekniği kullanılarak kolayca hesaplanabilmektedir. Kontrol poliçesi hesaplamalarında yekpare yöntem için önerilen ve algoritmaları önceki kısımda açıklanan "cp.exe" bilgisayar programının kullanımı mümkündür. Bu bilgisayar programı, TCT programından elde edilmiş gözeticiye ait .ADS dosyası, kontrol verisine ait .PDT dosyası ve gözetici hesaplanırken dikkate alınan sistem parçasındaki olay etiketlerini içeren .ADS dosyasını giriş olarak kullanmaktadır.



Şekil 7.29 CtlPN'ler kullanılarak modellenmiş AOS için önerilen modüler melez gözeticinin blok şeması

Modüler gözeticilerin hesaplanması önceki kısımlarda açıklanan şekilde yapılabilir. Modüler gözeticiler, sistemin kontrol edilmiş davranışını belirleyen spesifikasyonların her biri için sistemin sadece ilgili parçası dikkate alınarak TCT yazılımı ile elde edilebilir. CtlPN'ler için derlenmiş modüler gözetici tasarımı için takip edilecek işlem basamakları aşağıdaki gibi önerilmektedir.

1. Sistemin kontrol edilmemiş davranışını ifade eden Kontrollü Petri Ağı (CtlPN) modelinin verildiği varsayılır.
2. CtlPN modelinin her bir spesifikasyonu ilgilendiren kısmı PLANTx (x=1, 2, …, m m: modüler gözetici sayısı) şeklinde adlandırılarak bu alt modeldeki kontrol

mevkileri ve bu mevkileri geçişlere bağlayan oklar dikkate alınmadan geriye kalan PN modeli için eşdeğer tampon (buffer) modelleri elde edilir.

3. Spesifikasyonlar otomata olarak modellenir.
4. Her bir spesifikasyon (SPECx x=1, 2, …, m m: modüler gözetici sayısı) için sistemin ilgili parçası (PLANTx) belirlenerek bir modüler gözetici (MODSUPx x=1, 2, …, m m: modüler gözetici sayısı) ve bir kontrol verisi (CDATx x=1, 2, …, m) hesaplanır.
5. Hesaplanan modüler gözeticilerin birbirileri ile ve sistem ile ikilem (conflict) veya bloklanma (blocking) oluşturup oluşturmadığı TCT yazılımı ile test edilir.
6. Elde edilen modüler gözeticiler (MODSUPx) auto-net yapısına dönüştürülür.
7. Her bir modüler gözeticinin (MODSUPx) auto-neti için Incidence matrisi (Wx) (7.3) nolu eşitlik kullanılarak oluşturulur. Her bir CDATx kontrol verisi (7.4) nolu formül kullanılarak matris formuna dönüştürülür.
8. (7.5) ve (7.8) nolu formüller kullanılarak her bir gözetici için kontrol poliçesi (CPx) ve kontrol mevkilerinin başlangıç jeton işaretlemesi (μ_{cx0}) hesaplanır.
9. Her bir CPx matrisi ile hesaplanan oklar auto-netin geçişlerinden kontrol mevkilerine bağlanarak kapalı çevrim derlenmiş denetleyici elde edilir.

CtlPN'ler için derlenmiş modüler gözeticilerin hesaplanması için ilk olarak, sistemin kontrol edilmemiş davranışını tanımlayan CtlPN modelinin verildiği varsayılmaktadır. İkinci adımda, spesifikasyonlar dikkate alınarak CtlPN modeli alt modellere ayrılır. Bu alt Petri ağı modellerinde bulunan mevkilerin her biri eşdeğer tampon modellerine dönüştürülür. Bu dönüşüm işlemlerinde önceki kısımlarda anlatılan ve [90]'da önerilen yöntemler kullanılabilir. Bu modeller elde edilirken kontrol mevkileri ve bu mevkileri kontrol edilebilir geçişlere bağlayan yetkileme okları dikkate alınmaz. Oluşturulan tampon modelleri senkron kompozisyon işlemi ile birleştirilerek her bir spesifikasyon için ayrı ayrı sistem modelleri oluşturulur ve bu modeller PLANTx (x=1, 2, …, m m: modüler gözetici sayısı) olarak adlandırılır. Üçüncü adımda ise, spesifikasyonlar otomata olarak modellenir.

Dördüncü adımda, TCT yazılımı kullanılarak, her bir spesifikasyon (SPECx) için sistemin ilgili kısmı (PLANTx) belirlenerek bir modüler gözetici (MODSUPx) ve bir kontrol verisi (CDATx) hesaplanır. Beşinci adımda elde edilen gözeticilerin birbirleriyle ve sistem ile sorunsuz çalışabileceği test edilmelidir. Bu aşamada, modüler

gözeticilerin birbirleriyle ve sistemin tümüyle ikilemsiz ve bloklanmasız çalışıp çalışmadığı test edilir. Test işlemlerinin ardından elde edilen her bir otomata formundaki gözetici auto-net yapısına dönüştürülür.

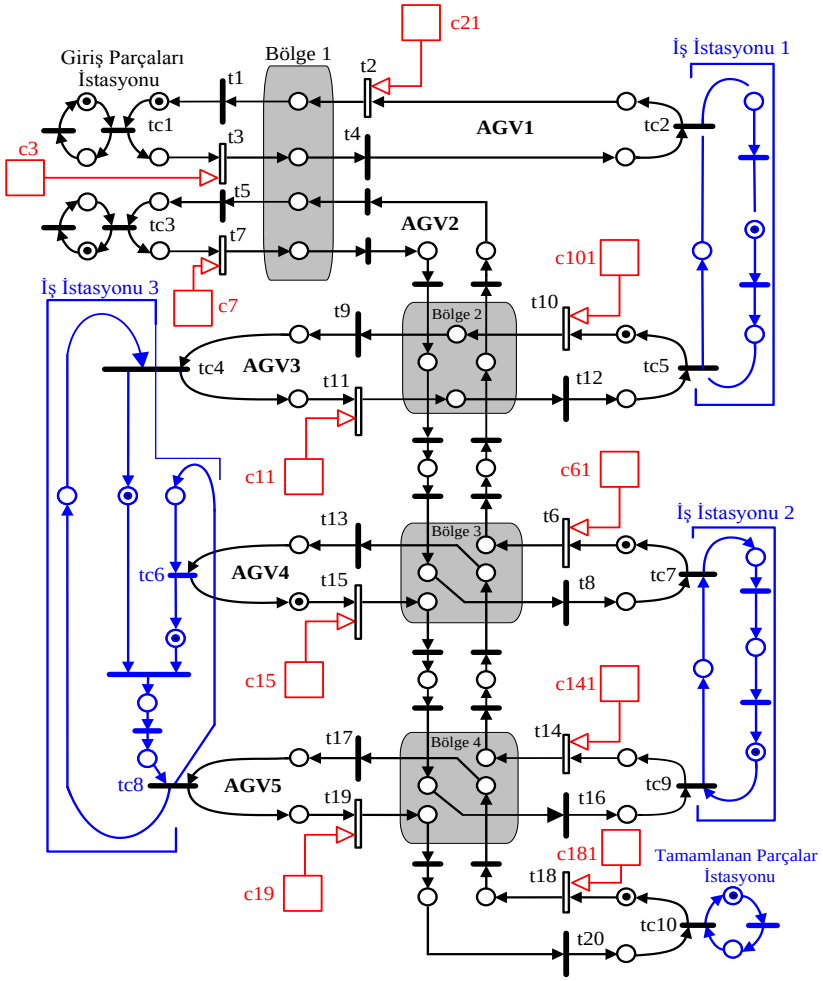
Modüler gözeticiler elde edildikten sonra, bu gözeticiler ile kontrol mevkileri arasındaki bağlantılar, CtlPN için bir önceki kısımda önerilen eşitlikler kullanılarak hesaplanabilir. Bu hesaplama için, yedinci adım olarak her bir modüler gözeticinin (MODSUPx) auto-neti için Incidence matrisi (Wx) (7.3) nolu eşitlik kullanılarak oluşturulur. Her bir CDATx kontrol verisi ise (7.4) nolu formül kullanılarak matris formuna dönüştürülür. Sekizinci adımda, bu matrisler ve (7.5) ile (7.8) nolu formüller kullanılarak her bir gözetici için kontrol poliçesi (CPx) ve kontrol mevkilerinin başlangıç jeton işaretlemesi (μ_{cn0}) hesaplanır.

Son adımda, hesaplanan her bir CPx matrisi ile tanımlanan oklar auto-netin geçişlerinden kontrol mevkilerine bağlanarak kapalı çevrim derlenmiş denetleyici elde edilir. Bu hesaplamaların daha kolay yapılabilmesi için bir önceki kısımda verilen algoritmalar kullanılarak elde edilen "cp.exe" programı kullanılabilir. Bu programa giriş olarak verilecek gözetici ve kontrol verisini içeren dosyalar için program, CP matrisini hesaplamakta ve CP matrisinin tanımladığı ok bağlantılarını bilgisayar ekranına ve bir log dosyasına yazmaktadır. Bu bilgiler kullanılarak kontrol poliçesi kolaylıkla gerçekleştirilebilmektedir.

7.3.1 Örnek: AGV içeren esnek üretim sistemi

Önerilen yönteme örnek olarak literatürde yaygın olarak kullanılan bir üretim sistemi incelenmiştir [30, 52, 93, 110, 111]. Bu üretim sistemi; üç adet iş istasyonu, iki adet giriş parçaları istasyonu ve bir adet tamamlanan parçalar istasyonu içermektedir. Parçaları istasyonlar arasında transfer etmek için ise beş adet otomatik yönlendirmeli araç (Automated Guided Vehicle-AGV) kullanılmaktadır. Bu araçlardan her biri parçaları diğer araçların da paylaştığı bölgelerden geçerek taşımaktadırlar. Bu sisteme ait CtlPN modeli Şekil 7.30'da görülmektedir [30]. Bu modelde kullanılan jetonlar, parçaları ve AGV'leri simgelemektedir. Ağın başlangıç jeton işaretlemesi sistemin başlangıç durumunu karakterize etmektedir. AGV'ler sadece istasyonlardan kontrol edilebilmektedir. Bu nedenle t2, t3, t6, t7, t10, t11, t14, t15, t18 ve t19 geçişleri kontrol

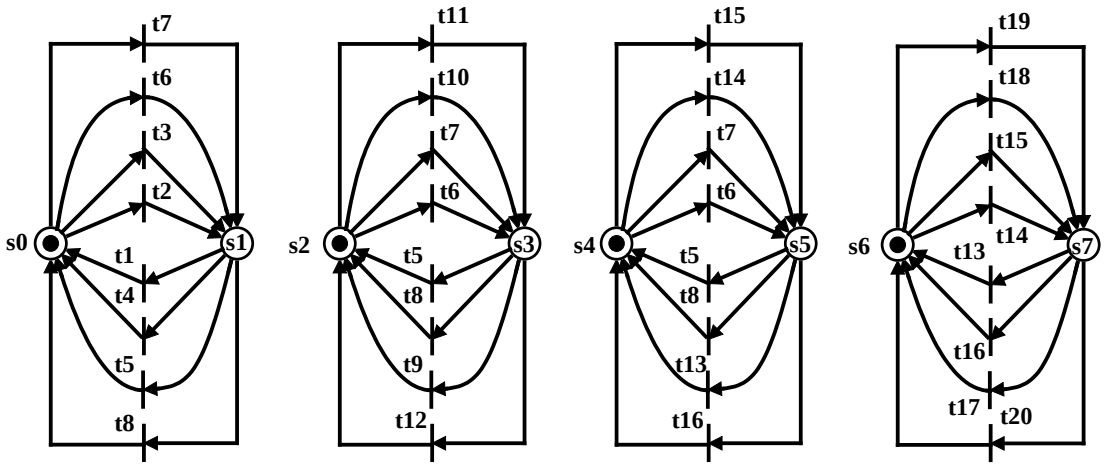
edilebilirdir. Şekilden görülebildiği gibi bu geçişlerin tetiklenebilmesi ancak geçişe bağlı olan kontrol mevkisinde jeton bulunması ile mümkündür.



Şekil 7.30 Esnek Üretim Sistemine ait CtlPN modeli [30]

AGV'ler bir istasyondan çıktıktan sonra varış istasyonuna ulaşana dek durdurulamazlar. Yani, seyahat halindeyken AGV'ler üzerine herhangi bir kontrol işlemi uygulanamaz. Model üzerinde gölgelendirilerek belirtilen ve Bölge 1, 2, 3 ve 4 olarak adlandırılan kısımlarda AGV'lerin yolları kesişmektedir. Bu paylaşımlı alanlarda aynı zamanda yalnızca bir AGV bulunabilmektedir. Bu üretim sistemi için tasarlanacak gözetici, paylaşımlı alanlardaki çarpışmayı önlemelidir. Bu örnek [52]'de incelenmiş ve önerilen melez yöntem kullanılarak problemin çözümü için Şekil 7.31'de görülen dört adet modüler gözetici hesaplanmıştır. Hesaplama için gereken adımların tüm detayları [52]'de mevcuttur. Hesaplamalarda her bir bölge için kontrolü sağlayacak modüler bir gözetici elde edilmiştir. Hesaplamalarda kullanılan TCT etiketleri ve işlemler

sonucunda elde edilen modüler gözeticiler ile kontrol verileri aşağıdaki görüldüğü gibidir.



a) b) c) d)

Şekil 7.31 a) Birinci bölge için hesaplanan MODSUP1'in auto-net modeli, b) İkinci bölge için hesaplanan MODSUP2'nin auto-net modeli, c) Üçüncü bölge için hesaplanan MODSUP3'ün auto-net modeli, d) Dördüncü bölge için hesaplanan MODSUP4'ün auto-net modeli [52]

Birinci bölgeye ait gözetici:

Sistem modeli PLANT1 için olay etiketlemesi

PLANT1:	t1	t2	t3	t4	t5	t6	t7	t8
TCT:	10	21	3	4	50	61	7	8

MODSUP1 # states: 2 state set: 0 ... 1 initial state: 0
marker states: 0 vocal states: none # transitions: 8
transitions: [0,3,1][0,7,1][0,21,1][0,61,1][1,4,0][1,8,0][1,10,0][1,50,0]
CDAT1
control data:
1: 3 7 21 61

İkinci bölgeye ait gözetici:

Sistem modeli PLANT2 için olay etiketlemesi

PLANT2:	t5	t6	t7	t8	t9	t10	t11	t12
TCT:	50	61	7	8	90	101	11	12

MODSUP2 # states: 2 state set: 0 ... 1 initial state: 0
marker states: 0 vocal states: none # transitions: 8
transitions: [0,7,1][0,11,1][0,61,1][0,101,1][1,8,0][1,12,0][1,50,0][1,90,0]

CDAT2
control data:
1: 7 11 61 101

Üçüncü bölgeye ait gözetici:

Sistem modeli PLANT3 için olay etiketlemesi

PLANT3:	t5	t6	t7	t8	t13	t14	t15	t16
TCT:	50	61	7	8	130	141	15	16

MODSUP3# states: 2 state set: 0 ... 1 initial state: 0
marker states: 0 vocal states: none # transitions: 8
transitions: [0,7,1][0,15,1][0,61,1][0,141,1][1,8,0][1,16,0][1,50,0][1,130,0]
CDAT3
control data:
1: 7 15 61 141

Dördüncü bölgeye ait gözetici:

Sistem modeli PLANT2 için olay etiketlemesi

PLANT4:	t13	t14	t15	t16	t17	t18	t19	t20
TCT:	130	141	15	16	170	181	19	20

MODSUP4 # states: 2 state set: 0 ... 1 initial state: 0
marker states: 0 vocal states: none # transitions: 8
transitions: [0,15,1][0,19,1][0,141,1][0,181,1][1,16,0][1,20,0][1,130,0][1,170,0]
CDAT4
control data: 1: 15 19 141 181

Şekil 7.31'de görülen modüler gözeticiler ile kontrol mevkileri arasındaki bağlantıları hesaplamak için gerekli olan matris işlemleri ve algoritmaları önceki kısımda anlatılan "cp.exe" programı kullanılarak gerçekleştirilmiştir. Bu program, giriş olarak verilen gözetici ve kontrol verisi bilgilerini kullanarak kontrol poliçesini hesaplamaktadır. Yukarıda görülen MODSUP gözeticileri ve CDAT kontrol verileri kullanılarak "cp.exe" programı ile hesaplanan kontrol poliçeleri Bölge 1, Bölge 2, Bölge 3 ve Bölge 4 için sırasıyla Şekil 7.32, Şekil 7.33, Şekil 7.34 ve Şekil 7.35'te görülmektedir.

Yazılan programın kullanımıyla, giriş olarak verilen MODSUP'e ait .ads dosyası ile CDAT'a air .pdt dosyası kullanılarak, modüler gözetici ile kontrol mevkileri arasındaki bağıntıyı tanımlayan CP matrisi hızlı bir biçimde hesaplanabilmektedir. Ayrıca program

CP matrisi ile tanımlanan bağlantıları daha kolay anlaşılabilir bir şekilde ekrana ve ilgili log dosyasına yazmaktadır. Kontrol mevkilerinin başlangıçta bulundurması gereken jeton işaretlemesi de program tarafından hesaplanmaktadır.

Şekillerdeki sonuçlardan görüleceği üzere, kontrol mevkilerinin Pre ve Post oklarının hangi geçişlere bağlanacağı ve başlangıçta kontrol mevkilerindeki jeton işaretlemesinin ne olacağı hesaplanmıştır. Hesaplanan bu bilgiler ışığında elde edilen auto-net formundaki gözetici CtlPN modeline bağlanarak kapalı çevrim kontrol sistemi Şekil 7.36'da görüldüğü gibi elde edilmiştir.

```
------------------------------------------------------------------------------
Computation for Supervisor:modsup1.ads  and Control data:cdat1.pdt
------------------------------------------------------------------------------

-->Transition Coding for Supervisor:modsup1.ads

Transition name       TCT Coding
---------------       ----------
ts1               -->  [0 3 1]
ts2               -->  [0 7 1]
ts3               -->  [0 21 1]
ts4               -->  [0 61 1]
ts5               -->  [1 4 0]
ts6               -->  [1 8 0]
ts7               -->  [1 10 0]
ts8               -->  [1 50 0]

-->Incedence Matrix (WS) of Supervisor:modsup1.ads

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

-->Control Data (cdat1.pdt) in Matrix Form

0  0  0  0
1  1  1  1
0  0  0  0
0  0  0  0
0  0  0  0

--> Control Policy

-1 -1 -1 -1  1  1  1  1
-1 -1 -1 -1  1  1  1  1
-1 -1 -1 -1  1  1  1  1
-1 -1 -1 -1  1  1  1  1


-->Initial Marking of Control Places

Control Place     Number of tokens

c3                 1
c21                1
c61                1
c7                 1

--> Control Policy in Informal Form

Control Place     :   c3
Pre Transitions   :   ts5 ts6 ts7 ts8
Post Transitions  :   ts1 ts2 ts3 ts4
Initial Marking   :   1

Control Place     :   c21
Pre Transitions   :   ts5 ts6 ts7 ts8
Post Transitions  :   ts1 ts2 ts3 ts4
Initial Marking   :   1

Control Place     :   c61
Pre Transitions   :   ts5 ts6 ts7 ts8
Post Transitions  :   ts1 ts2 ts3 ts4
Initial Marking   :   1

Control Place     :   c7
Pre Transitions   :   ts5 ts6 ts7 ts8
Post Transitions  :   ts1 ts2 ts3 ts4
Initial Marking   :   1
------------------------------------------------------------------------------
```

Şekil 7.32 MODSUP1 ve CDAT1 için hesaplanan kontrol poliçesi

```
--------------------------------------------------------------------------------
Computation for Supervisor:modsup2.ads  and Control data:cdat2.pdt
--------------------------------------------------------------------------------

-->Transition Coding for Supervisor:modsup2.ads

Transition name       TCT Coding
---------------       ----------
ts1               -->  [0 7 1]
ts2               -->  [0 11 1]
ts3               -->  [0 61 1]
ts4               -->  [0 101 1]
ts5               -->  [1 8 0]
ts6               -->  [1 12 0]
ts7               -->  [1 50 0]
ts8               -->  [1 90 0]

-->Incedence Matrix (WS) of Supervisor:modsup2.ads

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

-->Control Data (cdat2.pdt) in Matrix Form

 0  0  0  0
 1  1  1  1
 0  0  0  0
 0  0  0  0
 0  0  0  0

--> Control Policy

-1 -1 -1 -1  1  1  1  1
-1 -1 -1 -1  1  1  1  1
-1 -1 -1 -1  1  1  1  1
-1 -1 -1 -1  1  1  1  1


--> Initial Marking of Control Places

Control Place     Number of tokens

      c61               1
      c7                1
      c101              1
      c11               1

--> Control Policy in Informal Form

Control Place     :   c61
Pre Transitions   :   ts5 ts6 ts7 ts8
Post Transitions  :   ts1 ts2 ts3 ts4
Initial Marking   :   1

Control Place     :   c7
Pre Transitions   :   ts5 ts6 ts7 ts8
Post Transitions  :   ts1 ts2 ts3 ts4
Initial Marking   :   1

Control Place     :   c101
Pre Transitions   :   ts5 ts6 ts7 ts8
Post Transitions  :   ts1 ts2 ts3 ts4
Initial Marking   :   1

Control Place     :   c11
Pre Transitions   :   ts5 ts6 ts7 ts8
Post Transitions  :   ts1 ts2 ts3 ts4
Initial Marking   :   1
--------------------------------------------------------------------------------
```

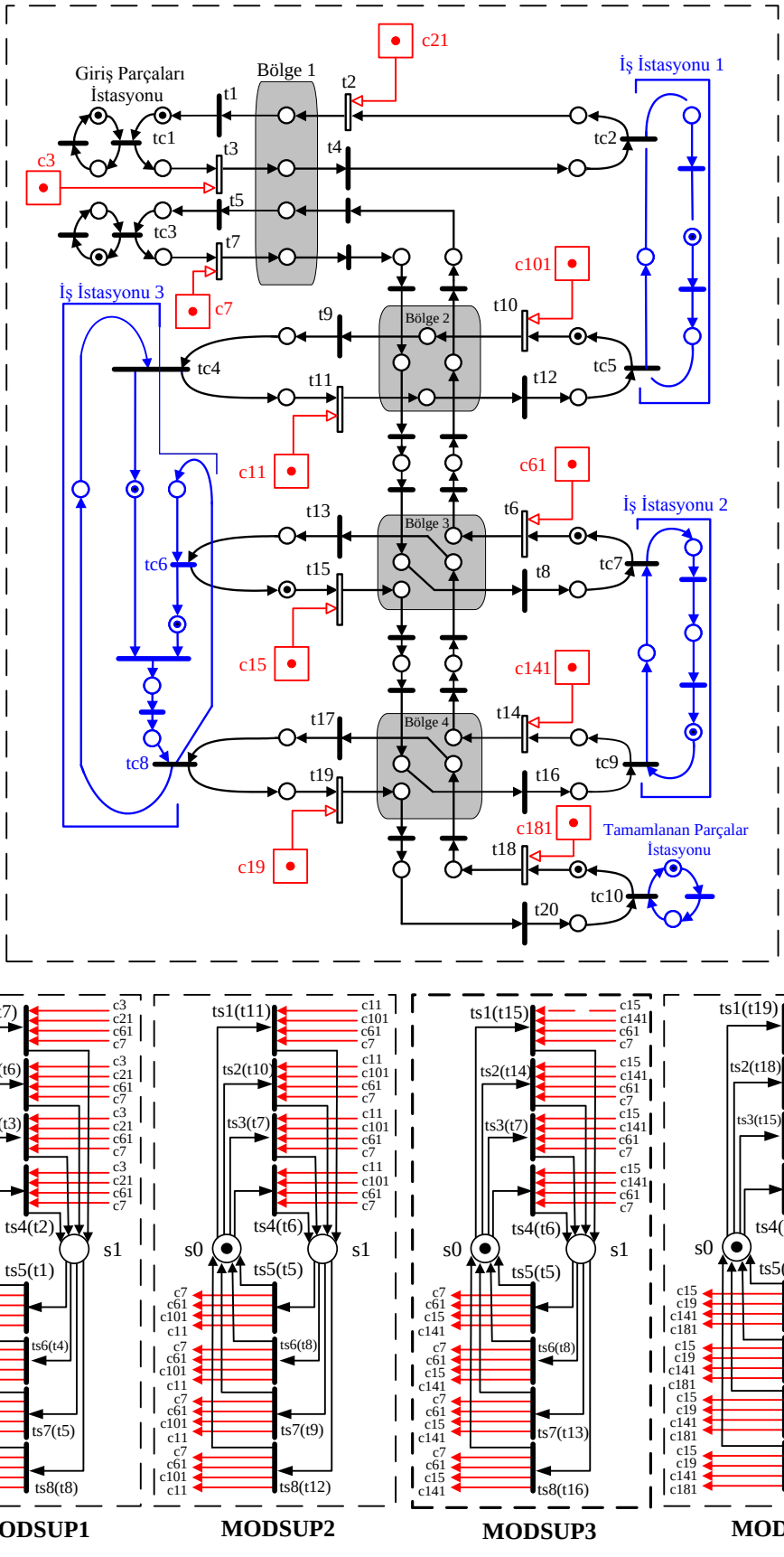
Şekil 7.33 MODSUP2 ve CDAT2 için hesaplanan kontrol poliçesi

```
--------------------------------------------------------------------------------
Computation for Supervisor:modsup3.ads  and Control data:cdat3.pdt
--------------------------------------------------------------------------------

-->Transition Coding for Supervisor:modsup3.ads

Transition name       TCT Coding
---------------       ----------
ts1               -->  [0 7 1]
ts2               -->  [0 15 1]
ts3               -->  [0 61 1]
ts4               -->  [0 141 1]
ts5               -->  [1 8 0]
ts6               -->  [1 16 0]
ts7               -->  [1 50 0]
ts8               -->  [1 130 0]

-->Incedence Matrix (WS) of Supervisor:modsup3.ads

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

-->Control Data (cdat3.pdt) in Matrix Form

 0  0  0  0
 1  1  1  1
 0  0  0  0
 0  0  0  0
 0  0  0  0

--> Control Policy

-1 -1 -1 -1  1  1  1  1
-1 -1 -1 -1  1  1  1  1
-1 -1 -1 -1  1  1  1  1
-1 -1 -1 -1  1  1  1  1


--> Initial Marking of Control Places

Control Place     Number of tokens

      c61               1
      c7                1
      c141              1
      c15               1

--> Control Policy in Informal Form

Control Place     :   c61
Pre Transitions   :   ts5 ts6 ts7 ts8
Post Transitions  :   ts1 ts2 ts3 ts4
Initial Marking   :   1

Control Place     :   c7
Pre Transitions   :   ts5 ts6 ts7 ts8
Post Transitions  :   ts1 ts2 ts3 ts4
Initial Marking   :   1

Control Place     :   c141
Pre Transitions   :   ts5 ts6 ts7 ts8
Post Transitions  :   ts1 ts2 ts3 ts4
Initial Marking   :   1

Control Place     :   c15
Pre Transitions   :   ts5 ts6 ts7 ts8
Post Transitions  :   ts1 ts2 ts3 ts4
Initial Marking   :   1
--------------------------------------------------------------------------------
```

Şekil 7.34 MODSUP3 ve CDAT3 için hesaplanan kontrol poliçesi

```
--------------------------------------------------------------------------------
Computation for Supervisor:modsup4.ads  and Control data:cdat4.pdt
--------------------------------------------------------------------------------

-->Transition Coding for Supervisor:modsup4.ads

Transition name       TCT Coding
---------------       ----------
ts1               -->  [0 15 1]
ts2               -->  [0 19 1]
ts3               -->  [0 141 1]
ts4               -->  [0 181 1]
ts5               -->  [1 16 0]
ts6               -->  [1 20 0]
ts7               -->  [1 130 0]
ts8               -->  [1 170 0]

-->Incedence Matrix (WS) of Supervisor:modsup4.ads

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

-->Control Data (cdat4.pdt) in Matrix Form

 0  0  0  0
 1  1  1  1
 0  0  0  0
 0  0  0  0
 0  0  0  0

-->Control Policy

-1 -1 -1 -1  1  1  1  1
-1 -1 -1 -1  1  1  1  1
-1 -1 -1 -1  1  1  1  1
-1 -1 -1 -1  1  1  1  1


--> Initial Marking of Control Places

Control Place     Number of tokens

      c141                1
      c15                 1
      c181                1
      c19                 1

-->Control Policy in Informal Form

Control Place     :   c141
Pre Transitions   :   ts5 ts6 ts7 ts8
Post Transitions  :   ts1 ts2 ts3 ts4
Initial Marking   :   1

Control Place     :   c15
Pre Transitions   :   ts5 ts6 ts7 ts8
Post Transitions  :   ts1 ts2 ts3 ts4
Initial Marking   :   1

Control Place     :   c181
Pre Transitions   :   ts5 ts6 ts7 ts8
Post Transitions  :   ts1 ts2 ts3 ts4
Initial Marking   :   1

Control Place     :   c19
Pre Transitions   :   ts5 ts6 ts7 ts8
Post Transitions  :   ts1 ts2 ts3 ts4
Initial Marking   :   1

--------------------------------------------------------------------------------
```

Şekil 7.35 MODSUP4 ve CDAT4 için hesaplanan kontrol poliçesi



Şekil 7.36 Esnek Üretim Sistemi için kapalı çevrim kontrol sistemi

7.4 Deneysel Endüstriyel Üretim Sisteminin CtlPN Kullanılarak Kontrolü

Bu kısımda, bir önceki kısımda detayları anlatılan Kontrollü Petri Ağları (CtlPN) için derlenmiş gözetici hesaplama yöntemlerinin, deneysel endüstriyel üretim sistemi üzerinde gerçekleştirilen uygulamaları incelenmektedir. İlk olarak önerilen yekpare yöntemin, üretim sisteminde bulunan halka besleme kolu kapasitesinin bir ve iki olduğu durumlar için, uygulanabilirliği açıklanmıştır. Daha sonra, TCT yazılımın *Supreduce(.)* komutu kullanılarak elde edilen indirgenmiş gözeticilerin halka besleme kolu kapasitesinin bir ve iki olduğu durumlar için PLC ile gerçekleştirilmesi açıklanmaktadır. Son olarak, bir önceki kısımda önerilmiş olan modüler yöntemin uygulanabilirliği halka besleme kolu kapasitesinin bir, iki ve beş olduğu durumlar için gösterilmiştir.

7.4.1 Kontrollü Petri ağı için yekpare gözetici tasarımı ve PLC ile gerçekleştirilmesi

7.4.1.1 Halka toplama kolu kapasitesinin 1 olduğu durum

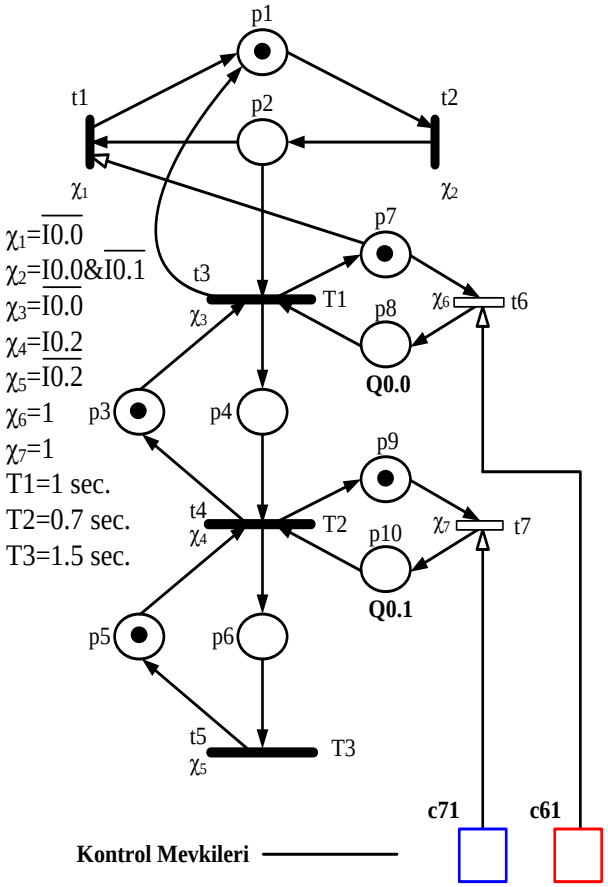
Detayları ikinci bölümde anlatılan deneysel endüstriyel üretim sistemin halka toplama kolu kapasitesinin 1 olduğu durum için kontrol edilmemiş CtlPN modeli Şekil 7.37'de görülmektedir. Bu model, daha önceki kısımlarda aynı sistem için kullanılan Petri ağı modelindeki kontrol edilebilir geçişlere yetkileme oku ile bağlanan kontrol mevkilerinin (c61, c71) eklenmesi ile elde edilmiştir.

Kontrol mevkileri, Petri ağında bulunan normal mevkilerden ayırt edilebilmesi amacıyla kare ile gösterilmiş ve ilgili geçişin TCT yazılımda kullanılan etiketi ile (t6 için TCT etiketi 61; t7 için TCT etiketi 71) adlandırılmıştır. Normal mevkilerin aksine kontrol mevkilerinin adlandırılmasında "p" harfi yerine "c" harfi kullanılmaktadır.

Şekil 7.37'den görülebileceği üzere CtlPN modelinde on adeti sıradan mevki P = {p1, p2, ..., p10} ve 2 adeti kontrol mevkisi C={c61, c71} olmak üzere toplam 12 adet mevki ile $\chi = \{\chi_1, \chi_2, ..., \chi_7\}$ harici tetikleme koşulları ile ilişkilendirilmiş yedi adet geçiş T = {t1, t2, ..., t7} bulunmaktadır. Harici tetikleme koşulları, sırası ile şunlardır: $\chi_1=\overline{I0.0}$, $\chi_2=I0.0\&\overline{I0.1}$, $\chi_3=\overline{I0.0}$, $\chi_4=I0.2$, $\chi_5=\overline{I0.2}$, $\chi_6=1$, $\chi_7=1$. Bu modelde, t3, t4 ve t5

geçişleri sırası ile 1 saniye, 0.7 saniye ve 1.5 saniye zaman gecikmelerine sahip zamanlı geçişlerdir. p7 ve p8 mevkileri sırası ile seçme selenoidinin *kapalı* ve *açık* durumunu göstermektedir. Benzer olarak p9 ve p10 mevkileri de birleştirme selenoidinin *kapalı* ve *açık* durumlarını simgelemektedir. p1, p3 ve p5 mevkilerinde bulunan jetonlar sırası ile seçme alanının, halka toplama kolunun ve birleştirme noktasının boş olduğunu simgelemektedir. p2, p4 ve p6 mevkilerinde jeton bulunması ise sırası ile seçme alanında, halka toplama kolunda ve birleştirme noktasında plastik halka olduğunu göstermektedir.

Başlangıçta, her iki selenoid te çalışmamakta yani kapalı durumdadır ve üretim sisteminin herhangi bir yerinde plastik halka yoktur. Seçme alanında plastik halka yoksa (p1 mevkisinde jeton varsa) ve seçme alanında plastik halka olduğu algılanmışsa ($\chi_2 = \mathrm{I0.0} \,\&\, \overline{\mathrm{I0.1}}$), t2 geçişi tetiklenir ve p1 mevkisindeki jeton alınarak p2 mevkisine bir jeton depolanır. Bu, seçme alanında bir plastik halka olduğu anlamına gelmektedir. Seçme alanındaki halka ya seçme selenoidi kullanılarak halka toplama koluna gider, ya da kullanılmayan halka toplama alanına gitmek üzere seçme alanından çıkar. Eğer halka seçme alanında bir halka varsa, seçme selenoidi çalışıyorsa, halka biriktirme kolu boş ise ve halka, seçme alanından ayrılmışsa ($\chi_3 = \overline{\mathrm{I0.0}}$) zamanlı geçiş t3, 1 saniye gecikme ile tetiklenmektedir. Bu geçişin tetiklenmesi ile p2 mevkisindeki jeton p4 mevkisine aktarılmaktadır. Bu, seçme alanındaki halkanın halka toplama koluna aktarıldığı ve bu işlemin bir saniye sürdüğü anlamına gelmektedir. Eğer p2 ve p7 mevkilerinde birer jeton bulunuyorsa ve $\chi_1 = \overline{\mathrm{I0.0}}$ tetikleme koşulu gerçekleşmişse t1 geçişi tetiklenir ve p2 mevkisindeki jeton p1 mevkisine aktarılır. Bu, seçme alanındaki halkanın, kullanılmayan halka toplama alanına gitmek üzere seçme alanından çıktığı anlamına gelir.



Şekil 7.37 Halka toplama kolu kapasitesinin 1 olduğu durum için CtlPN modeli

Eğer halka toplama kolunda halka varsa (p4 mevkisinde jeton varsa), birleştirme alanında halka yoksa (p5 mevkisinde jeton varsa), birleştirme (döner) selenoidi çalışıyorsa ve birleştirme noktasında bir halka algılanmışsa ($\chi_4 = \mathrm{I0.2}$), t4 zamanlı geçişi 0.7 saniye zaman gecikmesi ile tetiklenmektedir. Bu, halka toplama kolundaki halkanın, birleştirme selenoidi (döner selenoid) kullanılarak, birleştirme noktasına aktarıldığı ve bu işlemin 0.7 saniye sürdüğü anlamına gelmektedir.

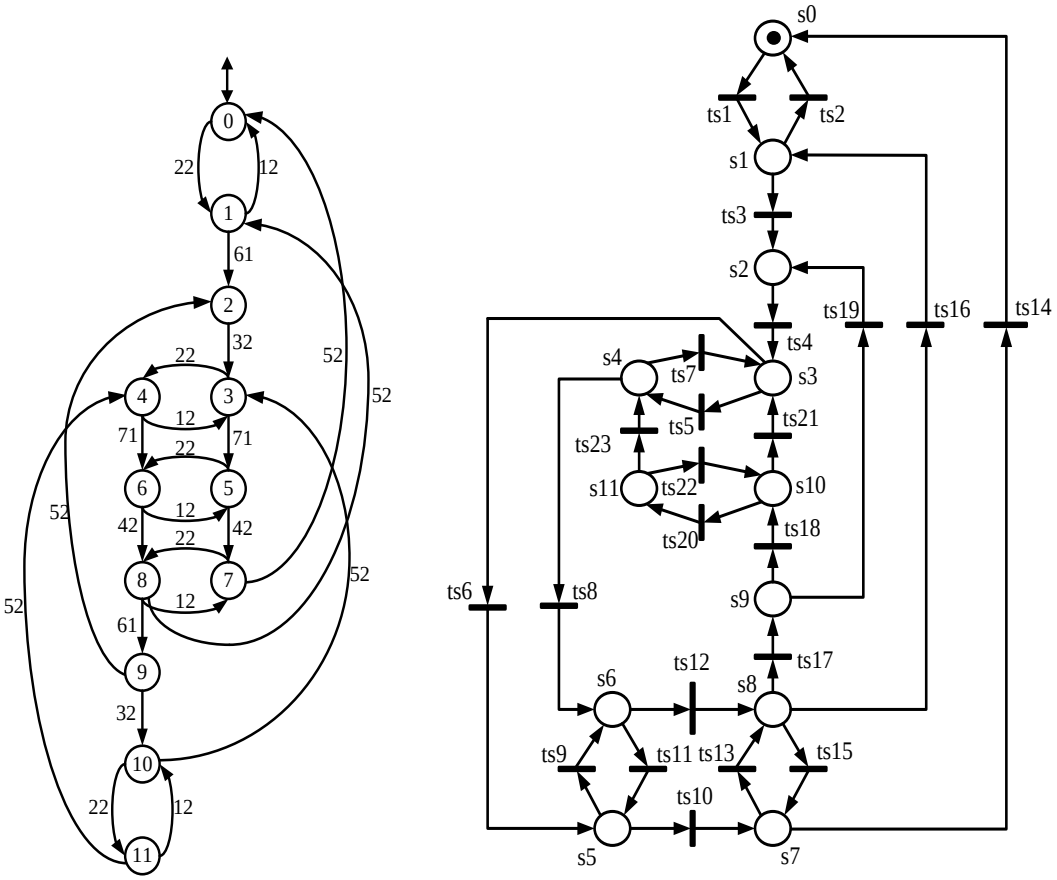
Eğer birleştirme noktasında bir halka varken (p6 mevkisinde bir jeton varsa), metal çubuk ile halka birleşerek birleştirme noktasından ayrılmış ise ($\chi_5 = \overline{\mathrm{I0.2}}$), t5 zamanlı geçişi 1.5 saniye gecikme ile tetiklenmektedir. Bu geçişin tetiklenmesi ile p6 mevkisindeki jeton p5 mevkisine aktarılmaktadır. Bu işlem, sistemde bir halka ile bir metal çubuğun birleştiğini ve sistemi terk ettiğini göstermektedir.

Şekil 7.37'den de görüleceği üzere Q0.0 ve Q0.1 aksiyonları p8 ve p10 mevkilerine atanmıştır. p8 mevkisinde jeton bulunması ile Q0.0 aksiyonu gerçekleşmekte ve itici selenoid (seçme selenoidi) çalışmaktadır. Benzer olarak p10 mevkisinde bir adet jeton bulunması ile Q0.1 aksiyonu gerçekleşmekte ve döner selenoid çalıştırılmaktadır. p8 ve p10 mevkilerinde jeton bulunabilmesi yani itici ve döner selenoidin çalışabilmesi t6 ve t7 kontrol edilebilir geçişlerinin tetiklenmesi ile mümkündür. Bu geçişlerin tetiklenmesi ise ilgili kontrol mevkilerinde jeton bulunması ile gerçekleşebilir. Kontrol mevkilerinde ne zaman jeton olup ne zaman jeton olmayacağını, kontrol spesifikasyonları dikkate alınarak hesaplanan gözetici sağlamaktadır. Halka toplama kolu kapasitesinin 1 olduğu durum için kontrol spesifikasyonları aşağıdaki gibidir:

1. Seçme selenoidi (itici selenoid) seçme alanında bir halka bulunduğunda ve halka toplama kolunda boş yer olduğunda çalışacaktır (SPEC1).
2. Birleştirme selenoidi (döner selenoid), halka toplama kolunda halka olduğunda ve birleştirme noktası boş olduğunda çalışacaktır (SPEC2).

Yukarıdaki spesifikasyonlar göz önünde bulundurularak, halka toplama kolu kapasitesinin 1 olduğu durum için [67]'de hesaplanan yekpare gözeticinin otomata modeli Şekil 7.38.(a)'da, auto-net modeli ise Şekil 7.38.(b)'de görülmektedir. Bu gözetici hesaplanırken, mevkilerin otomata karşılığına dönüştürülmesi aşamasında kontrol mevkileri ve bu mevkileri ilgili geçişe bağlayan yetkileme okları ihmal edilmektedir. Elde edilen gözetici ve kontrol verisi kullanılarak bir önceki kısımda algoritmaları açıklanan "cp.exe" programı ile gözetici ve kontrol mevkileri arasındaki bağlantıları tanımlayan kontrol poliçesi hesaplanmıştır. Hesaplamalarla elde edilen sonuçları gösteren program çıktısı Ek-D'de görülmektedir. Ek-D'de sunulan program çıktısından görülebileceği gibi kontrol mevkileri c61 ve c71 için başlangıç jeton işaretlemesi 0 olarak hesaplanmıştır. Bu, başlangıçta kontrol mevkilerinde jeton bulunmayacağı anlamına gelmektedir. Sisteme ait CtlPN modelinde bulunan kontrol mevkileri ile auto-nete dönüştürülmüş gözeticinin geçişleri arasındaki sıradan ok bağlantılarını belirleyen kontrol poliçesi, matris olarak "cp.exe" programı yardımıyla hesaplanmıştır. Auto-net gözeticinin geçişlerinden kontrol mevkilerine yönlenmiş olan sıradan oklar ***Pre,*** kontrol mevkilerinden auto-netin geçişlerine yönlenmiş oklar ise ***Post*** olarak adlandırılmaktadır. "cp.exe" programı Pre ve Post olarak tanımlanan okların hangi geçiş ve hangi kontrol mevkisi arasında olacağını liste halinde de çıktı olarak

üretmektedir. Halka toplama kolu kapasitesinin 1 olduğu durum için hesaplanan kontrol poliçesinin tamamı Çizelge 7.4'te görülmektedir. Hesaplanan bu kontrol poliçesinin gösterdiği sıradan ok bağlantıları ışığında, halka toplama kolu kapasitesinin 1 olduğu durum için elde edilen kapalı çevrim kontrol sistemi Şekil 7.39'da görülmektedir.



a) b)

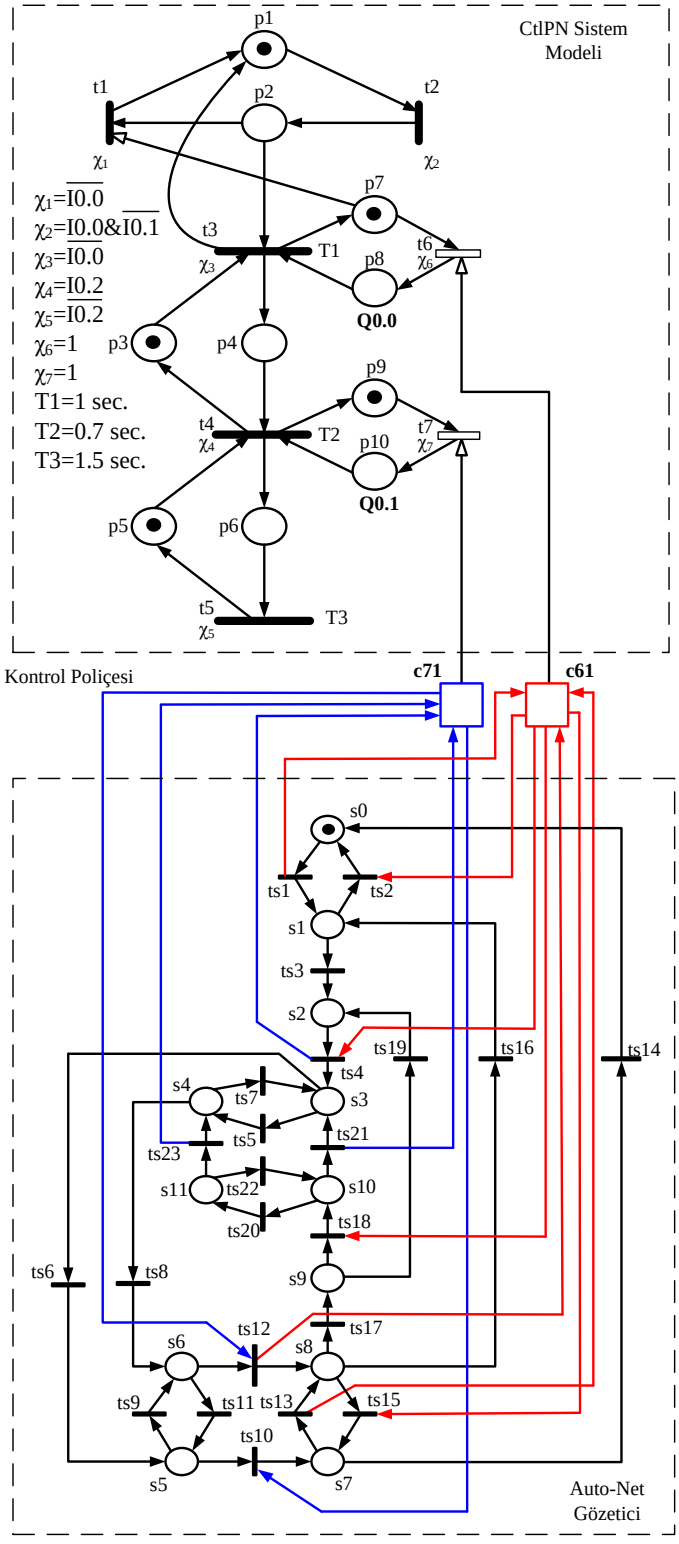
Şekil 7.38 a) Halka toplama kolu kapasitesinin 1 olduğu durum için hesaplanan yekpare gözeticinin otomata modeli, b) auto-net modeli [67]

Çizelge 7.4 Halka toplama kolu kapasitesinin 1 olduğu durum için hesaplanan kontrol poliçesi

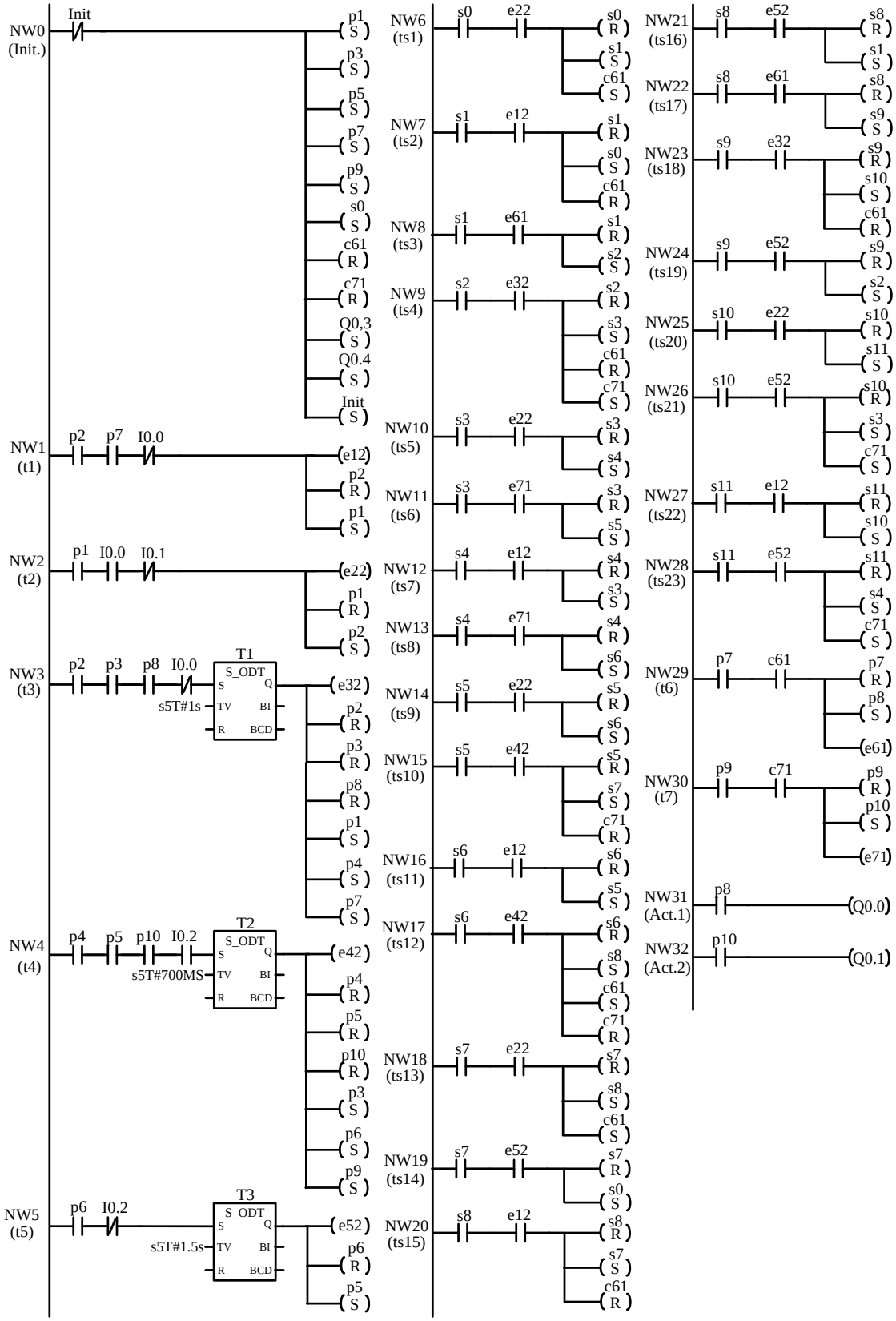
Kontrol mevkisi	Başlangıç Jeton Sayısı	Pre oklarının bağlanacağı geçişler	Post oklarının bağlanacağı geçişler
c61	0	ts1 ts12 ts13	ts2 ts4 ts15 ts18
c71	0	ts4 ts21 ts23	ts10 ts12

Şekil 7.39'da görülen kapalı çevrim kontrol sisteminin PLC ile gerçekleştirilmesi için önceki kısımlarda açıklanan metotlar kullanılmaktadır. Bu modeli gerçekleştirmek üzere elde edilen PLC merdiven diyagramı kodu Şekil 7.40'ta görülmektedir. Kodun ilk basamağında (NW0) başlangıç jeton işaretlemesi gerçekleştirilmektedir. Başlangıçta,

p1, p3, p5, p7, p9, s0, mevkileri ile alt ve üst konveyör motorlarının bağlı olduğu Q0.3 ve Q0.4 çıkışları set (lojik ‘1’) yapılmaktadır. Aynı basamakta, başlangıçta kontrol mevkilerinde jeton bulunmadığından bu mevkileri simgeleyen c61 ve c71 hafıza bitleri ise reset yapılmıştır.



Şekil 7.39 Halka toplama kolu kapasitesinin 1 olduğu durum için yekpare gözetici ile elde edilen kapalı çevrim melez kontrol sistemi modeli



Şekil 7.40 Halka toplama kolu kapasitesinin 1 olduğu durum için yekpare gözetici ile elde edilen kapalı çevrim melez kontrol sistemine ait PLC merdiven diyagram kodu

Şekil 7.39'da görülen kapalı çevrim sistemdeki auto-net modelinde bulunan geçişler (ts1, ts2, …, ts23), CtlPN modelinde bulunan geçişlerle (t1, t2, …, t7) senkron çalışmaktadır. Auto-net modelinde bulunan geçişlerin, CtlPN modelindeki hangi geçişe senkron olduğu Çizelge 7.5'ten görülebilir.

Çizelge 7.5 Halka toplama kolu kapasitesinin 1 olduğu durum için hesaplanan yekpare gözeticide bulunan geçişlerin senkron olduğu CtlPN geçişleri

Auto-net geçiş adı	Otomata geçişi	Senkron olduğu CtlPN geçişi
ts1	[0 22 1]	t2
ts2	[1 12 0]	t1
ts3	[1 61 2]	t6
ts4	[2 32 3]	t3
ts5	[3 22 4]	t2
ts6	[3 71 5]	t7
ts7	[4 12 3]	t1
ts8	[4 71 6]	t7
ts9	[5 22 6]	t2
ts10	[5 42 7]	t4
ts11	[6 12 5]	t1
ts12	[6 42 8]	t4
ts13	[7 22 8]	t2
ts14	[7 52 0]	t5
ts15	[8 12 7]	t1
ts16	[8 52 1]	t5
ts17	[8 61 9]	t6
ts18	[9 32 10]	t3
ts19	[9 52 2]	t5
ts20	[10 22 11]	t2
ts21	[10 52 3]	t5
ts22	[11 12 10]	t1
ts23	[11 52 4]	t5

Şekil 7.40'ta görülen PLC merdiven diyagramı kodunun 1. ve 5. basamakları arasında (NW1-NW5) sistem modelinde bulunan kontrol edilemeyen geçişler gerçekleştirilmiştir. Bu gerçekleştirme işleminde önceki kısımlarda açıklanan jeton aktarma lojiği (Token Passing Logic) metodu kullanılmıştır. Bu metoda göre, Petri ağı modelindeki her bir mevki için bir hafıza biti atanır. Atanan hafıza bitinin set olması o mevkide jeton bulunması, reset olması ise o mevkide jeton bulunmaması anlamına gelir. Petri ağı modelinde bulunan bir jetonun; bir mevkiden diğer bir mevkiye geçişi, harici tetikleme koşulunun sağlanması ile gerçekleşir. PLC merdiven diyagramı kodunda jetonun ağdaki dolaşımı, geçişe atanmış ilgili harici tetikleme koşulunun sağlanması ile

giriş mevkisi için atanan hafıza bitinin reset, çıkış mevkisi için atanan hafıza bitinin de set edilmesi ile sağlanmaktadır. CtlPN modelinde bulunan geçişlere karşılık gelen olay işaretleri (e12, e22, e32, e42, e52, e61 ve e71), her bir geçişin tetiklenmesi durumunda ilgili set ve reset işlemlerinden önce normal bir çıkış bobini [--( )] kullanılarak ilgili hafıza bitine atama işlemi yapılarak elde edilmiştir. Örneğin, CtlPN modelinde bulunan t1 geçişi NW1 nolu basamakta gerçekleştirilmiştir. CtlPN modelinde, t1 geçişinin tetiklenebilmesi için, p2 ve p7 mevkilerinde jeton bulunması {p2 ve p7 etiketli hafıza bitlerinin set ('1') olması} ve $\chi_1 = \overline{I0.0}$ harici tetikleme koşulunun sağlanması gerekmektedir. Bu geçişin tetiklenmesi ile p2 mevkisinden jeton alınarak p1 mevkisine aktarılmaktadır. Yani, p2 etiketli hafıza biti resetlenip, p1 etiketli hafıza biti ise set edilmektedir. t1 geçişinin tetiklenmiş olduğunu bildiren "12" etiketli olayı simgelemek üzere kullanılan e12 hafıza bitine normal çıkış bobini [--(e12)] ile atama yapılmıştır. Bu atama ile t1 geçişinin tetiklenmesi sırasında e12 hafıza biti çıkış bobini yardımıyla sadece bir PLC tarama süresince aktif olmakta ve tetikleme koşulunun kalkması ile bir sonraki PLC program taramasıyla beraber aktif olmaktan çıkmaktadır

NW6 ile NW28 arasındaki basamaklarda gözeticiye ait auto-net modelinde bulunan geçişler gerçekleştirilmiştir. CtlPN modelinde bulunan kontrol edilebilir geçişler t6 ve t7 ise sırasıyla NW29 ve NW30'da gerçekleştirilmiştir. CtlPN modelinde bulunan kontrol mevkileri, bu kontrol edilebilir geçişlere yetkileme okları ile bağlıdır. Örneğin, t6 geçişinin tetiklenebilmesi için c61 kontrol mevkisinde ve p7 mevkisinde jeton bulunması gerekmektedir. Yani ilgili mevkileri karakterize eden hafıza bitlerinin set olması gerekmektedir. t6 geçişinin tetiklenmesi ile p7 mevkisindeki jeton alınarak, p8 mevkisine jeton aktarılmaktadır. c61 kontrol mevkisindeki jeton ise bu mevki, geçişe yetkileme oku ile bağlandığından alınmamaktadır. Bu işlemler set ve reset komutları kullanılarak Şekil 7.40'ta görüldüğü gibi gerçekleştirilmiştir. Son olarak NW 31 ve NW32 basamaklarında çıkış aksiyonları gerçekleştirilmiştir.

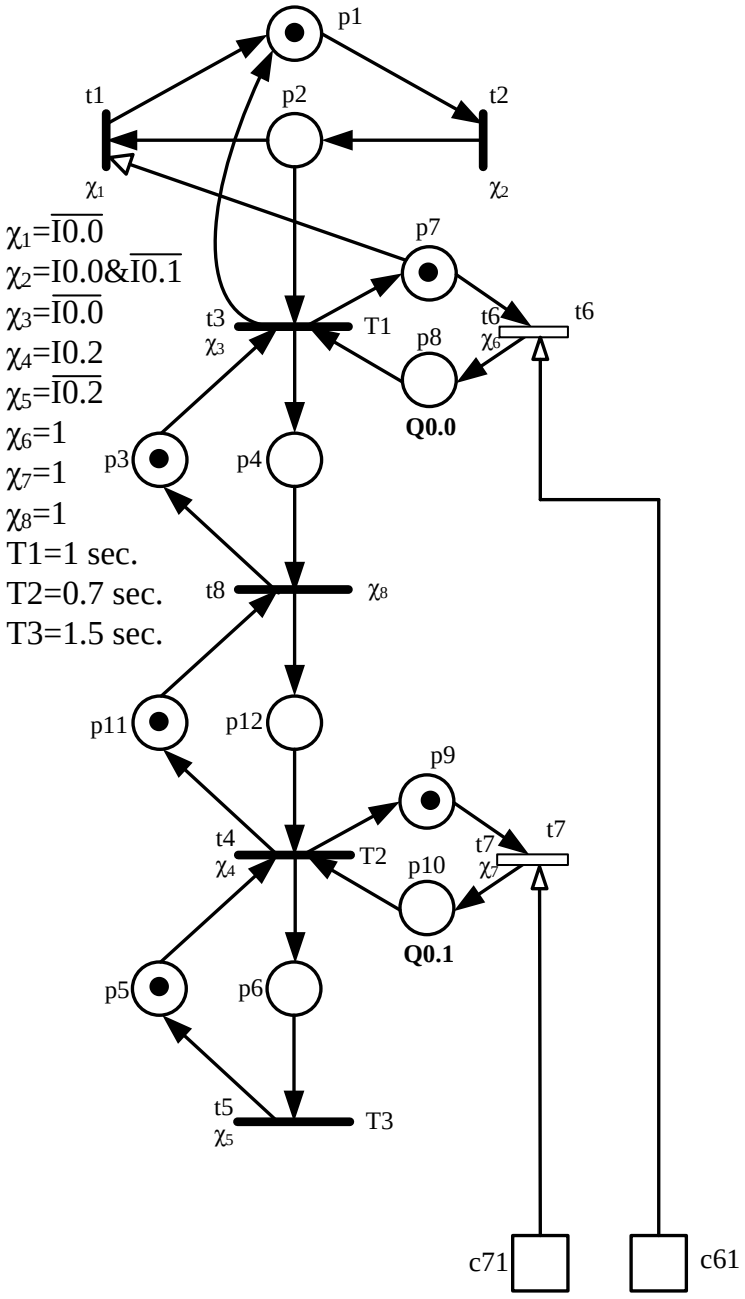
Şekil 7.40'ta görülen PLC merdiven diyagramı kodu Siemens S7-300 PLC'ye yüklenerek, sistemin spesifikasyonlarda belirtilen çalışmayı başarılı bir şekilde gerçekleştirdiği gözlemlenmiştir.

7.4.1.2 Halka toplama kolu kapasitesinin 2 olduğu durum

Deneysel endüstriyel üretim sistemin halka kolu kapasitesinin 2 olduğu durum için kontrol edilmemiş CtlPN modeli Şekil 7.41'de görülmektedir. Bu model Şekil 7.37'de görülen modele p11 ve p12 mevkilerinin eklenmesi ile elde edilmiştir. Kontrol spesifikasyonları, halka toplama kolu kapasitesinin 1 olduğu duruma benzer olarak aşağıdaki gibidir:

1. Seçme selenoidi (itici selenoid), seçme alanında bir halka bulunduğunda ve halka toplama kolunda en az bir boş yer olduğunda çalışacaktır (SPEC1).
2. Birleştirme selenoidi (döner selenoid) sadece halka toplama kolunda en az bir halka olduğunda ve birleştirme noktası boş olduğunda çalışacaktır (SPEC2).

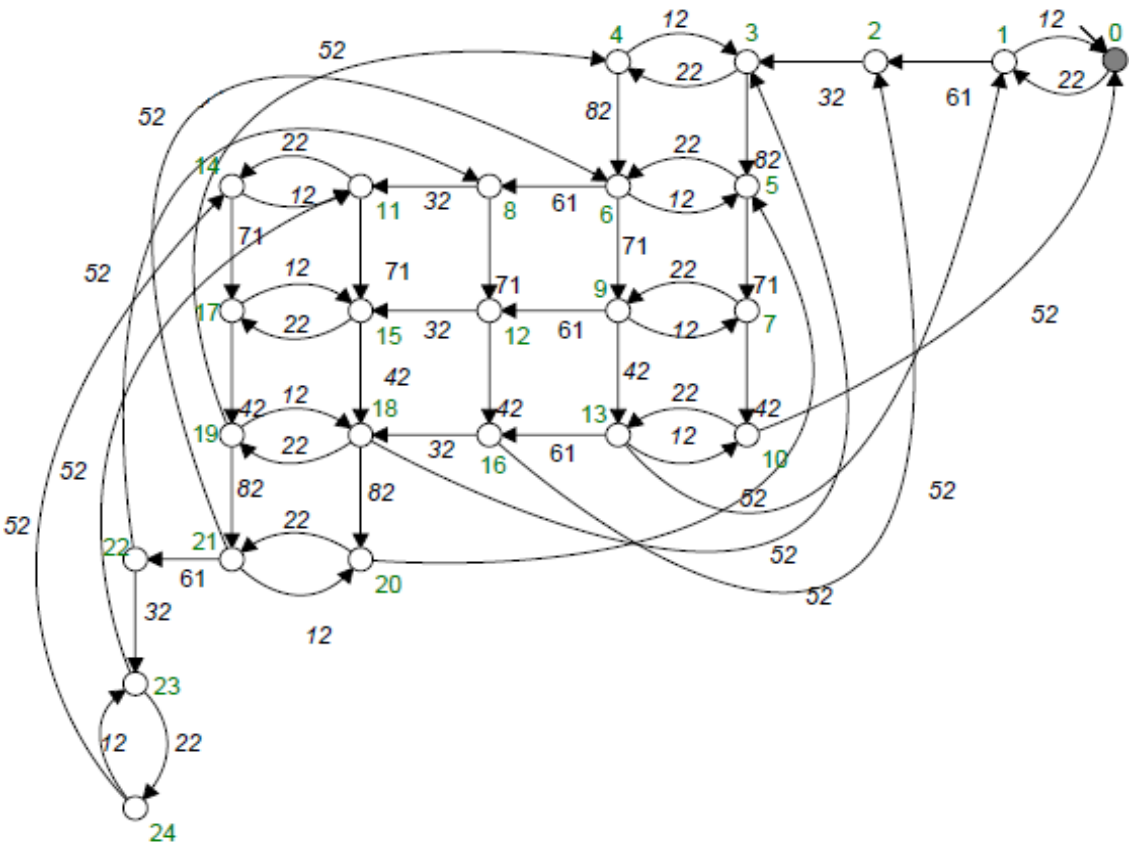
Bu spesifikasyonları dikkate alarak halka toplama kolu kapasitesinin 2 olduğu durum için elde edilen yekpare gözeticinin otomata modeli Şekil 7.42.(a)'da, auto-net modeli ise Şekil 7.42.(b)'de görülmektedir. "cp.exe" programı ile bu örnek için hesaplanan kontrol poliçesinin detayları Ek-E'de program çıktısı olarak sunulmuştur. Halka toplama kolu kapasitesinin iki halka olduğu durum için hesaplanan kontrol poliçesi Çizelge 7.6'da görülmektedir. Hesaplanan bu kontrol poliçesinin gösterdiği sıradan ok bağlantıları ışığında, halka toplama kolu kapasitesinin 2 olduğu durum için elde edilen kapalı çevrim kontrol sistemi Şekil 7.43'te görülmektedir.



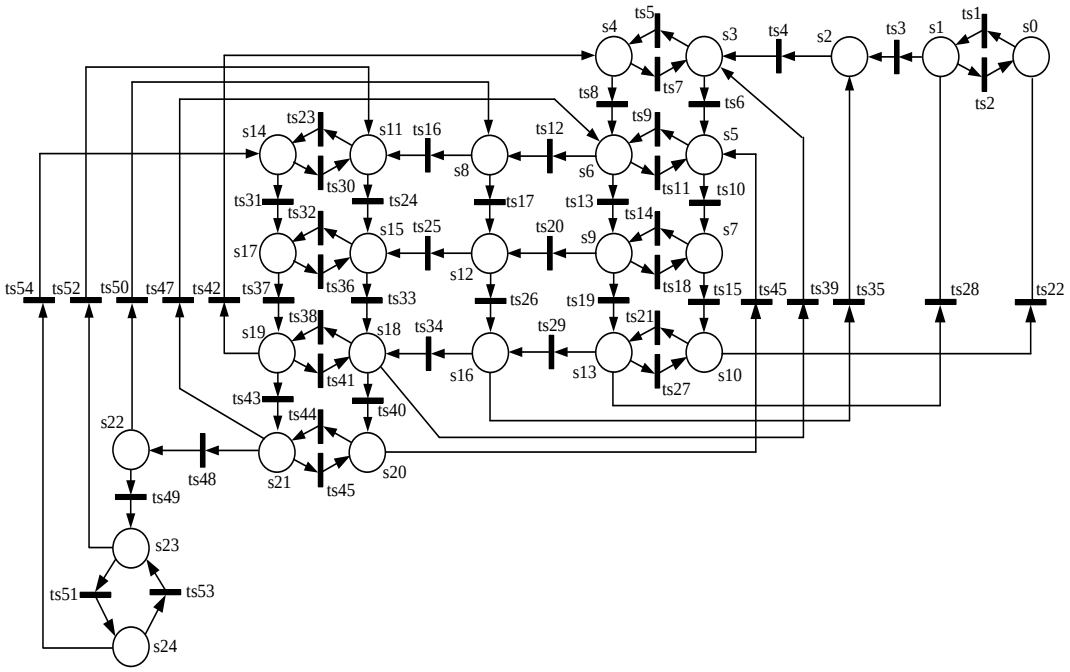
Şekil 7.41 Halka toplama kolu kapasitesinin 2 olduğu durum için CtlPN modeli

Çizelge 7.6 Halka toplama kolu kapasitesinin 2 olduğu durum için hesaplanan kontrol poliçesi

Kontrol mevkisi	Başlangıç jeton sayısı	Pre oklarının bağlanacağı geçişler	Post oklarının bağlanacağı geçişler
c61	0	ts1 ts8 ts9 ts14 ts21 ts43 ts44	ts2 ts4 ts11 ts16 ts18 ts25 ts27 ts34 ts46 ts49
c71	0	ts6 ts8 ts45 ts47 ts50 ts52 ts54	ts15 ts19 ts26 ts33 ts37

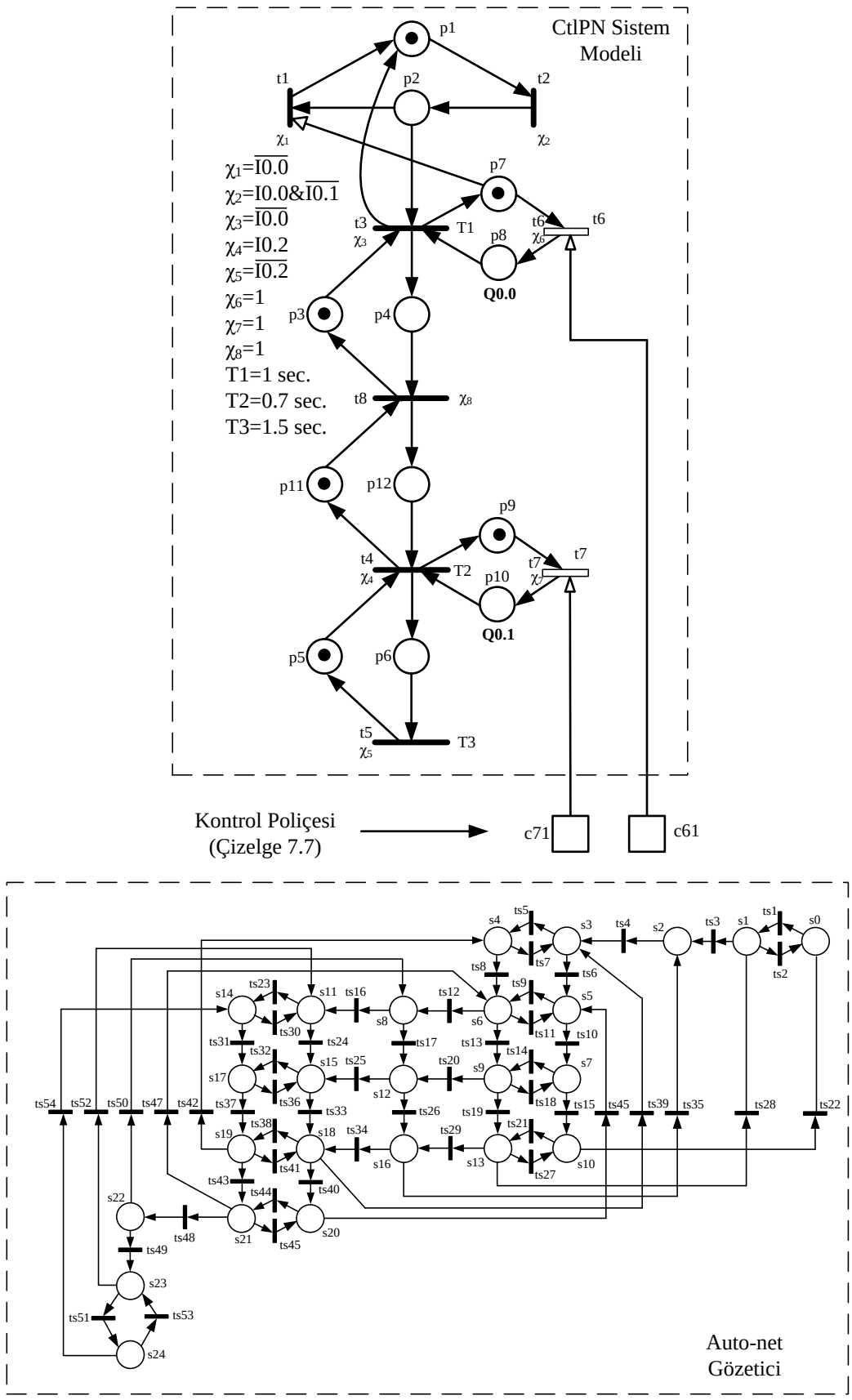


a)



b)

Şekil 7.42 a) Halka toplama kolu kapasitesinin 2 olduğu durum için elde edilen gözeticinin otomata modeli, b) auto-net modeli

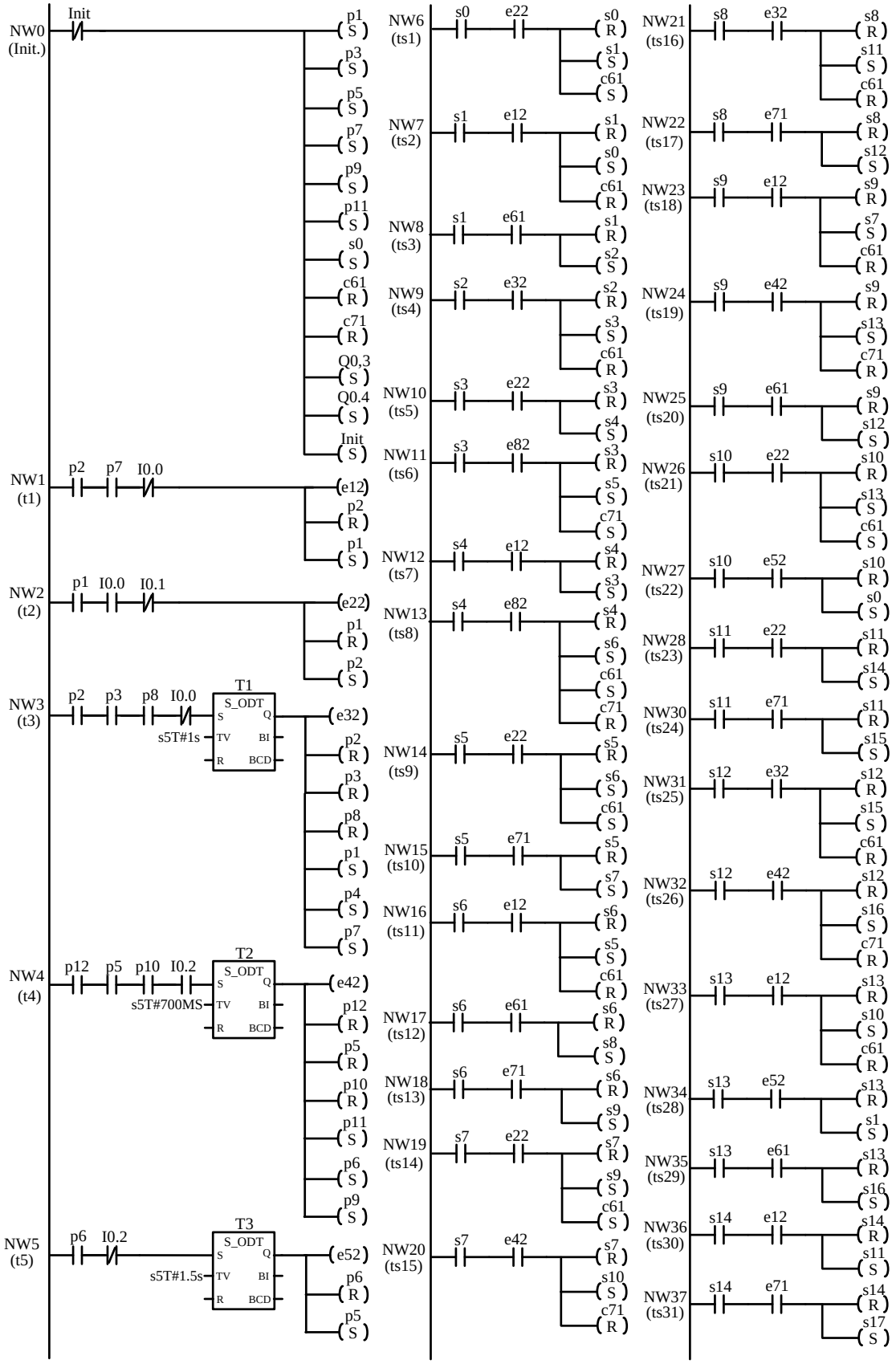


Şekil 7.43 Halka toplama kolu kapasitesinin 2 olduğu durum için yekpare gözetici ile elde edilen kapalı çevrim melez kontrol sistemi modeli

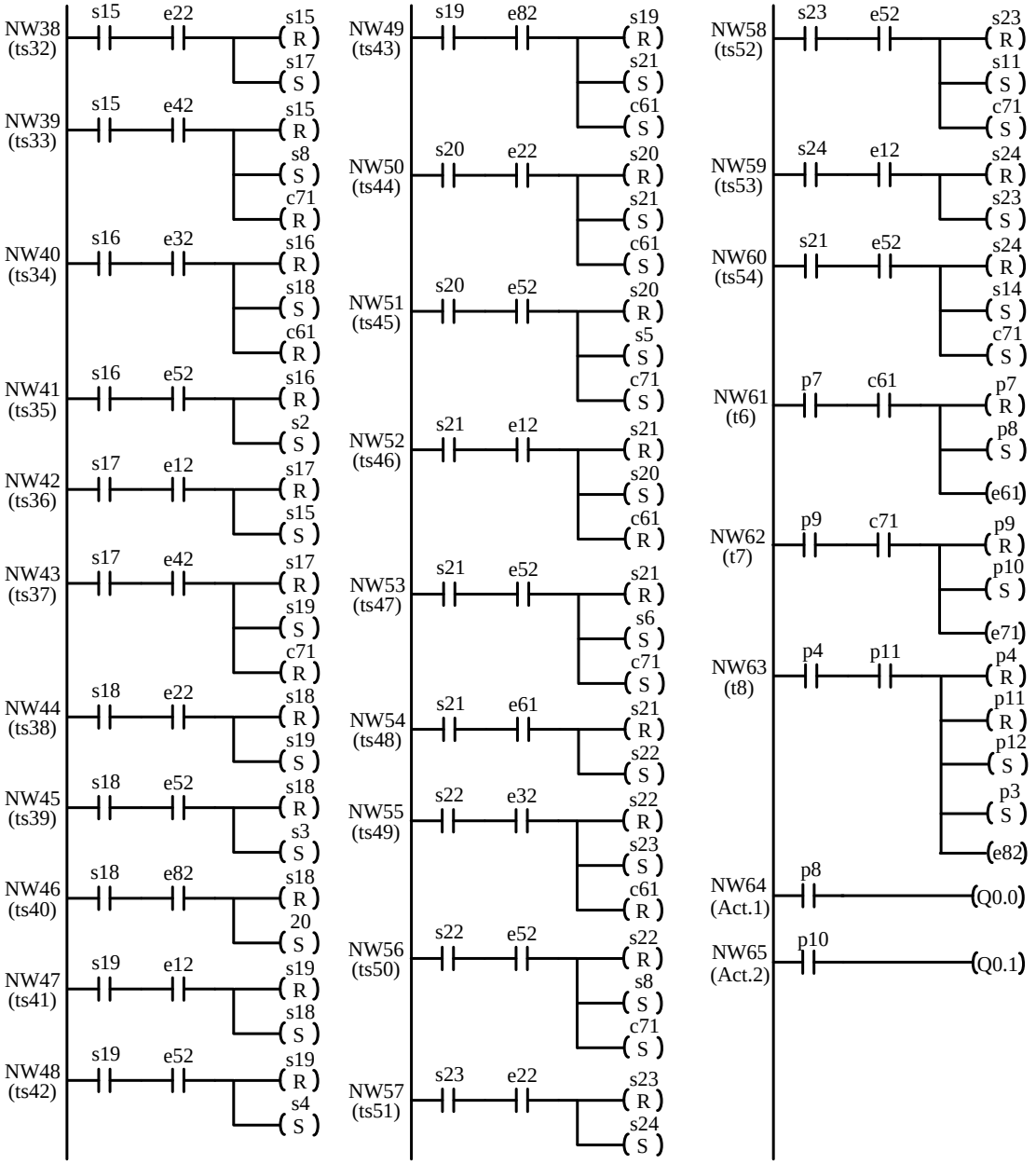
Şekil 7.43'te görülen halka toplama kolu kapasitesinin 2 olduğu durum için elde edilen kapalı çevrim melez kontrol sistemine ait merdiven diyagramı kodu Şekil 7.44'te verilmiştir. Şekil 7.44'ten görüleceği üzere kodun ilk basamağında (NW0) başlangıç jeton işaretlemeleri gerçekleştirilmiştir. PLC merdiven diyagramı kodunun NW1 ile NW5 arasında CtlPN modelindeki kontrol edilemeyen geçişler t1, t2, t3, t4 ve t5 sırasıyla gerçekleştirilmiştir. NW6 ile NW60 arasındaki basamaklarda gözetici set ve reset komutları yardımıyla gerçekleştirilmiştir. NW61, NW62 ve MW63 basamaklarında sırası ile t6, t7 ve t8 geçişleri gerçekleştirilmiştir. Son olarak NW64 ve NW65 basamaklarında çıkış aksiyonları gerçekleştirilmiştir. Şekil 7.44'te görülen PLC merdiven diyagramı kodu Siemens S7-300 PLC'ye yüklenerek, sistemin spesifikasyonlarda belirtilen çalışmayı başarılı bir şekilde gerçekleştirdiği gözlemlenmiştir.

Çizelge 7.7 Halka toplama kolu kapasitesinin 2 olduğu durum için hesaplanan yekpare gözeticide bulunan geçişlerin senkron olduğu CtlPN geçişleri

Auto-net geçiş adı	Otomata geçişi	Senkron olduğu CtlPN geçişi	Auto-net geçiş adı	Otomata geçişi	Senkron olduğu CtlPN geçişi
ts1	[0 22 1]	t2	ts28	[13 52 1]	t5
ts2	[1 12 0]	t1	ts29	[13 61 16]	t6
ts3	[1 61 2]	t6	ts30	[14 12 11]	t1
ts4	[2 32 3]	t3	ts31	[14 71 17]	t7
ts5	[3 22 4]	t2	ts32	[15 22 17]	t2
ts6	[3 82 5]	t8	ts33	[15 42 18]	t4
ts7	[4 12 3]	t1	ts34	[16 32 18]	t3
ts8	[4 82 6]	t8	ts35	[16 52 2]	t5
ts9	[5 22 6]	t2	ts36	[17 12 15]	t1
ts10	[5 71 7]	t7	ts37	[17 42 19]	t4
ts11	[6 12 5]	t1	ts38	[18 22 19]	t2
ts12	[6 61 8]	t6	ts39	[18 52 3]	t5
ts13	[6 71 9]	t7	ts40	[18 82 20]	t8
ts14	[7 22 9]	t2	ts41	[19 12 18]	t1
ts15	[7 42 10]	t4	ts42	[19 52 4]	t5
ts16	[8 32 11]	t3	ts43	[19 82 21]	t8
ts17	[8 71 12]	t7	ts44	[20 22 21]	t2
ts18	[9 12 7]	t1	ts45	[20 52 5]	t5
ts19	[9 42 13]	t4	ts46	[21 12 20]	t1
ts20	[9 61 12]	t6	ts47	[21 52 6]	t5
ts21	[10 22 13]	t2	ts48	[21 61 22]	t6
ts22	[10 52 0]	t5	ts49	[22 32 23]	t3
ts23	[11 22 14]	t2	ts50	[22 52 8]	t5
ts24	[11 71 15]	t7	ts51	[23 22 24]	t2
ts25	[12 32 15]	t3	ts52	[23 52 11]	t5
ts26	[12 42 16]	t4	ts53	[24 12 23]	t1
ts27	[13 12 10]	t1	ts54	[24 52 14]	t5



Şekil 7.44 Halka toplama kolu kapasitesinin 2 olduğu durum için yekpare gözetici ile elde edilen kapalı çevrim melez kontrol sistemine ait PLC merdiven diyagramı kodu



Şekil 7.44 Halka toplama kolu kapasitesinin 2 olduğu durum için yekpare gözetici ile elde edilen kapalı çevrim melez kontrol sistemine ait PLC merdiven diyagramı kodu (Devam)

Şekil 7.43'te görülen kapalı çevrim sistemdeki auto-net modelinde bulunan geçişler (ts1, ts2, …, ts54), CtlPN modelinde bulunan geçişlerle (t1, t2, …, t8) senkron çalışmaktadır. Auto-net modelinde bulunan geçişlerin, CtlPN modelindeki hangi geçişlerle senkron olduğu Çizelge 7.7'de görülebilir.

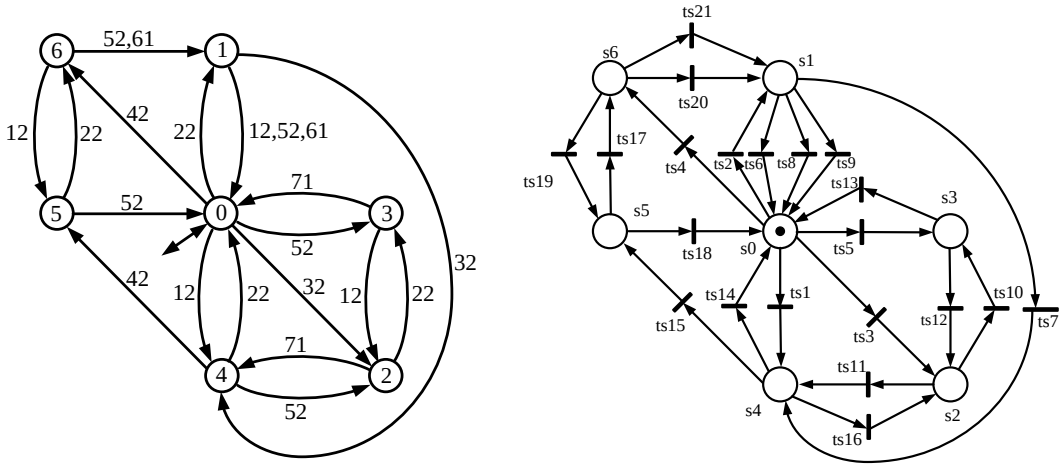
7.4.2 Kontrollü Petri ağı için indirgenmiş yekpare gözetici tasarımı ve PLC ile gerçekleştirilmesi

7.4.2.1 Halka toplama kolu kapasitesinin 1 olduğu durum

Halka toplama kolu kapasitesinin 1 olduğu durum için elde edilmiş olan yekpare gözetici Şekil 7.38'de görülmektedir. Bu gözeticinin TCT yazılımında bulunan *Supreduce(.)* komutu yardımıyla durum sayısının azaltılmasıyla indirgenmiş gözetici elde edilmektedir. İndirgenmiş gözetici aşağıdaki TCT işlemleri yardımıyla elde edilmiştir.

SUPER = Supcon(PLANT,SPEC) (12,23)
CDAT = Condat(PLANT,SUPER) Controllable.
RED_SUPER = Supreduce(PLANT,SUPER,CDAT) (7,21;slb=7)
RED_CDAT = Condat(PLANT,RED_SUPER) Controllable.

Bu işlemler sonucunda elde edilen indirgenmiş gözetici 7 durum ve 21 geçişten oluşmaktadır. Halka toplama kolu kapasitesinin 1 olduğu durum için hesaplanan indirgenmiş gözeticinin otomata modeli Şekil 7.45.(a)'da, bu otomata modeli için elde edilen auto-net modeli ise Şekil 7.45.(b)'de görülmektedir. Aynı kontrol işlemini sağlayan gözeticinin Şekil 7.39'da görülen yekpare (indirgenmemiş) modeli 12 durum ve 23 geçişten oluşmaktadır. Buradan da görülebileceği indirgenmiş gözetici daha az durum ve geçiş sayısına sahiptir.



a) b)

Şekil 7.45 a) Halka toplama kolu kapasitesinin 1 olduğu durum için elde edilen indirgenmiş gözeticinin otomata modeli, b) Auto-net modeli

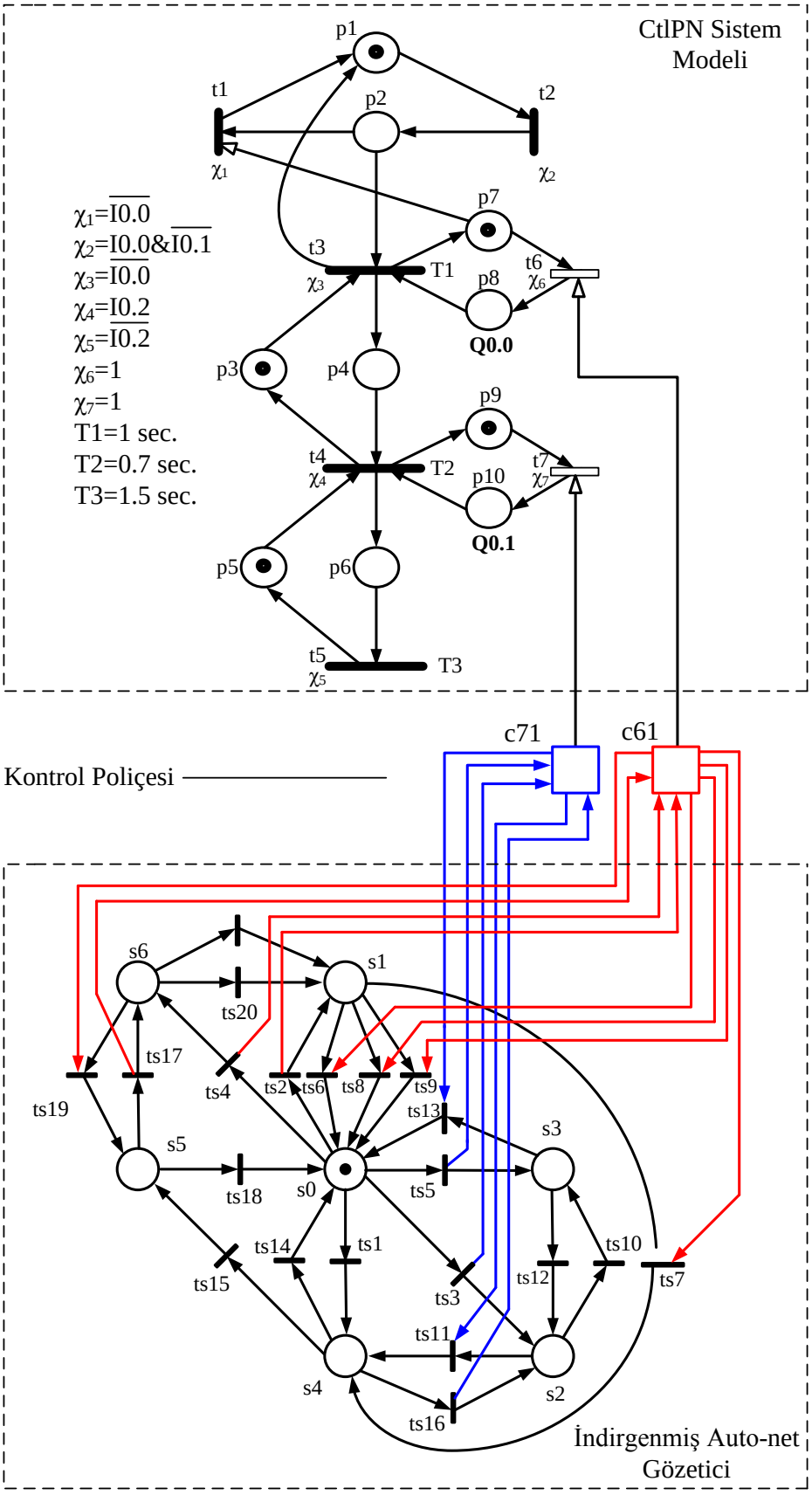
Şekil 7.45'te görülen gözetici dikkate alınarak "cp.exe" programı ile hesaplanan kontrol poliçesinin detayları Ek-E'de görülmektedir. Bu gözetici için hesaplanan kontrol poliçesindeki okların bağlanacağı geçiş isimleri ve kontrol mevkilerinde başlangıçta bulunacak jeton sayısı Çizelge 7.8'de görülmektedir.

Çizelge 7.8 Halka toplama kolu kapasitesinin 1 olduğu durum için hesaplanan indirgenmiş gözeticiye ait kontrol poliçesi

Kontrol mevkisi	Başlangıç jeton sayısı	Pre oklarının bağlanacağı geçişler	Post oklarının bağlanacağı geçişler
c61	0	ts2 ts4 ts17	ts6 ts7 ts8 ts9 ts19
c71	0	ts3 ts5 ts16	ts11 ts13

Hesaplanan bu kontrol poliçesinin gösterdiği sıradan ok bağlantıları ışığında, halka toplama kolu kapasitesinin 1 olduğu durum için indirgenmiş yekpare gözetici ile elde edilen kapalı çevrim kontrol sistemi Şekil 7.46'da görülmektedir.

Şekil 7.46'da görülen kapalı çevrim sistemdeki auto-net modelinde bulunan geçişler (ts1, ts2, …, ts21), CtlPN modelinde bulunan geçişlerle (t1, t2, …, t7) senkron çalışmaktadır. Auto-net modelinde bulunan geçişlerin, CtlPN modelindeki hangi geçişlerle senkron olduğu Çizelge 7.9'da görülmektedir.



Şekil 7.46 Halka toplama kolu kapasitesinin 1 olduğu durum için indirgenmiş yekpare gözetici ile elde edilen kapalı çevrim melez kontrol sistemi modeli

Çizelge 7.9 Halka toplama kolu kapasitesinin 1 halka olduğu durum için hesaplanan indirgenmiş yekpare gözeticide bulunan geçişlerin senkron olduğu CtlPN geçişleri

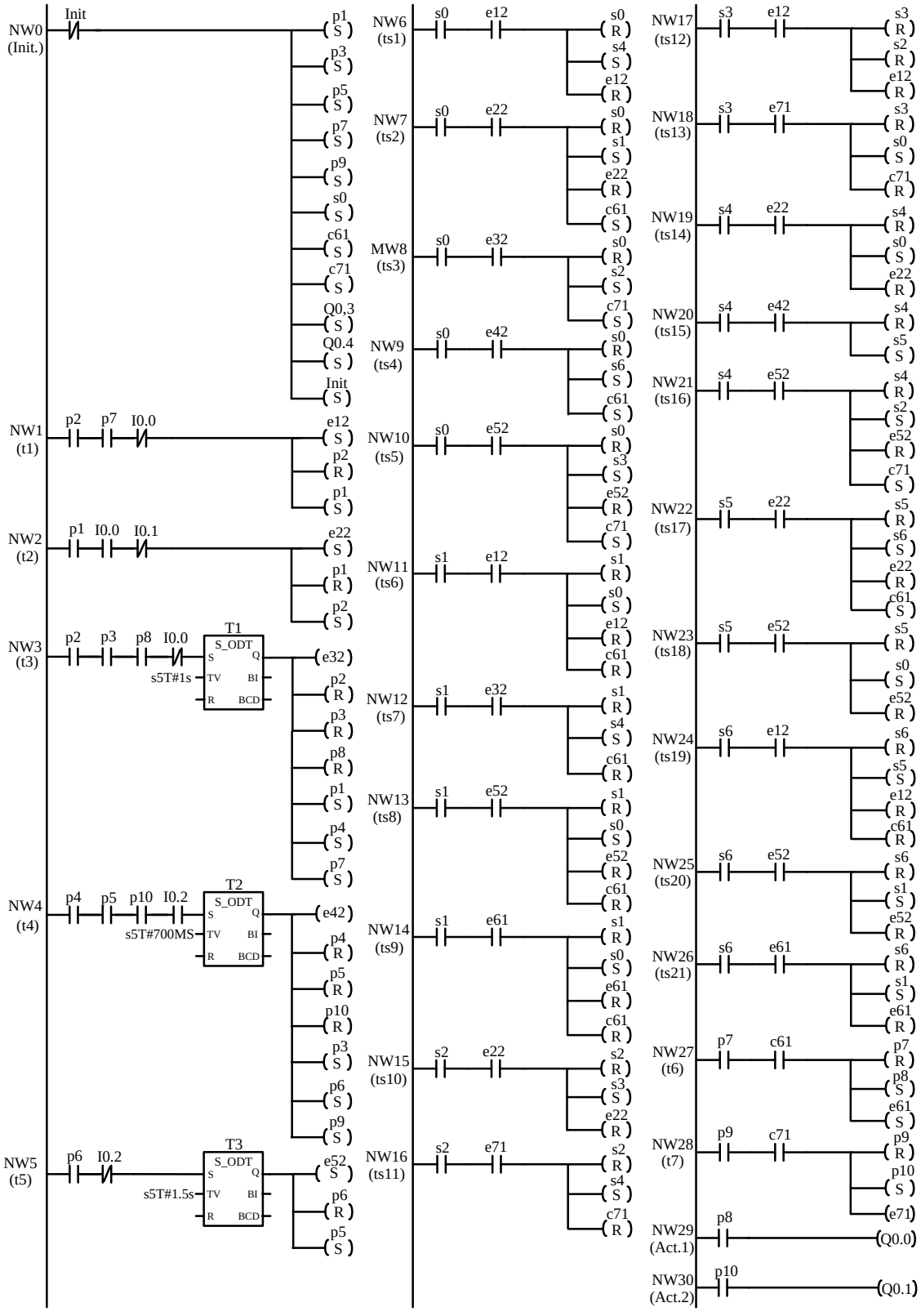
Auto-net geçiş adı	Otomata geçişi	Senkron olduğu CtlPN geçişi
ts1	[0 12 4]	t1
ts2	[0 22 1]	t2
ts3	[0 32 2]	t3
ts4	[0 42 6]	t4
ts5	[0 52 3]	t5
ts6	[1 12 0]	t1
ts7	[1 32 4]	t3
ts8	[1 52 0]	t5
ts9	[1 61 0]	t6
ts10	[2 22 3]	t2
ts11	[2 71 4]	t7
ts12	[3 12 2]	t1
ts13	[3 71 0]	t7
ts14	[4 22 0]	t2
ts15	[4 42 5]	t4
ts16	[4 52 2]	t5
ts17	[5 22 6]	t2
ts18	[5 52 0]	t5
ts19	[6 12 5]	t1
ts20	[6 52 1]	t5
ts21	[6 61 1]	t6

Şekil 7.45.(a)'da görülen otomata modeli detaylı olarak incelenecek olursa bu modelin PLC ile gerçekleştirilmesi esnasında çığ etkisi (avalanche effect) problemi ile karşılaşıldığı görülmektedir. Çığ etkisi, otomata modellerinin PLC ile gerçekleştirilirken programın aynı PLC tarama döngüsü içerisinde rastgele sayıda duruma sıçramasıdır. Bu problem yazılan kodun istenildiği gibi çalışmasını engelleyen büyük bir sorundur. Bu problemin çözümüne yönelik etkin bir yöntem [66]'da önerilmiş ve detayları önceki kısımlarda açıklanmıştır. Önerilen bu yönteme göre ilk olarak hangi olayların çığ etkisine neden olduğu belirlenmelidir. Bu işlem detayları dördüncü bölümünde açıklanan "Av_effect_detector.exe" programı ile kolayca yapılabilmektedir. Otomata modeline ait .ADS dosyasının giriş olarak kullanılmasıyla "Av_effect_detector.exe" programı hangi olayların çığ etkisine sebep olduğunu hesaplamaktadır. Şekil 7.45.(a)'da görülen otomata modeli için "Av_effect_detector.exe" programı ile yapılan hesaplama sonucunda "12, 22, 52 ve 61" etiketli olayların çığ etkisine sebep olduğu belirlenmiştir.

PLC merdiven diyagramı kodu gerçekleştirmesi yapılırken sistemin CtlPN modelindeki her bir geçişin gerçekleştirildiği basamakta ilgili olaya ait hafıza bitinin bir tarama süresince set edilmesi için çıkış bobini kullanılarak atama yapılmaktadır. Çığ etkisini ortadan kaldırmak için, çığ etkisine sebep olan olaylar için bu atama işlemi çıkış bobini [--( )] kullanılarak değil set komutu [--(S)] ile yapılır. Bu atama işlemi ile olay oluşumu algılanmaktadır. Olayın oluşumu ile set edilen hafıza biti, gözeticinin gerçekleştirildiği basamaklardan hangi basamakta kullanılmışsa o basamakta reset komutu kullanılarak resetlenir. Böylelikle bir tarama döngüsünde olayın oluşumu yalnızca bir geçişe izin vermiş olacaktır.

Şekil 7.46'da görülen kapalı çevrim sistemi gerçekleştiren PLC merdiven diyagramı kodu Şekil 7.47'de görülmektedir. Bu kodda ilk basamakta diğer kodlarda olduğu gibi başlangıç işlemleri gerçekleştirilmektedir. NW1 ile NW5 arasındaki basamaklarda sistemin CtlPN modeli, NW6 ile NW26 arasındaki basamaklarda gözeticinin geçişleri, NW27 ile NW28'de kontrol mevkileri ve son olarak NW29 ile NW30 basamaklarında çıkış aksiyonları gerçekleştirilmiştir. Sistemin CtlPN modelinin gerçekleştirildiği basamaklardan çığ etkisine sebep olan olayların bulunduğu basamaklarda (t1, t2, t5 ve t6'nın gerçekleştirildiği basamaklar) olay oluşumunu simgeleyen hafıza biti set komutu [--(S)] ile set edilmiştir. Gözetici gerçekleştirilirken bu bitin kullanıldığı basamakta, ilgili bit reset komutu [--(R)] ile resetlenmiştir. Örneğin, 12 etiketine sahip olay çığ etkisine sebep olmaktadır. Bu olayın oluşumu sistemin CtlPN modelinde bulunan t1 geçişinin tetiklenmesi ile ilişkilidir. Bu nedenle t1 geçişinin gerçekleştirildiği basamak NW1'de 12 etiketine sahip olayı simgeleyen hafıza biti e12 set edilmektedir. Gözeticide bulunan ts1 geçişi [s0 e12 s4] NW6 basamağında gerçekleştirilirken ilgili işlemlerin ardından reset komutu ile e12 biti resetlenmektedir. Çığ etkisine sebep olan olayların tümü benzer şekilde gerçekleştirilmektedir. Çığ etkisine sebep olmayan olaylar diğer örneklerde olduğu gibi gerçekleştirilmektedir.

Şekil 7.47'de görülen PLC merdiven diyagramı kodu Siemens S7-300 PLC'ye yüklenerek, sistemin spesifikasyonlarda belirtilen çalışmayı başarılı bir şekilde gerçekleştirdiği gözlemlenmiştir.



Şekil 7.47 Halka toplama kolu kapasitesinin 1 olduğu durum için elde edilmiş olan indirgenmiş yekpare gözeticili kapalı çevrim melez kontrol sistemine ait merdiven diyagramı kodu

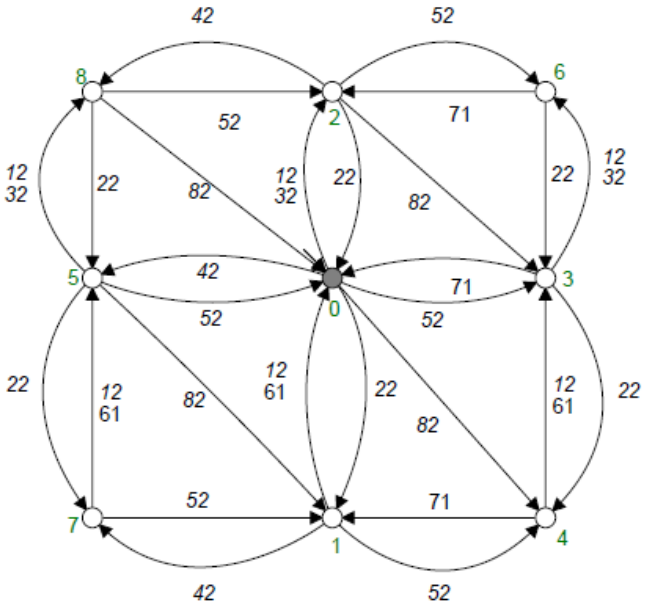
7.4.2.2 Halka toplama kolu kapasitesinin 2 olduğu durum

Halka toplama kolu kapasitesinin 2 olduğu durum için hesaplanan indirgenmiş yekpare gözeticinin otomata modeli Şekil 7.48.(a)'da, auto-net modeli ise Şekil 7.48.(b)'de görülmektedir. Bu gözetici için "cp.exe" programı ile hesaplanan kontrol poliçesinin detayları Ek-G'de görülmektedir. Hesaplanan kontrol poliçesindeki okların bağlanacağı geçiş isimleri ve kontrol mevkilerinde başlangıçta bulunacak jeton sayısı Çizelge 7.10'da görülmektedir.

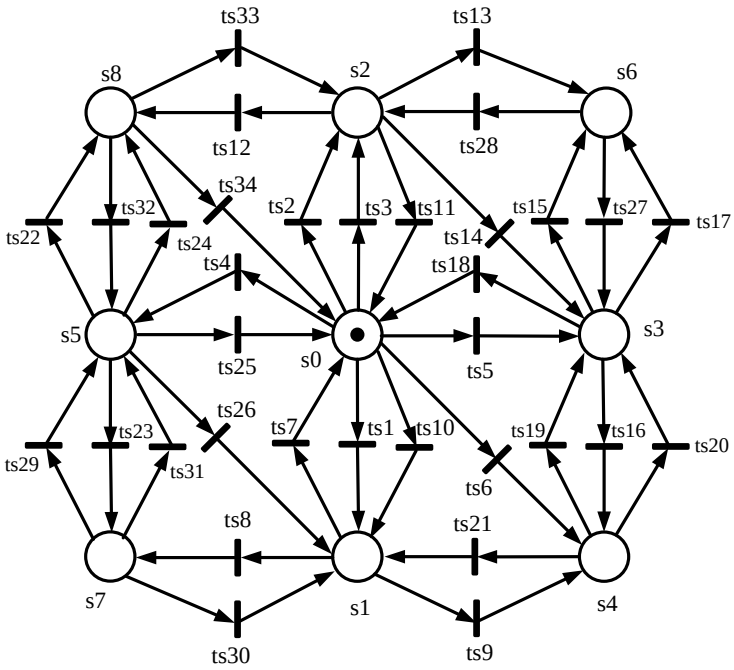
Çizelge 7.10 Halka toplama kolu kapasitesinin 2 olduğu durum için hesaplanan indirgenmiş gözeticiye ait kontrol poliçesi

Kontrol mevkisi	Başlangıç jeton sayısı	Pre oklarının bağlanacağı geçişler	Post oklarının bağlanacağı geçişler
c61	0	ts2 ts6 ts16 ts23 ts26	ts7 ts10 ts19 ts20 ts29 ts31
c71	0	ts5 ts6 ts9 ts13 ts14	ts18 ts21 ts28

Hesaplanan bu kontrol poliçesinin gösterdiği sıradan ok bağlantıları ışığında, halka toplama kolu kapasitesinin 2 olduğu durum için elde edilen kapalı çevrim kontrol sistemi Şekil 7.49'da görülmektedir. Şekil 7.48'de görülen otomata modelinde "Av_effect_detector.exe" programı ile yapılan hesaplama sonucunda "12, 22, 52 ve 82" etiketli olayların çığ etkisine sebep olduğu belirlenmiştir. Şekil 7.49'da görülen kapalı çevrim sisteminden elde edilen PLC merdiven diyagramı kodu Şekil 7.50'de görülmektedir. Bu örnek için de, bir önceki örnekte olduğu gibi çığ etkisine sebep olan olaylar dikkate alınarak kod düzenlenmiştir. Şekil 7.50'de görülen PLC merdiven diyagramı kodu Siemens S7-300 PLC'ye yüklenerek, sistemin spesifikasyonlarda belirtilen çalışmayı başarılı bir şekilde gerçekleştirdiği gözlemlenmiştir



a)



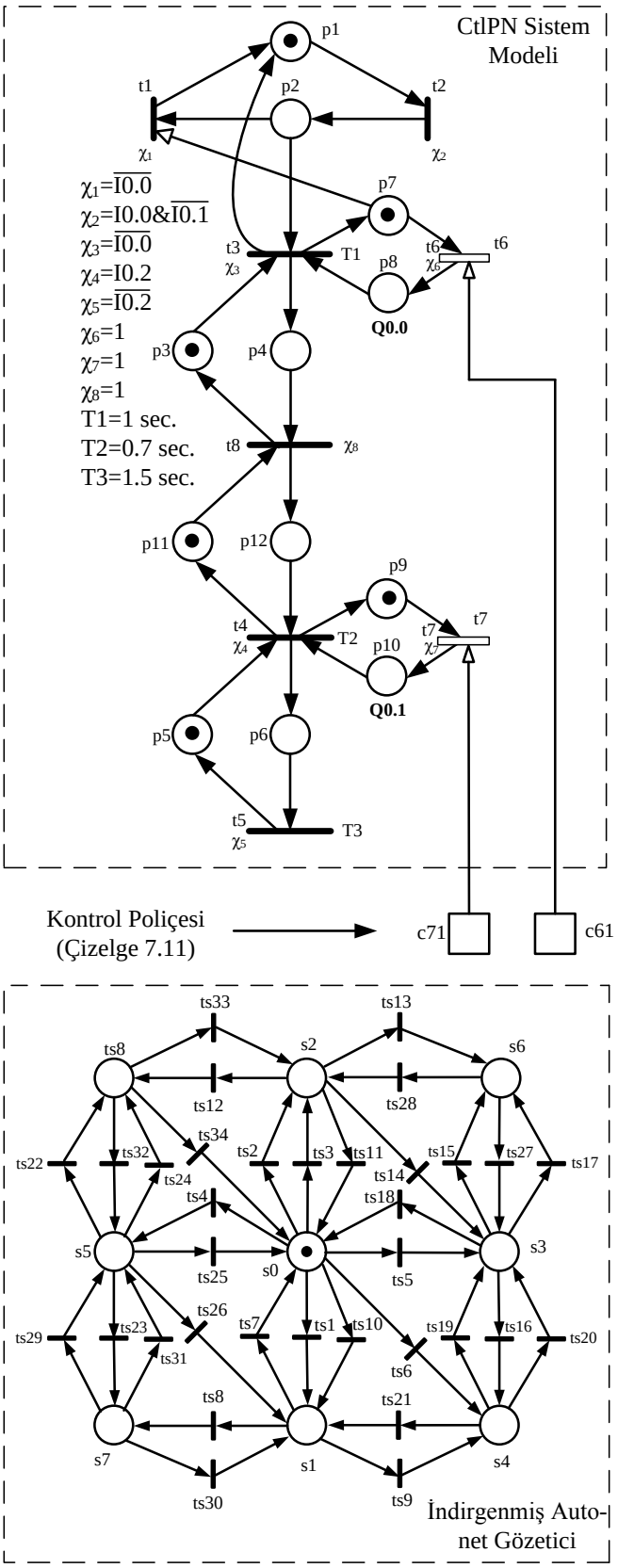
b)

Şekil 7.48 a) Halka toplama kolu kapasitesinin 2 olduğu durum için hesaplanan indirgenmiş gözeticinin otomata modeli, b) Auto-net modeli

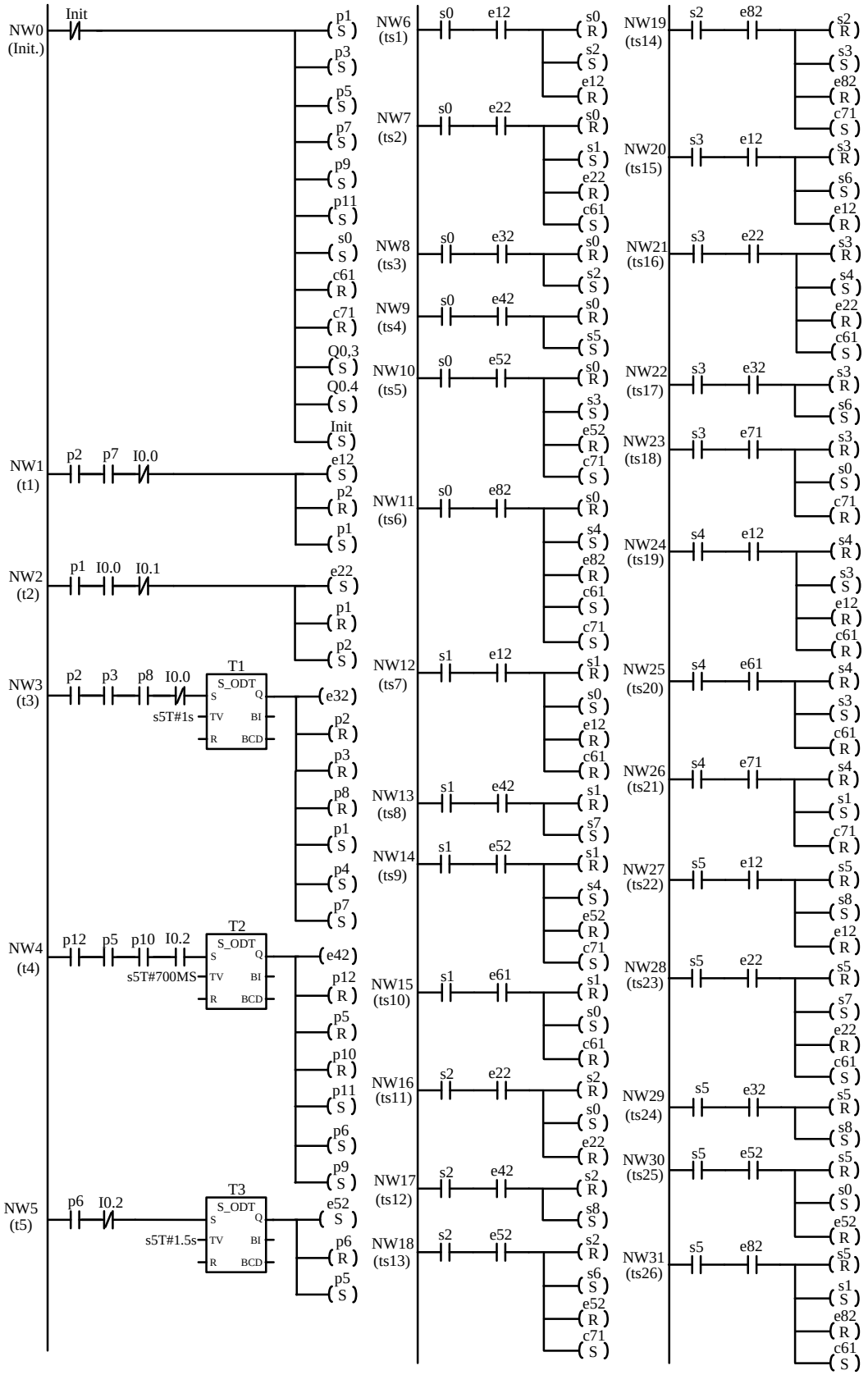
Şekil 7.49'da görülen kapalı çevrim sistemdeki auto-net modelinde bulunan geçişler (ts1, ts2, …, ts34), CtlPN modelinde bulunan geçişlerle (t1, t2, …, t8) senkron çalışmaktadır. Auto-net modelinde bulunan geçişlerin, CtlPN modelindeki hangi geçişlere senkron olduğu Çizelge 7.11'den görülebilir.

Çizelge 7.11 Halka toplama kolu kapasitesinin 2 olduğu durum için hesaplanan yekpare gözeticide bulunan geçişlerin senkron olduğu CtlPN geçişleri

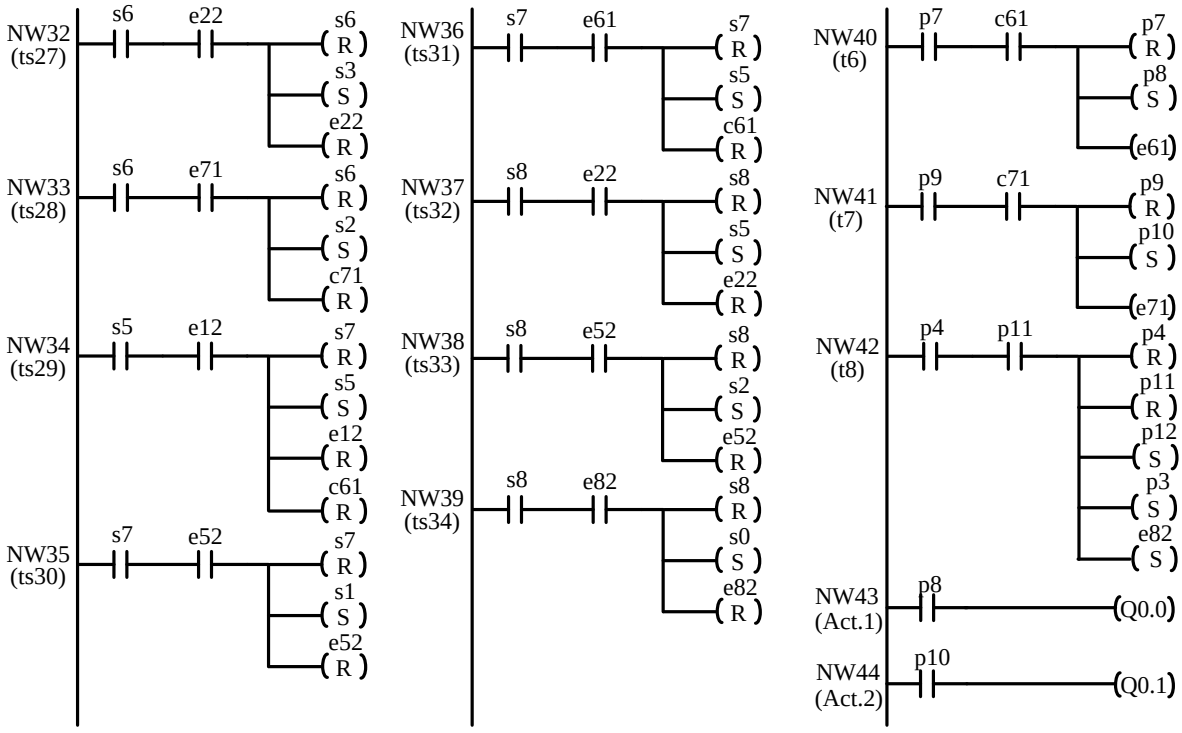
Auto-net geçiş adı	Otomata geçişi	Senkron olduğu CtlPN geçişi	Auto-net geçiş adı	Otomata geçişi	Senkron olduğu CtlPN geçişi
ts1	[0 12 2]	t1	ts18	[3 71 0]	t7
ts2	[0 22 1]	t2	ts19	[4 12 3]	t1
ts3	[0 32 2]	t3	ts20	[4 61 3]	t6
ts4	[0 42 5]	t4	ts21	[4 71 1]	t7
ts5	[0 52 3]	t5	ts22	[5 12 8]	t1
ts6	[0 82 4]	t8	ts23	[5 22 7]	t2
ts7	[1 12 0]	t1	ts24	[5 32 8]	t3
ts8	[1 42 7]	t4	ts25	[5 52 0]	t5
ts9	[1 52 4]	t5	ts26	[5 82 1]	t8
ts10	[1 61 0]	t6	ts27	[6 22 3]	t2
ts11	[2 22 0]	t2	ts28	[6 71 2]	t7
ts12	[2 42 8]	t4	ts29	[7 12 5]	t1
ts13	[2 52 6]	t5	ts30	[7 52 1]	r5
ts14	[2 82 3]	t8	ts31	[7 61 5]	t6
ts15	[3 12 6]	t1	ts32	[8 22 5]	t2
ts16	[3 22 4]	t2	ts33	[8 52 2]	t5
ts17	[3 32 6]	t3	ts34	[8 82 0]	t8



Şekil 7.49 Halka toplama kolu kapasitesinin 2 olduğu durum için elde edilmiş olan indirgenmiş yekpare gözeticili kapalı çevrim melez kontrol sistemi modeli



Şekil 7.50 Halka toplama kolu kapasitesinin 2 olduğu durum için elde edilmiş olan indirgenmiş yekpare gözeticili kapalı çevrim melez kontrol sistemine ait merdiven diyagramı kodu



Şekil 7.50 Halka toplama kolu kapasitesinin 2 olduğu durum için elde edilmiş olan indirgenmiş yekpare gözeticili kapalı çevrim melez kontrol sistemine ait merdiven diyagramı kodu (Devam)

7.4.3 Kontrollü Petri ağı için modüler gözetici tasarımı ve PLC ile gerçekleştirilmesi

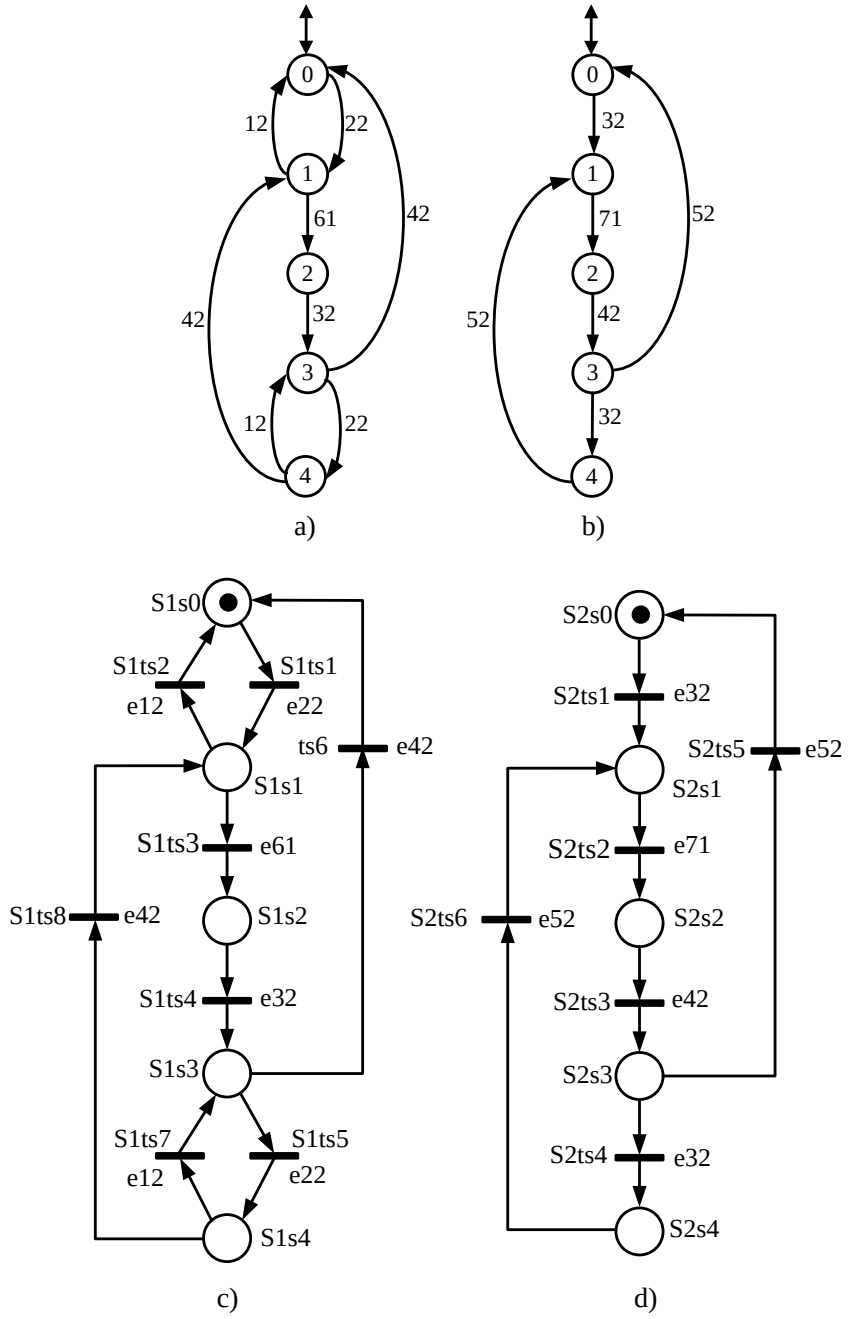
7.4.3.1 Halka toplama kolu kapasitesinin 1 olduğu durum

Deneysel endüstriyel üretim sisteminde bulunan halka toplama kolu kapasitesinin 1 olduğu durum için CtlPN modeli Şekil 7.37'de görülmektedir. Halka toplama kolu kapasitesinin 1 olduğu durum için kontrol spesifikasyonları aşağıdaki gibidir:

1. Seçme selenoidi (itici selenoid) seçme alanında bir halka bulunduğunda ve halka toplama kolu boş olduğunda çalışacaktır (SPEC1)
2. Birleştirme selenoidi (döner selenoid) sadece halka toplama kolunda halka olduğunda ve birleştirme noktası boş olduğunda çalışacaktır (SPEC2)

Yekpare gözetici hesaplanırken spesifikasyonlar dikkate alınarak bir adet yekpare gözetici hesaplanmaktadır. Modüler gözetici hesaplanırken ise spesifikasyonların her

biri için sistemin ilgili parçası dikkate alınarak ayrı bir gözetici elde edilmektedir. Deneysel endüstriyel üretim sisteminin, yukarıdaki kontrol spesifikasyonları için modüler gözeticilerin nasıl hesaplanabileceği tezin önceki kısımlarında detaylı olarak açıklanmıştır. Halka toplama kolu kapasitesinin 1 olduğu durumda yukarıda belirlenen kontrol spesifikasyonları için elde edilen gözeticiler sırası ile MODSUP1 ve MODSUP2'ye ait otomata modelleri Şekil 7.51.(a) ve Şekil 7.51.(b)'de, auto-net modelleri ise sırası ile Şekil 7.51.(c) ve Şekil 7.51.(d)'de görülmektedir.



Şekil 7.51 Halka toplama kolu kapasitesinin 1 olduğu durum için elde edilen;
a) MODSUP1'in otomata modeli, b) MODSUP2'nin otomata modeli,
c) MODSUP1'in auto-net modeli, d) MODSUP2'nin auto-net modeli

Bu gözeticiler için “cp.exe” programı ile hesaplanan kontrol poliçelerinin detayları Ek-H’de görülmektedir. Bu gözeticiler için hesaplanan kontrol poliçelerindeki okların bağlanacağı geçiş isimleri ve kontrol mevkilerinde başlangıçta bulunacak jeton sayısı Çizelge 7.12’de görülmektedir.

Çizelge 7.12 Halka toplama kolu kapasitesinin 1 olduğu durum için hesaplanan modüler gözeticilere ait kontrol poliçeleri

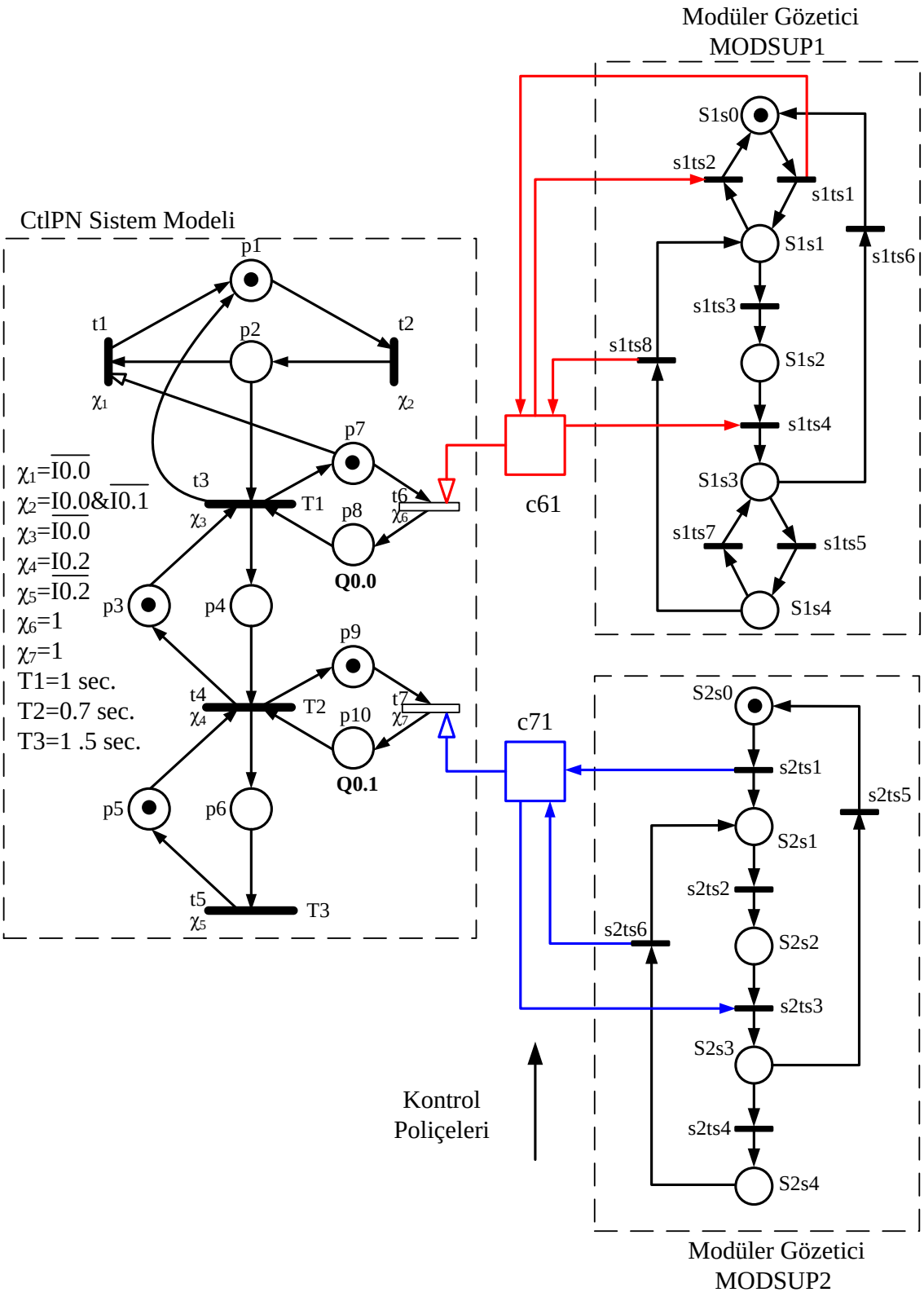
Gözetici	Kontrol mevkisi	Başlangıç jeton sayısı	Pre oklarının bağlanacağı geçişler	Post oklarının bağlanacağı geçişler
MODSUP1	c61	0	S1ts1 S1ts8	S1ts2 S1ts4
MODSUP2	c71	0	S2ts1 S2ts6	S2ts3

Hesaplanan bu kontrol poliçelerinin gösterdiği sıradan ok bağlantıları ışığında, halka toplama kolu kapasitesinin iki olduğu durum için elde edilen kapalı çevrim kontrol sistemi Şekil 7.52’de görülmektedir. Bu kapalı çevrim kontrol sistemi için elde edilen PLC merdiven diyagramı kodu ise Şekil 7.53’te görülmektedir. Bu kodun ilk basamağında başlangıç atamaları yapılmıştır. Buna göre başlangıçta içerisinde jeton bulunan mevkileri simgeleyen hafıza bitleri set edilmiştir. NW1 ve NW5 arasındaki basamaklarda sistemin CtlPN modeli, NW6 ile NW13 arasındaki basamaklarda MODUSP1, NW14 ile NW19 arasındaki basamaklarda MODSUP2, NW20 ve NW21 basamaklarında kontrol edilebilir geçişler (t6 ve t7), ve son olarak NW22 ile NW23 arasındaki basamaklarda ise çıkış aksiyonları gerçekleştirilmiştir. Şekil 7.53’te görülen PLC merdiven diyagramı kodu Siemens S7-300 PLC’ye yüklenerek, sistemin spesifikasyonlarda belirtilen çalışmayı başarılı bir şekilde gerçekleştirdiği gözlemlenmiştir.

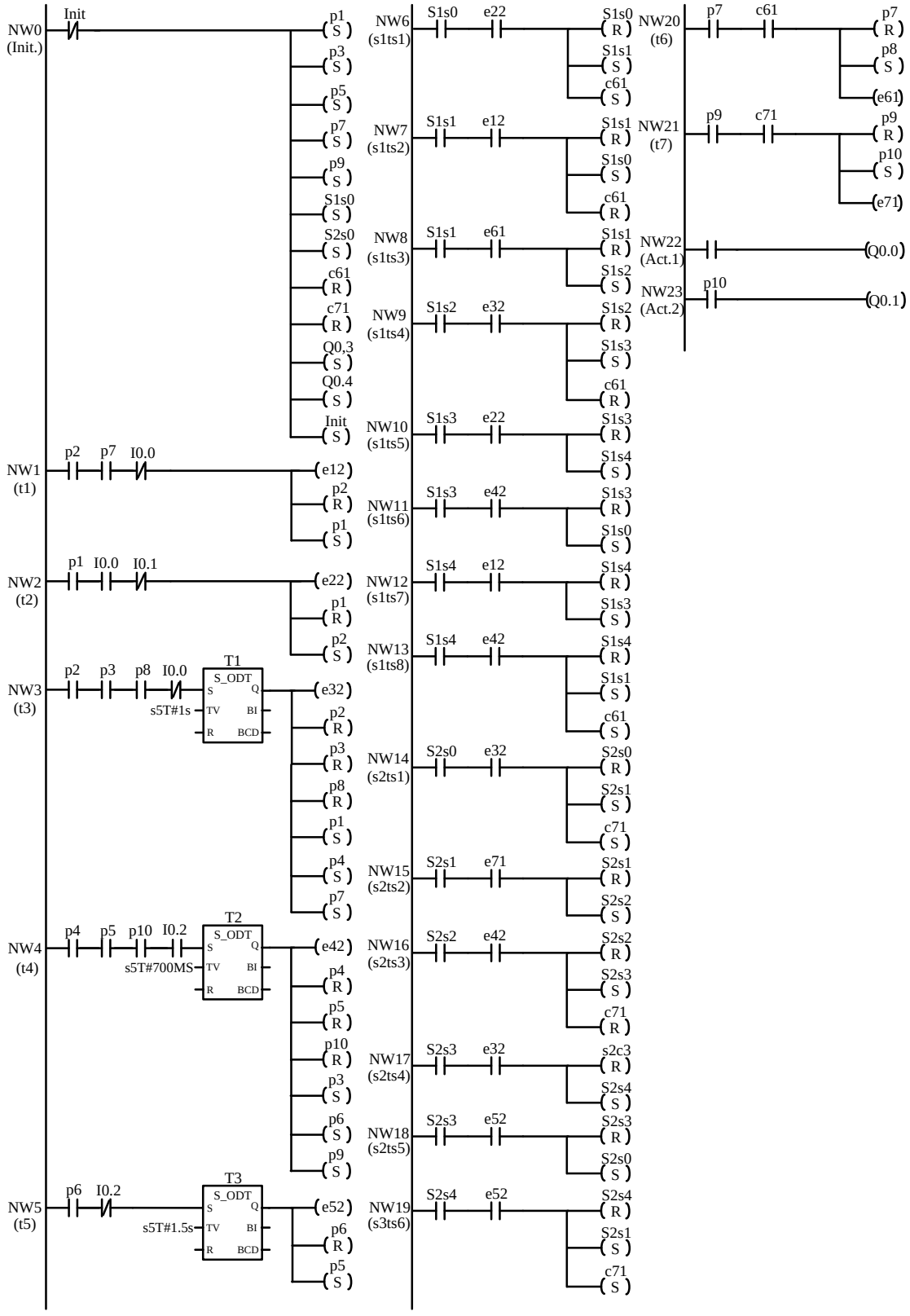
Şekil 7.52’de görülen kapalı çevrim sistemindeki modüler gözeticiler MODSUP1 ve MODSUP2 modellerinde bulunan geçişler (s1ts1, s1ts2, …, s1ts8, s2ts1, s2ts2, …, s2ts6), CtlPN modelinde bulunan geçişlerle (t1, t2, …, t7) senkron çalışmaktadır. MODSUP1 ve MODSUP2 modellerinde bulunan geçişlerin, CtlPN modelindeki hangi geçişlerle senkron olduğu Çizelge 7.13’ten görülmektedir.

Çizelge 7.13 Halka toplama kolu kapasitesinin 1 olduğu durum için hesaplanan modüler gözeticilerde bulunan geçişlerin senkron olduğu CtlPN geçişleri

MODSUP1			MODSUP2		
Auto-net geçiş adı	Otomata geçişi	Senkron olduğu CtlPN geçişi	Auto-net geçiş adı	Otomata geçişi	Senkron olduğu CtlPN geçişi
s1ts1	[0 22 1]	t2	s2ts1	[0 32 1]	t3
s1ts2	[1 12 0]	t1	s2ts2	[1 71 2]	t7
s1ts3	[1 61 2]	t6	s2ts3	[2 42 3]	t4
s1ts4	[2 32 3]	t3	s2ts4	[3 32 0]	t3
s1ts5	[3 22 4]	t2	s2ts5	[3 52 4]	t5
s1ts6	[3 42 0]	t4	s2ts6	[4 52 1]	t5
s1ts7	[4 12 3]	t1			
s1ts8	[4 42 1]	t4			



Şekil 7.52 Halka toplama kolu kapasitesinin 1 olduğu durum için
elde edilen modüler gözeticili kapalı çevrim melez kontrol sistemi



Şekil 7.53 Halka toplama kolu kapasitesinin 1 olduğu durum için
modüler gözeticili kapalı çevrim melez kontrol sistemine ait merdiven diyagramı kodu

7.4.3.2 Halka toplama kolu kapasitesinin 2 olduğu durum

Deneysel endüstriyel üretim sisteminde bulunan halka toplama kolu kapasitesinin 2 olduğu durum için CtlPN modeli Şekil 7.41'de görülmektedir. Halka toplama kolu kapasitesinin 2 olduğu durum için kontrol spesifikasyonları aşağıdaki gibidir:

1. Seçme selenoidi (itici selenoid) seçme alanında bir halka bulunduğunda ve halka toplama kolunda en az bir boş yer olduğunda çalışacaktır (SPEC1).
2. Birleştirme selenoidi (döner selenoid) sadece halka toplama kolunda en az bir halka olduğunda ve birleştirme noktası boş olduğunda çalışacaktır (SPEC2).

Halka toplama kapasitesinin 2 olduğu durumda yukarıda belirlenen kontrol spesifikasyonları için elde edilen gözeticiler sırası ile MODSUP1 ve MODSUP2'ye ait otomata modelleri Şekil 7.54.(a) ve Şekil 7.54.(b)'de, auto-net modelleri ise sırası ile Şekil 7.54.(c) ve Şekil 7.54.(d)'de görülmektedir. Bu gözeticiler için "cp.exe" programı ile hesaplanan kontrol poliçelerinin detayları Ek-I'da görülmektedir. Bu gözeticiler için hesaplanan kontrol poliçelerindeki okların bağlanacağı geçiş isimleri ve kontrol mevkilerinde başlangıçta bulunacak jeton sayıları Çizelge 7.14'te görülmektedir.

Çizelge 7.14 Halka toplama kolu kapasitesinin 2 olduğu durum için hesaplanan modüler gözeticilere ait kontrol poliçeleri

Gözetici	Kontrol mevkisi	Başlangıç jeton sayısı	Pre oklarının bağlanacağı geçişler	Post oklarının bağlanacağı geçişler
MODSUP1	c61	0	S1ts1 S1ts8	S1ts2 S1ts4
MODSUP2	c71	0	S2ts1 S2ts6	S2ts3

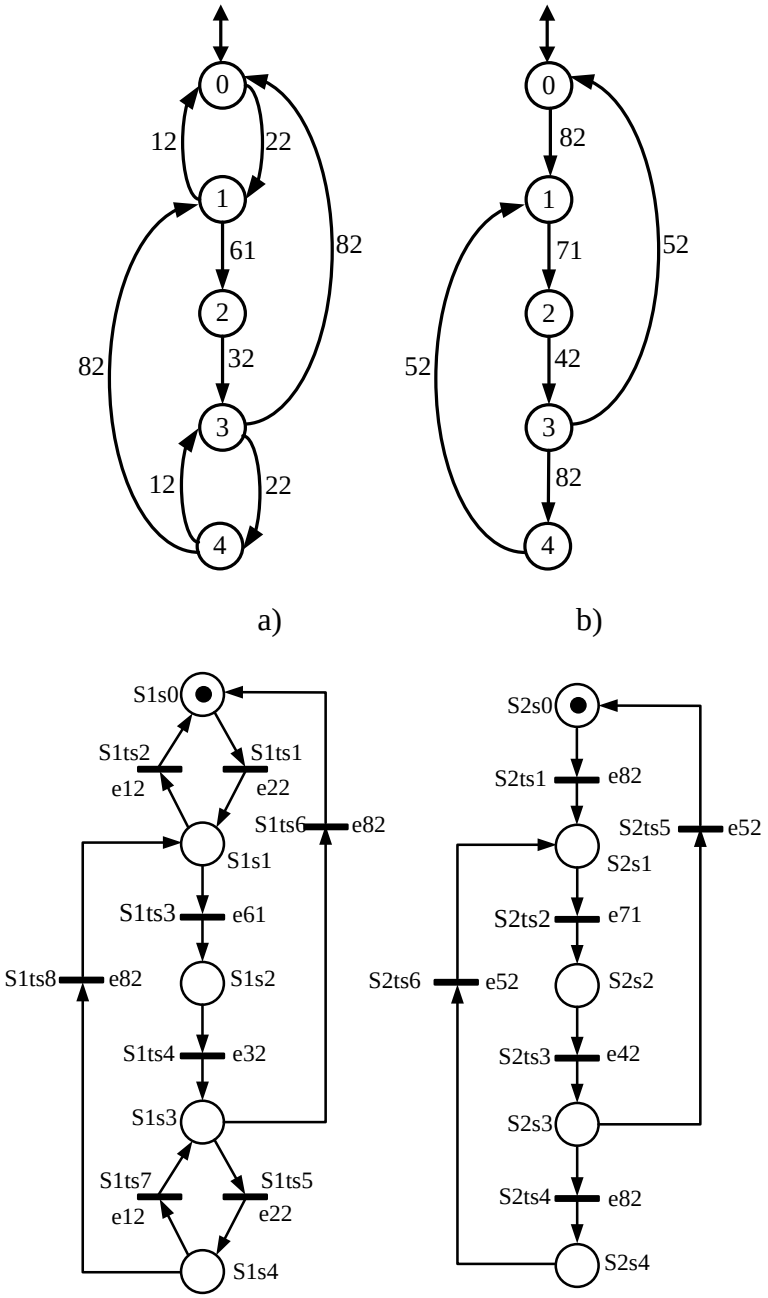
Hesaplanan bu kontrol poliçelerinin gösterdiği sıradan ok bağlantıları ışığında, halka toplama kolu kapasitesinin 2 olduğu durum için elde edilen kapalı çevrim kontrol sistemi Şekil 7.55'te görülmektedir. Bu kapalı çevrim kontrol sistemi için elde edilen PLC merdiven diyagramı kodu ise Şekil 7.56'da görülmektedir. Bu kodun ilk basamağında başlangıç atamaları yapılmıştır. Buna göre başlangıçta içerisinde jeton bulunan mevkileri simgeleyen hafıza bitleri set edilmiştir. NW1 ve NW5 arasındaki basamaklarda sistemin CtlPN modeli, NW6 ile NW13 arasındaki basamaklarda MODUSP1, NW14 ile NW19 arasındaki basamaklarda MODSUP2, NW20 ve NW21

arasındaki basamaklarda kontrol edilebilir geçişler (t6 ve t7) gerçekleştirilmiştir. CtlPN modelinde bulunan kontrol edilemeyen ve harici tetikleme koşulu ‘1’ olan t8 geçişi NW22’de gerçekleştirilmiştir. Son olarak NW23 ile NW24 arasındaki basamaklarda ise çıkış aksiyonları gerçekleştirilmiştir. Şekil 7.56’da görülen PLC merdiven diyagramı kodu Siemens S7-300 PLC’ye yüklenerek, sistemin spesifikasyonlarda belirtilen çalışmayı başarılı bir şekilde gerçekleştirdiği gözlemlenmiştir.

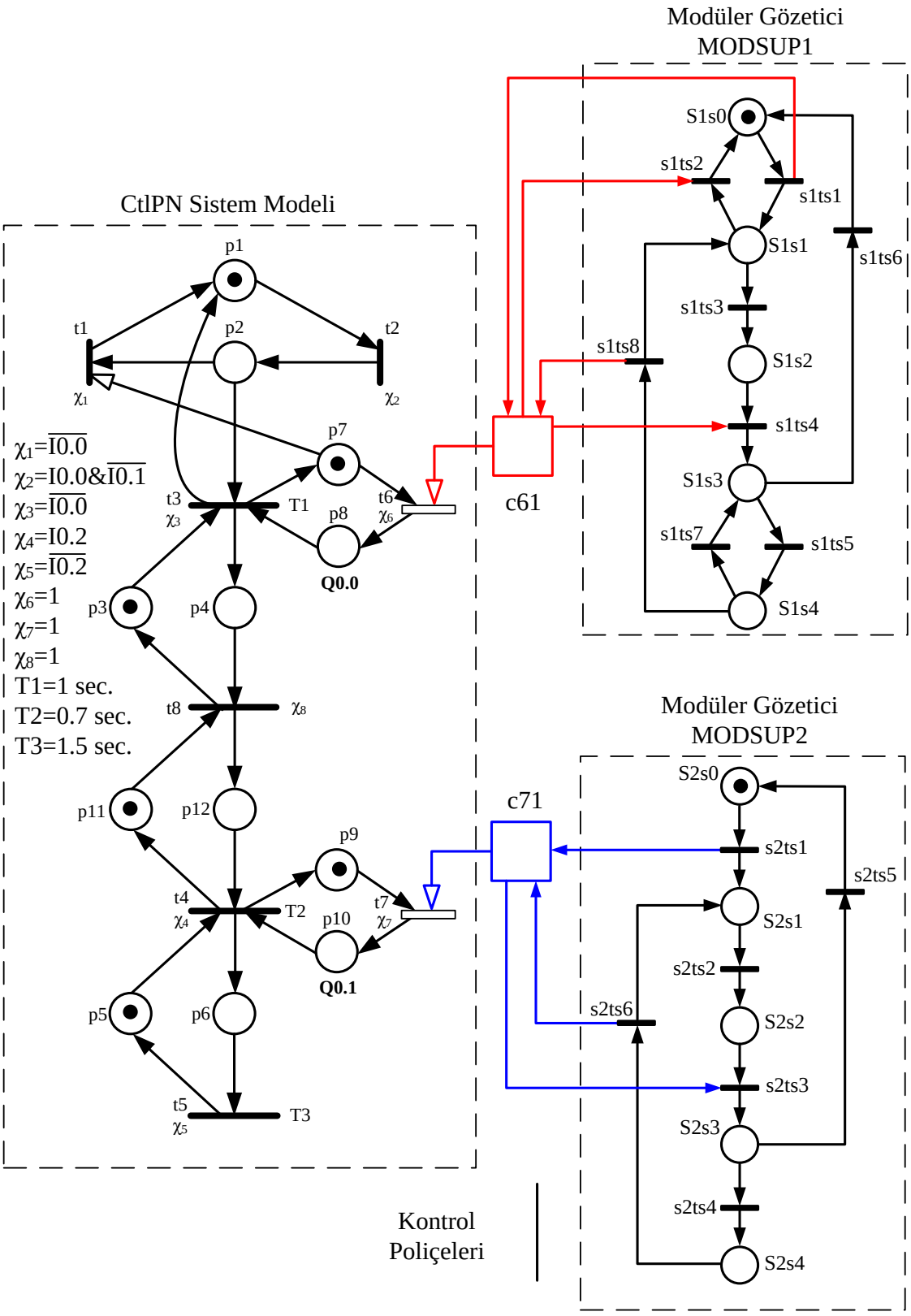
Şekil 7.55’te görülen kapalı çevrim sistemindeki modüler gözeticiler MODSUP1 ve MODSUP2 modellerinde bulunan geçişler (s1ts1, s1ts2, …, s1ts8, s2ts1, s2ts2, …, s2ts6), CtlPN modelinde bulunan geçişlerle (t1, t2, …, t8) senkron çalışmaktadır. MODSUP1 ve MODSUP2 modellerinde bulunan geçişlerin, CtlPN modelindeki hangi geçişlerle senkron olduğu Çizelge 7.15’te görülmektedir.

Çizelge 7.15 Halka toplama kolu kapasitesinin 2 olduğu durum için hesaplanan modüler gözeticilerde bulunan geçişlerin senkron olduğu CtlPN geçişleri

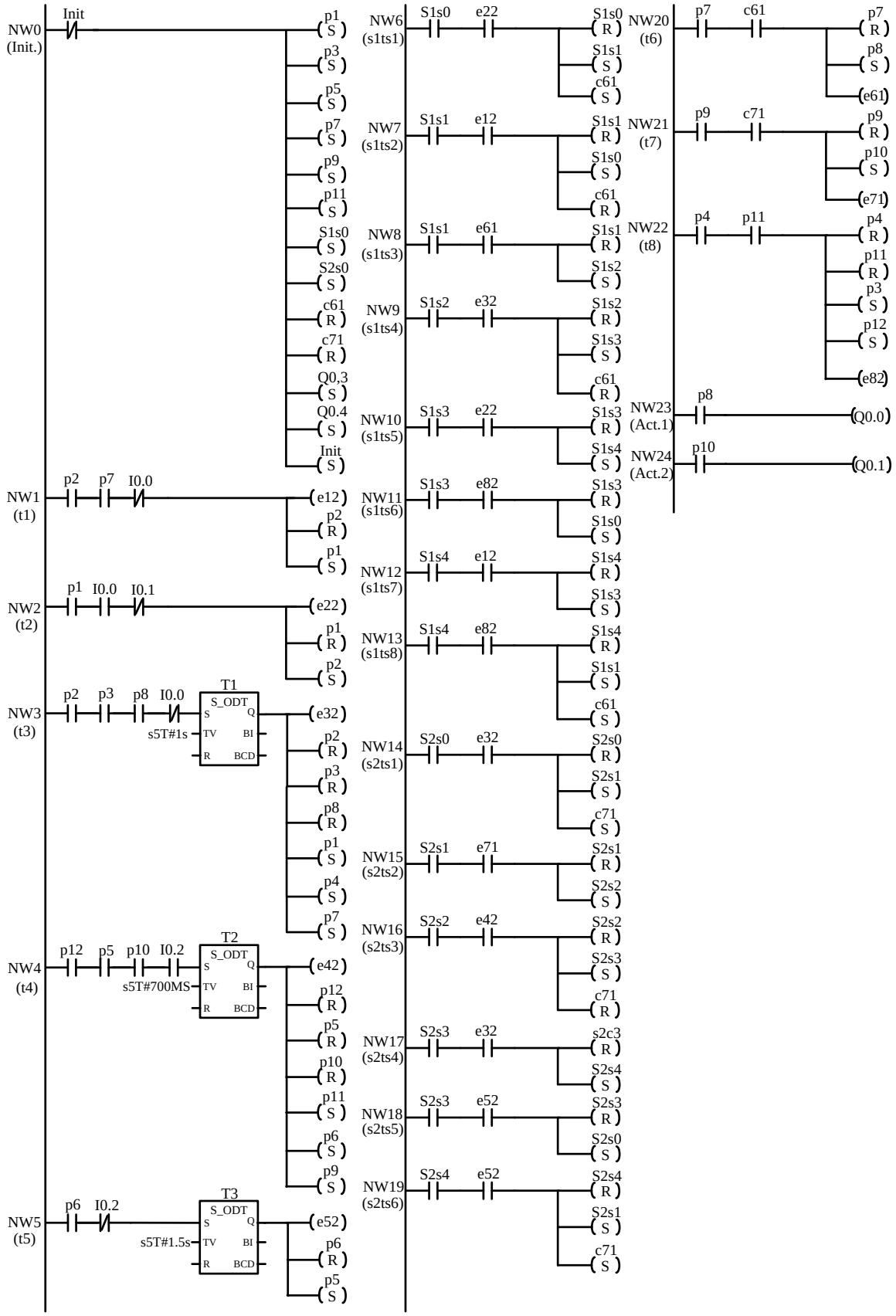
MODSUP1			MODSUP2		
Auto-net geçiş adı	Otomata geçişi	Senkron olduğu CtlPN geçişi	Auto-net geçiş adı	Otomata geçişi	Senkron olduğu CtlPN geçişi
s1ts1	[0 22 1]	t2	s2ts1	[0 82 1]	t8
s1ts2	[1 12 0]	t1	s2ts2	[1 71 2]	t7
s1ts3	[1 61 2]	t6	s2ts3	[2 42 3]	t4
s1ts4	[2 32 3]	t3	s2ts4	[3 82 0]	t8
s1ts5	[3 22 4]	t2	s2ts5	[3 52 4]	t5
s1ts6	[3 82 0]	t8	s2ts6	[4 52 1]	t5
s1ts7	[4 12 3]	t1			
s1ts8	[4 82 1]	t8			



Şekil 7.54 Halka toplama kolu kapasitesinin 2 olduğu durum için elde edilen;
(a) MODSUP1'in otomata modeli, b) MODSUP2'nin otomata modeli,
c) MODSUP1'in auto-net modeli, d) MODSUP2'nin auto-net modeli



Şekil 7.55 Halka toplama kolu kapasitesinin 2 olduğu durum için elde edilen modüler gözeticili kapalı çevrim melez kontrol sistemi



Şekil 7.56 Halka toplama kolu kapasitesinin 2 olduğu durum için modüler gözeticili kapalı çevrim melez kontrol sistemine ait merdiven diyagramı kodu

7.4.3.3 Halka toplama kolu kapasitesinin 5 olduğu durum

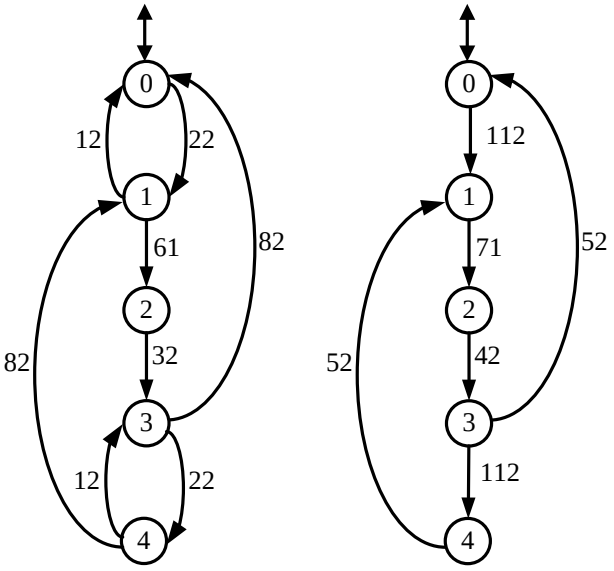
Halka toplama kolu kapasitesinin 5 olduğu durum için kontrol spesifikasyonları aşağıdaki gibidir:

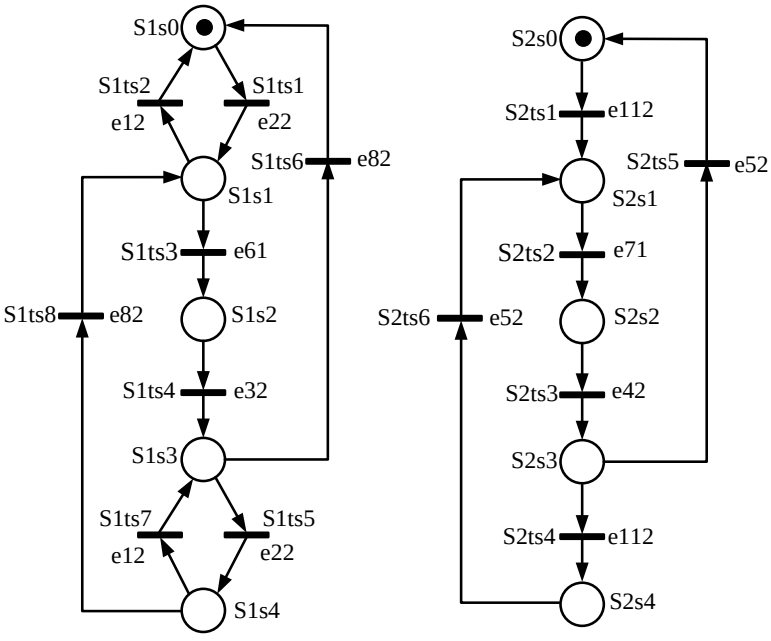
1. Seçme selenoidi (itici selenoid) seçme alanında bir halka bulunduğunda ve halka toplama kolunda en az bir boş yer olduğunda çalışacaktır (SPEC1).
2. Birleştirme selenoidi (döner selenoid) sadece halka toplama kolunda en az bir halka olduğunda ve birleştirme noktası boş olduğunda çalışacaktır (SPEC2).

Halka toplama kolu kapasitesinin 5 olduğu durumda yukarıda belirlenen kontrol spesifikasyonları için elde edilen gözeticiler sırası ile MODSUP1 ve MODSUP2'ye ait otomata modelleri Şekil 7.57.(a) ve Şekil 7.57.(b)'de, auto-net modelleri ise sırası ile Şekil 7.57.(c) ve Şekil 7.57.(d)'de görülmektedir. Bu modüler gözeticiler için "cp.exe" programı ile hesaplanan kontrol poliçelerinin detayları Ek-J'de görülmektedir. Bu gözeticiler için hesaplanan kontrol poliçelerindeki okların bağlanacağı geçiş isimleri ve kontrol mevkilerinde başlangıçta bulunacak jeton sayıları Çizelge 7.16'da görülmektedir.

Çizelge 7.16 Halka toplama kolu kapasitesinin 5 olduğu durum için hesaplanan modüler gözeticilere ait kontrol poliçeleri

Gözetici	Kontrol mevkisi	Başlangıç jeton sayısı	Pre oklarının bağlanacağı geçişler	Post oklarının bağlanacağı geçişler
MODSUP1	c61	0	S1ts1 S1ts8	S1ts2 S1ts4
MODSUP2	c71	0	S2ts1 S2ts6	S2ts3





Şekil 7.57 Halka toplama kolu kapasitesinin 5 olduğu durum için elde edilen;
(a) MODSUP1'in otomata modeli, b) MODSUP2'nin otomata modeli,
c) MODSUP1'in auto-net modeli, d) MODSUP2'nin auto-net modeli

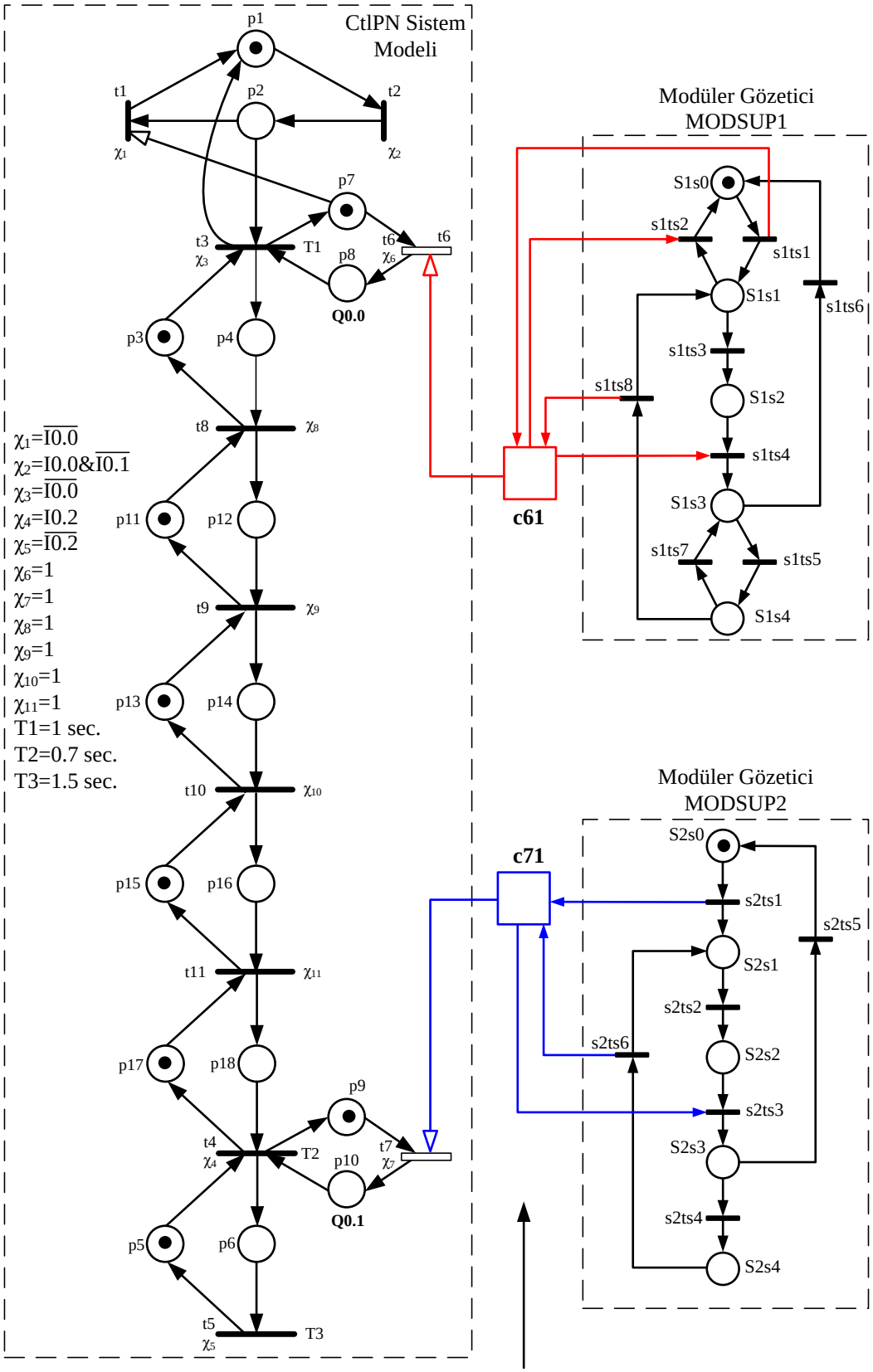
Hesaplanan bu kontrol poliçelerinin gösterdiği sıradan ok bağlantıları ışığında, halka toplama kolu kapasitesinin 5 olduğu durum için elde edilen kapalı çevrim kontrol sistemi Şekil 7.58'de görülmektedir. Bu kapalı çevrim kontrol sistemi için elde edilen PLC merdiven diyagramı kodu ise Şekil 7.59'da görülmektedir. Bu kodun ilk

basamağında başlangıç atamaları yapılmıştır. Buna göre başlangıçta içerisinde jeton bulunan mevkileri simgeleyen hafıza bitleri set edilmiştir. NW1 ve NW5 arasındaki basamaklarda sistemin CtlPN modeli, NW6 ile NW13 arasındaki basamaklarda MODUSP1, NW14 ile NW19 arasındaki basamaklarda MODSUP2, NW20 ve NW21 arasındaki basamaklarda kontrol edilebilir geçişler (t6 ve t7) gerçekleştirilmiştir. CtlPN modelinde bulunan kontrol edilemeyen ve harici tetikleme koşulu '1' olan t8, t9, t10 ve t11 geçişleri NW22 ile NW25 arasındaki basamaklarda gerçekleştirilmiştir. Son olarak NW26 ile NW27 arasındaki basamaklarda ise çıkış aksiyonları gerçekleştirilmiştir. Şekil 7.59'da görülen PLC merdiven diyagramı kodu Siemens S7-300 PLC'ye yüklenerek, sistemin spesifikasyonlarda belirtilen çalışmayı başarılı bir şekilde gerçekleştirdiği gözlemlenmiştir.

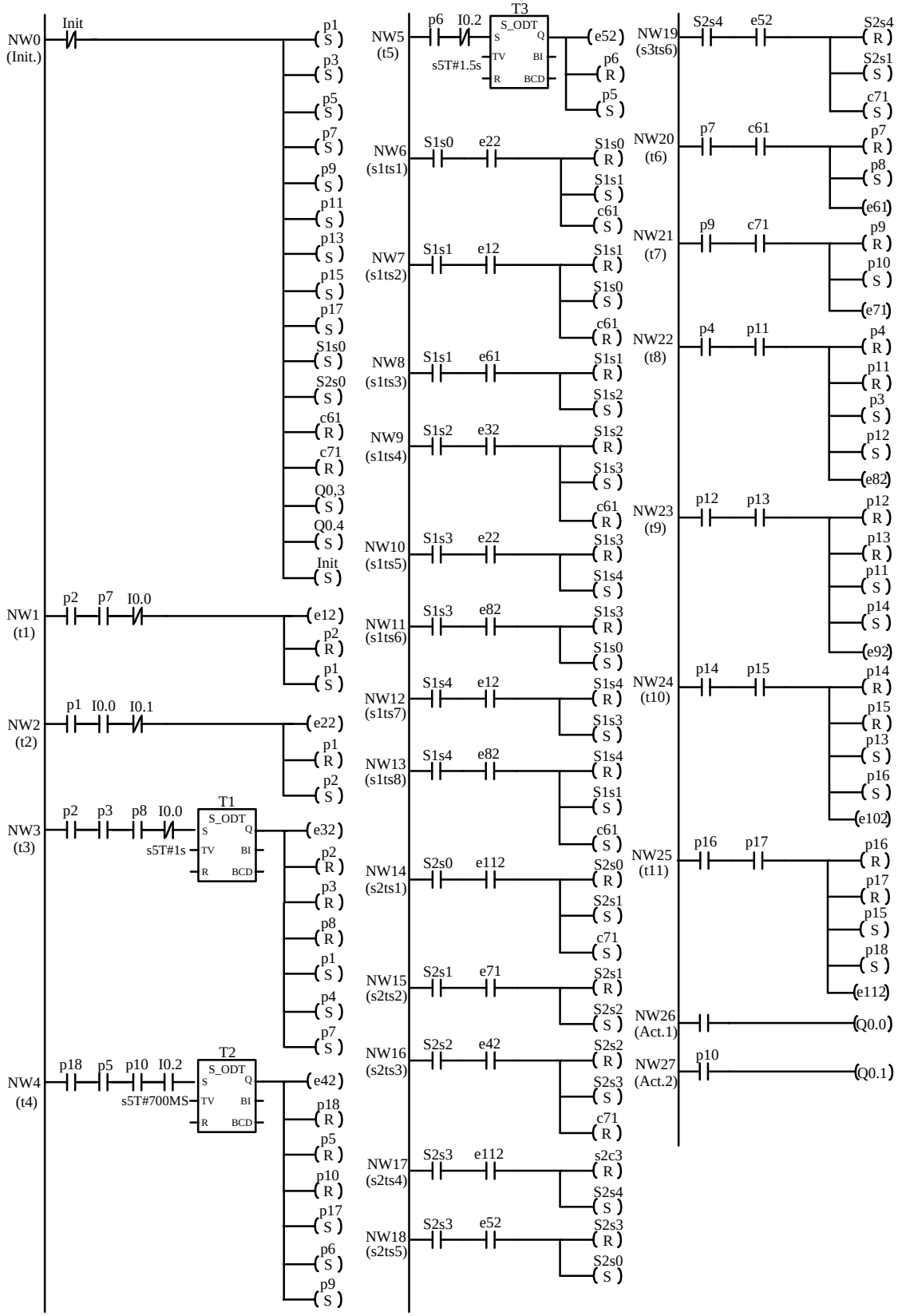
Şekil 7.58'de görülen kapalı çevrim sistemindeki modüler gözeticiler MODSUP1 ve MODSUP2 modellerinde bulunan geçişler (s1ts1, s1ts2, …, s1ts8, s2ts1, s2ts2, …, s2ts6), CtlPN modelinde bulunan geçişlerle (t1, t2, …, t11) senkron çalışmaktadır. MODSUP1 ve MODSUP2 modellerinde bulunan geçişlerin, CtlPN modelindeki hangi geçişlerle senkron olduğu Çizelge 7.17'den görülmektedir.

Çizelge 7.17 Halka toplama kolu kapasitesinin 5 halka olduğu durum için hesaplanan modüler gözeticilerde bulunan geçişlerin senkron olduğu CtlPN geçişleri

MODSUP1			MODSUP2		
Auto-net geçiş adı	Otomata geçişi	Senkron olduğu CtlPN geçişi	Auto-net geçiş adı	Otomata geçişi	Senkron olduğu CtlPN geçişi
s1ts1	[0 22 1]	t2	s2ts1	[0 112 1]	t11
s1ts2	[1 12 0]	t1	s2ts2	[1 71 2]	t7
s1ts3	[1 61 2]	t6	s2ts3	[2 42 3]	t4
s1ts4	[2 32 3]	t3	s2ts4	[3 112 0]	t11
s1ts5	[3 22 4]	t2	s2ts5	[3 52 4]	t5
s1ts6	[3 82 0]	t8	s2ts6	[4 52 1]	t5
s1ts7	[4 12 3]	t1			
s1ts8	[4 82 1]	t8			



Şekil 7.58 Halka toplama kolu kapasitesinin 5 olduğu durum için elde edilen modüler gözeticili kapalı çevrim melez kontrol sistemi



Şekil 7.59 Halka toplama kolu kapasitesinin 5 olduğu durum için
modüler gözeticili kapalı çevrim melez kontrol sistemine ait merdiven diyagramı kodu

7.5 Sonuç

Bu kısmında CtlPN kullanılarak modellenmiş ayrık olay sistemleri için, derlenmiş denetleyiciler elde edilmesi için yöntemler önerilmiştir. Ayrıca, önerilen hesaplama yöntemlerinin deneysel endüstriyel üretim sistemi kullanılarak gerçek sistemlere nasıl uygulanabileceği gösterilmiştir. Bu amaç doğrultusunda deneysel endüstriyel üretim sisteminin halka toplama kolundaki halka sayısının farklı durumları için, yekpare, indirgenmiş ve modüler gözeticiler hesaplanmıştır. Hesaplamalar sonucunda elde edilen gözeticilerin PLC ile gerçekleştirilmelerinin nasıl yapılabileceği açıklanmıştır.

Deneysel endüstriyel üretim sisteminin çeşitli durumları için hesaplanan gözeticileri gerçekleştirmek için yazılan PLC kodlarının boyutları Çizelge 7.18'de görülmektedir. Buna göre halka toplama kolu kapasitesinin 1 olduğu durumda hesaplanan yekpare gözeticiyi gerçekleştirmek için yazılan kod hafızada 382 Byte yer kaplamaktadır. Aynı durumda aynı kontrol işlemini yerine getiren indirgenmiş gözetici 398 Byte, modüler gözeticiler ise 302 Byte yer kaplamaktadır. Benzer olarak, halka toplama kolu kapasitesinin 2 olduğu durum için, yekpare gözetici 680, indirgenmiş gözetici 544 ve modüler gözeticiler ise 318 Byte yer kaplamaktadır. Çizelgedeki sonuçlardan görüldüğü üzere, aynı kontrol işlemini yerine getiren gözeticilerden hafızada en fazla yeri yekpare gözetici kaplamaktadır. İndirgenmiş gözeticiler hafızada yekpare gözeticilerden daha az yer kaplamaktadır. Hafızada en az yeri ise modüler gözeticiler kaplamaktadır. Özel olarak halka toplama kolu kapasitesinin 1 olduğu durumda indirgenmiş gözetici için yazılan PLC kodu, yekpare gözetici için yazılan PLC kodundan fazla yer kaplamıştır. Bunun sebebi, çığ etkisi probleminin çözümü için kullanılan hafızanın, gözeticinin indirgenmesinden kazanılan hafızadan daha büyük olmasıdır. Halka toplama kolu kapasitesinin 5 olduğu durum için gerçekleştirilen modüler gözeticiler hafızada 366 Byte yer kaplamaktadır. Sonuçlardan görüleceği üzere, deneysel endüstriyel üretim sisteminin kontrolünde elde edilen modüler gözeticilerin kullanımı daha uygun olacaktır.

Çizelge 7.18 Yazılan PLC kodlarının boyutları

Halka Kapasitesi	Yekpare Gözetici	İndirgenmiş Gözetici	Modüler Gözeticiler
1	382 byte	398 byte	302 byte
2	680 byte	544 byte	318 byte

BÖLÜM VIII

SONUÇLAR

Gözetimli Kontrol Teorisi, AOS'lerin kontrolü konusunda önerilen genel ve formal bir yöntem olmasına karşın, durum patlaması problemi nedeniyle büyük boyutlu sistemlerde etkin bir çözüm sunamamaktadır. AOS'ler için denetleyici hesaplamasında yaygın olarak kullanılan bir diğer yöntem ise Petri ağlarının kullanımıdır. Petri ağları, vektör eklentili bir modelleme aracı olduğundan büyük boyutlu sistemler için kompakt modeller elde edilmesine imkân vermektedir. Ancak, bazı durumlarda Petri ağları ile elde edilen modeller için optimum denetleyicilerin bulunması zordur. Bu nedenlerden dolayı, AOS'lerinin etkin kontrolü için SCT ve Petri ağları, Uzam ve Wonham tarafından birleştirilerek melez denetleyicilerin elde edilebileceği yeni bir yaklaşım ortaya konulmuştur [52].

Tez kapsamında yapılan ilk çalışma [52]'de önerilen melez yöntemin gerçek sistemlere uygulanabilirliğinin gösterilmesidir. Bu amaçla yapılan uygulamalardan ilki Niğde Üniversitesi Elektrik-Elektronik Mühendisliği Bölümü laboratuarında bulunan uzaktan kumandalı otomatik kapı modelinin kontrolüdür. Bu uygulamadan elde edilen sonuçlar ulusal bir sempozyumda bildiri olarak sunulmuştur [66]. Yapılan ikinci uygulamada ise deneysel endüstriyel üretim sistemi kullanılmış ve elde edilen sonuçlar SCIE kapsamında taranan uluslar arası bir dergilerde makale olarak yayınlanmıştır [67]. Yapılan çalışmalarda [66, 67] melez yöntemin gerçek bir AOS'nin kontrolünde başarılı bir biçimde kullanılabileceği gösterilmiştir.

Yukarıdaki uygulamalarda elde edilen denetleyici yapılarındaki gözeticiler yekpare (monolithic) dir. Yekpare gözeticilerin durum sayısının fazlalığından dolayı gerçekleştirilmeleri için yazılan PLC merdiven diyagramı kodunun boyutu da büyük olmaktadır. Çok karmaşık sistemler için elde edilecek gözeticilerin durum sayısı milyonlarla ifade edilmektedir. Böyle bir sistem için yekpare gözetici yapısının kullanılması imkânsız olacaktır.

Bu sorunun çözümüne yönelik yapılan ilk çalışmada indirgenmiş gözeticilerin bulunduğu denetleyici yapıları dikkate alınmıştır. İndirgenmiş gözetici yapılarının

durum sayısında azalma sağlaması ve TCT yazılımı ile kolaylıkla hesaplanabilmesine karşın, bu gözetici yapıları yekpare gözetici yapılarına göre çok daha kompakttır. Bundan dolayı indirgenmiş gözeticilerin PLC ile gerçekleştirilmeleri esnasında çeşitli problemlerle karşılaşılmaktadır [78]. Bunlardan en önemlisi çığ etkisi problemidir. Tez kapsamında etkin bir çığ etkisi problemi önleme yöntemi (AEEM) önerilmiştir. Bu yöntem için bir bilgisayar programı (Av_effect_detector.exe) yazılmıştır. AEEM ile ilgili çalışmalardan elde edilen sonuçlar ulusallar arası bir sempozyumda bildiri olarak sunulmuştur [77]. Melez yöntem ile hesaplanan indirgenmiş yekpare gözeticilerin PLC ile gerçekleştirilmesi esnasında karşılaşılan çığ etkisi probleminin AEEM kullanılarak çözümü ve gerçek sistemlere uygulanabilirliği deneysel endüstriyel üretim sistemi kullanılarak gösterilmiştir.

Melez yöntemin temelini, Petri ağlarından otomata yapılarına dönüşüm oluşturmaktadır. Petri ağları ve otomata yapıları arasındaki dönüşüm için [52]'de temel bir yaklaşım ortaya konmasına karşın, Petri ağlarının çeşitli durumları ve bileşenleri için (örneğin ağırlıklı yetkileme ve yasaklama oku) genel yöntemler ortaya konmamıştır. Petri ağlarından otomata yapılarına dönüşüm için genel yaklaşımlar ve PN olarak hesaplanmış AOS denetleyicilerinin TCT yazılımı kullanılarak doğruluk analizinin yapılabileceği bir yöntem tez kapsamında yapılan çalışmalar sonrasında önerilmiştir [90].

Tez kapsamında melez yöntemin etkinliğini arttırmak amacı ile modüler gözeticileri içeren melez yaklaşımlar ortaya konulmuştur. Sistemin kontrolünü belirten her bir spesifikasyon ve bu spesifikasyonu ilgilendiren sistem parçası dikkate alınarak modüler gözeticiler elde edilmektedir. Petri ağı yapısı ile entegre edilerek elde edilen modüler melez denetleyici yapısının PLC merdiven diyagramı koduna dönüşümü ve gerçek sistemlerin kontrolünde kullanılması da gösterilmiştir.

Petri ağlarının genişletilmiş biçimi olan kontrollü Petri ağları (CtlPN) için literatürde önerilen denetleyici yapıları gönderim (mapping) gözetici türündendir. Bu yapıda, denetleyici çevrimiçi (on-line) olarak sistemin her durumu için yeni bir kontrol poliçesi hesaplamaktadır. Bu yapı çevrimiçi olarak hesaplama gerektirir. Tez kapsamında, [52]'de önerilen yaklaşım temel alınarak CtlPN'ler için derlenmiş melez

denetleyicilerin elde edilebilmesine imkân sunan yeni yöntemler ortaya konmuştur. Bu kapsamda yapılan çalışmalar aşağıda sıralanmıştır:

- CtlPN'ler için derlenmiş yekpare melez denetleyici hesaplama yöntemi: Bu yöntem [52]'de önerilen yaklaşım temel alınarak geliştirilmiştir. Tez kapsamında, [52]'de önerilen yaklaşım kullanılarak hesaplanan auto-net gözetici ile CtlPN arasındaki bağlantıları tanımlayan kontrol poliçesinin hesaplanmasına yönelik bir bilgisayar programı (cp.exe) yazılmıştır. Bu programın çıktısı ile belirlenen kontrol poliçesi, auto-net ve CtlPN birleştirilerek derlenmiş melez denetleyici elde edilmektedir. Elde edilen denetleyicilerin gerçek sistemlerin kontrolüne uygulanması deneysel endüstriyel üretim sistemi kullanılarak gösterilmiştir.
- CtlPN'ler için derlenmiş indirgenmiş melez denetleyici hesaplama yöntemi: Bu yöntemde Petri ağları için önerilen indirgenmiş yöntemde olduğu gibi gözetici yapısının durum sayısının azaltılması temel yaklaşımdır. Elde edilen gözeticide karşılaşılabilecek muhtemel çığ etkisi problemi önerilen çığ etkisi önleme metodu (Avalanche Effect Elimination Method - AEEM) ile giderilebilmektedir. Kontrol poliçesi "cp.exe" programı ile hesaplanmaktadır. Elde edilen denetleyicilerin gerçek sistemlerin kontrolüne uygulanması deneysel endüstriyel üretim sistemi kullanılarak gösterilmiştir.
- CtlPN'ler için derlenmiş modüler melez denetleyici hesaplama yöntemi: Önerilen bu yöntemde de gözeticiler Petri ağları için önerilen melez yöntemde olduğu gibi modüler olarak elde edilmektedir. Kontrol poliçesi "cp.exe" programı ile hesaplanmaktadır. Elde edilen denetleyicilerin gerçek sistemlerin kontrolüne uygulanması deneysel endüstriyel üretim sistemi kullanılarak gösterilmiştir.

Tez kapsamında AOS'ler için denetleyicilerin etkin olarak hesaplanabileceği ve PLC ile gerekleştirilebileceği yeni yöntemler ortaya konulmuştur. Ortaya konan bu yöntemlerin en etkini modüler gözeticili olan yöntemlerdir. Önerilen modüler gözeticilerin durum sayısının azaltılması konusunda gelecekte çalışmalar yapılması planlanmaktadır.

Günümüzde modern üretim sistemlerinde modüler imalat hücreleri çok yaygın olarak kullanılmaktadır. Bu tür sistemlerin kontrolünde her bir modül kendi kumanda sistemi

tarafından kontrol edilmektedir. Bütün modüllerin kumanda sisteminin bağlı olduğu ve hiyerarşik olarak lokal kontrol mekanizmalarının üzerinde olan denetleyiciler (supervisor) de koordinasyon görevi üstlenmektedir. Modüler üretim sistemlerine örnek olarak esnek imalat sistemleri (flexible manufacturing systems - FMS) verilebilir. Bu tez kapsamında önerilen yeni kontrol metotlarının tamamı endüstriyel sistemlerin kontrolüne yönelik olup, önerilen metotların tamamının deneysel endüstriyel üretim sistemi kullanılarak gerçek sistemlerin kontrolünde uygulanabilirliği gösterilmiştir. Tez kapsamında önerilen metotlar günümüz endüstrisinin ihtiyaçlarını karşılayabilecek nitelikte olup büyük boyutlu ve karmaşık yapıya sahip üretim sistemleri için denetleyici tasarlanması konusunda faydalı olacağı değerlendirilmektedir.

KAYNAKLAR

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

[2] Koussoulas, N. T., Discrete Event Dynamic Systems: Which Model to Choose?, Computer Aided Control System Design: Methods, Tools and Related Topics, M.A. Bryds and K. Malinowski (Eds.), World Sci. Publishing Co. Pte. Ltd., 495-513, 1994.

[3] Wonham, W. M. and Ramadge, P.J., On the Supremal Controllable Sublanguage of a Given Language, SIAM Journal on Control and Optimization, 25, 3 637-659, 1987.

[4] Ramadge, P. J. and Wonham, W. M., Supervisory Control of a Class of Discrete Event Processes, SIAM Journal on Control and Optimization, 25, 1, 206-230, 1987.

[5] Ramadge, P. J. and Wonham W. M., The Control of Discrete Event Systems, Proc. IEEE, 77, 1, 81-98, 1989.

[6] Lin, F. and Wonham W. M., On Observability of Discrete Event Systems, Info. Sciences, 44, 173-198, 1988.

[7] Ramadge, P. J. and Wonham, W. M., Modular Feedback Logic for Discrete Event System, SIAM Journal on Control and Optimization, 25, 5, 1202-1218, 1987.

[8] Ramadge, P. J. and Wonham, W. M., Modular Supervisory Control of Discrete Event Systems, Proc. of the Seventh Int. Conf. on the Analysis and Optimization of Systems, Antibes, 202-214, Lecture Notes in Control and Optimization of Systems Vol. 83 (A. Bensoussan and J.L. Lions, eds.) Springer-Verlag, New York, 1986.

[9] Lin, F. and Wonham, W. M., Decentrialised Supervisory Control of Discrete Event Systems, Info. Sciences, 44, 199-224, 1988.

[10] Balemi, S., Hoffmann, G. J., Gyugyi, P., Wongto, H. and Franklin, G. F., Supervisory Control of a Rapid Thermal Multiprocessor, IEEE Tran. on Automatic Control, 38, 7,1040-1059, 1993.

[11] Lauzon, S. C., Ma, A. K. L., Mills, J. K. and Benhabib, B., Application of Discrete Event System Theory to Flexible Manufacturing, IEEE Control Systems Magazine, 16-1, 41-48, 1996.

[12] Nourelfath, M. and Niel, E., Modular Supervisory Control of an Experimental Automated Manufacturing System, Control Engineering Practice, 12, 2, 205-216, 2004.

[13] Leduc, R.J., Lawford, M. and Dai, P.C., Hierarchical Interface Based Supervisory Control of a Flexible Manufacturing System, IEEE Tran. on Control Systems Technology, 14, 4, 654-668, 2006.

[14] Yalcin, A., Khemuka, A. and Deshpande, P., Modelling Inter Task Dependencies and Control of Workflow Managements Systems Based on Supervisory Control Theory, Int. Journal of Production Research, 43, 20, 4359-4379, 2005.

[15] Zhou, M.C. and DiCesare, F., Petri Net Synthesis for Discrete Event Control of Manufacturing Systems, Kluwer Academic Publishers, Boston, 1993.

[16] Petri, C. A., Kommunikation mit Automaten Schriften des Rheinisch" Westfalischen Inst. fur Intrumentelle Mathematik and der Universitat Bonn, 1962, Çeviri Green C. F., Applied Data Research Inc., Suppl 1 to Tech report RADC-TR-65-337, NY, 1963.

[17] Zurawski, R. and Zhou, M.C., Petri Nets and Industrial Applications: A Tutorial, IEEE Trans. on Industrial Electronics, 41, 6, 567-583, 1994.

[18] David, R. and Alla, H., Petri Nets for Modelling of Dynamic Systems: A Survey, Automatica, 30, 2, 175-202, 1994.

[19] Desrochers, A. A. and Al-Jaar, R.Y., Applications of Petri Nets in Manufacturing Systems, Modelling, Control and Performance Analysis, IEEE Press, New York, 1995.

[20] Genrish, H. and Lautenbach, K., System Modelling with High Level Petri Nets, Theoretical Computer Science, 13, 109 – 136, 1981.

[21] Jensen, K. Coloured Petri Nets and the Invariant Method, Theoretical Computer Science, 14, 317-336, 1981.

[22] Reising, W., Petri Nets : An Introduction, Springer-Verlag, 1985.

[23] Sibertin-Blanc, C., High Level Petri Nets with Data Structures, 6th European Workshop on Applications and Theory of Petri Nets, Helsinki, Findland, 1985.

[24] Chen, S., Ke, J. and Chang, J., Knowledge Representation Using Fuzzy Petri Nets, IEEE Trans. on Knowledge and Data Eng., 2, 3,311 – 319, 1990.

[25] Garg, M.L., Ahson, S.I. and Gupta, P.V., A Fuzzy Petri Net for Knowledge Representation and Reasoning, Information Processing Letters, 39, 165 – 171, 1991.

[26] Looney, C.G., Fuzzy Petri Nets for Rule-Based Decision Making, IEEE Transactions on Systems, Man, and Cybernetics, 18, 1, 178-183, 1988.

[27] Valette, R., Cardoso, J. and Dubois, D., Monitoring Manufacturing Systems by Means of Petri Nets with Imprecise Markings, IEEE Int. Symposium on Intelligent Control, 233 - 237, 1989.

[28] Suziki, I. and Lu, H., Temporal Petri Nets and Their Application to Modelling and Analysis of a Handshake Daisy Chain Arbiter, IEEE Trans. on Computers, 38, 5, 696 – 704, 1989.

[29] Guia, A., Petri Net Techniques for Supervisory Control of Discrete Event Systems, 1st Int. Workshop on Manufacturing and Petri Nets, Osaka - JAPAN, 1 – 21, 1996.

[30] Holloway, L. E. and Krogh, B.H., Synthesis of Feedback Control Logic for a Class of Controlled Petri Nets, IEEE Trans. on Aut. Cont., 35, 5, 514-523, 1990.

[31] Krogh, B. H. and Holloway, L.E., Synthesis of Feedback Control Logic for Discrete Manufacturing Systems, Automatica, 27, 4, 641-651, 1991.

[32] Sreenivas, R.S., On Asymptotically Efficient Solutions for a Class of Supervisory Control Problems, IEEE Int. Conf. on Systems, Man and Cybernetics, San Antonio, Texas, 1994.

[33] Sreenivas, R.S., On the Implications of Solvability of the Supervisory Control Problem for Certain Infinite State Discrete Event Dynamic Systems, 28th Ann. Conf. in Information Sciences and Systems, Princeton Univ., New Jersey, 1994.

[34] Sreenivas, R.S., On Asymptotically Efficient Solutions for a Class of Supervisory Control Problems, IEEE Trans. on Automatic Control, 41, 12, 1736-1750, 1996.

[35] Holloway, L.E., Guan, X. and Zhang, L., A Generalization of State Avoidance Policies for Controlled Petri Nets, IEEE Trans. on Automatic Control, 41, 6, 804 – 816, 1996.

[36] Yamalidou, K., Moody, J., Lemmon, M., and Antsaklis, P., Feedback Control of Petri Nets Based on Place Invariants, Automatica, 32, 1, 15-28, 1996.

[37] Uzam, M., Jones, A. H. and Yücel, İ., Using a Petri-Net-Based Approach for the Real-Time Control of an Experimental Manufacturing System, International Journal of Advanced Manufacturing Technology, 16, 498-515, 2000.

[38] Uzam, M. and Jones, A. H., A New Petri-Net-Based Synthesis Technique for Supervisory Control of Discrete Event Systems, Turkish Journal of Electrical Engineering and Computer Sciences, 10, 1, 85-109, 2002.

[39] Jones A.H, and Uzam M., Towards a Unified Approach to the Design of Knowledge Based Agile Manufacturing Systems, The Journal of Integrated Design and Process Science, 2, 3, 55-76, 1998.

[40] Uzam, M., The Enabling Arc Method for the Synthesis of Petri-Net-Based Supervisors for Discrete Event Systems, Niğde Üniversitesi, Mühendislik-Mimarlık Fakültesi, Mühendislik Bilimleri Dergisi, 2, 1, 53-68, 1998.

[41] Uzam, M., Jones, A.H. and Yücel, İ., A Rule Based Methodology for Supervisory Control of Discrete Event Systems Modelled as Automation Petri Nets, International Journal of Intelligent Control and Systems, 3, 3,297-326, 1999.

[42] Uzam, M., Synthesis of Feedback Control Elements for Discrete Event Systems Using Petri Net Models and Theory of Regions, The International Journal of Advanced Manufacturing Technology, 24, 1-2, 48-69, 2004.

[43] Badouel, E. and Darondeau, P., Theory of Regions, Lectures on Petri Nets I : Basic Models, Advances in Petri Nets, Lecture Notes in Computer Science, Springer-Verlag, Berlin, 491, 529-586, 1998.

[44] Uzam, M., An Optimal Deadlock Prevention Policy for Flexible Manufacturing Systems Using Petri Net Models with Resources and the Theory of Regions, The International Journal of Advanced Manufacturing Technology, 19, 192-208, 2002.

[45] Ghaffari, A., Rezg, N. and Xie, X.L., Design of a Live and Maximally Permissive Petri Net Controller Using the Theory of Regions, IEEE Trans. on Robotics and Automation, 19, 1, 137-142, 2003.

[46] Uzam, M., The Use of Petri Net Reduction Approach for an Optimal Deadlock Prevention Policy for Flexible Manufacturing Systems, The International Journal of Advanced Manufacturing Technology, 23, 3-4, 204-219, 2004.

[47] Reveliotis, S.A. and Choi J.Y., Designing Reversibility Enforcing Supervisors of Polynomial Complexity for Bounded Petri Nets Through the Theory of Regions, Lecture Notes in Computer Science, 4024, 322-341, 2006.

[48] Wonham, W. M., Supervisory Control Theory: Models and Methods, 24th International Conference on Applications and Theory of Petri Nets (ATPN 2003), Eindhoven, The Netherlands, 1-14, 2003.

[49] Ghaffari, A., Rezg, N. and Xie, X. L, Feedback Control Logic for Forbidden-State Problems of Marked Graphs: Application to a Real Manufacturing System, IEEE Trans. Automatic Control, 48, 1, 18-29, 2003.

[50] Uzam, M. and Wonham, W. M., Comments on “Feedback Control Logic for Forbidden-State Problems of Marked Graphs: Application to a Real Manufacturing System”, IEEE Transactions on Automatic Control, 49, 7, 1216-1217, 2004.

[51] Basile, F., Chiacchio, P. and Guia, A., Suboptimal Supervisory Control of Petri Nets in Presence of Uncontrollable Transitions via Monitor Places, Automatica, 42, 6, 995-1004, 2006.

[52] Uzam, M. and Wonham, W. M., A Hybrid Approach to Supervisory Control of Discrete Event Systems Coupling RW Supervisors to Petri Nets, The International Journal of Advanced Manufacturing Technology, 28,7-8, 747-760, 2006.

[53] Bolton, W., Programmable Logic Controllers, Newnes; 4th edition, 2006.

[54] John, K. H. and TiegelKamp, M., IEC 61131-3: Programming Industrial Automation Systems, Springer-Verlag, Berlin, Hieldberg, Newyork, 2001.

[55] Peng, S. S. and Zhou, M.C., Ladder Diagram and Petri Net Based Discrete Event Control Design Methods, IEEE Transaction on Systems, Man, and Cybernetics, Part C-Applications and Reviews, 34, 4, 523-531, 2004.

[56] Hellgren, A., Fabian, M. and Lennartson, B., On the Execution of Sequential Function Charts, Control Engineering Practice, 13, 10, 1283-1293, 2005.

[57] De Querioz, M.H. and Cury, J. E. R., Synthesis and Implementation of Local Modular Supervisory Control for a Manufacturing Cell, 6th International Workshop on Discrete Event Systems, 377-382, 2002.

[58] Jones, A. H., Uzam, M., Khan, A. H., Karimzadgan, D. and Kenway, S. B, A General Methodology for Converting Petri Nets Into Ladder Logic: The TPLL Methodology, 5th Int. Conf. on Computer Integrated Manufacturing and Automation Technology - CIMAT'96, France, 357-362, 1996.

[59] Uzam, M. and Jones A. H., Design of a Discrete Event Control System for a Manufacturing System Using Token Passing Ladder Logic, CESA'96 IMACS Multiconference, Symposium on Discrete Events and Manufacturing Systems, Lille, France, 513-518, 1996.

[60] Uzam M., Petri Net Based Supervisory Control of Discrete Event Systems and Their Ladder Logic Diagram İmplementations", PhD Thesis, University of Salford, UK. 1998.

[61] Uzam, M. and Jones, A. H., Discrete Event Control System Design Using Automation Petri Nets and Their Ladder Diagram Implementation, International Journal of Advanced Manufacturing Technology, 14, 10, 716-728, 1998.

[62] Lai, H. F. and Lee, C. E., A Hybrid Specification Method for the Design of a Workcell Controller in Manufacturing Systems, International Journal of Advanced Manufacturing Technology, 17, 12, 928-938, 2001.

[63] Lucas, M. R. and Tilbury, D. M., A Study of Current Logic Design Practices in The Automotive Manufacturing Industry, International Journal of Human-Computer Studies 59, 5, 725-753, 2003.

[64] Ramirez-Serrano, A., Zhu, S.C., Chan, S.K.H. and Chan, S.S.W., Ficocelli, M., and Benhabib B., A hybrid PC/PLC Architecture Manufacturing System Control Theory and Implementation, Journal of Intelligent Manufacturing, 13, 261-281, 2002.

[65] Karlık, B., Uzam, M., Cinsdikici, M. and Jones, A. H., Neurovision-Based Logic Control of an Experimental Manufacturing Plant Using Convolutional Neural Net Le-Net5 and Automation Petri Nets, Journal of Intelligent Manufacturing, 16, 4-5, 527-548, 2005.

[66] Gelen, G., ve Uzam, M., Ayrık Olay Sistemlerinin R&W Yöntemi ve Petri Ağları Temelli Melez Kontrolü, TOK 50nci Yıl, Otomatik Kontrol Ulusal Toplantısı, İTÜ - İstanbul, 719-724, 2008.

[67] Uzam, M. and Gelen, G., The Real-time Supervisory Control of an Experimental Manufacturing System Based on a Hybrid Method, Control Engineering Practice, 17, 10, 1174-1189, 2009.

[68] TCT, A software tool implementing Ramadge-Wonham's Supervisory Control Theory. Systems Control Group, ECE Dept., University of Toronto, Posted at URL: www.control.utoronto.ca/DES 2010.

[69] Yarımağan, Ü., Özdevinirler (Otomatlar) Kuramı ve Biçimsel Diller, Bıçaklar Kitapevi, 2003.

[70] Uzam, M., Gelen, G. and Dalcı, R., A New Approach for the Ladder Logic Implementation of Ramadge-Wonham Supervisors, 22nd International Symposium on Information, Communication and Automation Technologies-ICAT 2009, Sarajevo, Bosnia and Herezegovina, 113-119, October 29-31, 2009.

[71] Wonham, W. M., Notes on Control of Discrete Event Systems, ECE Dept. University of Toronto. http://www.control.utoronto.ca/DES .2010.

[72] Peterson, J. L., Petri Net Theory and the Modeling of Systems, Prentice-Hall Inc., Englewood Cliffs, N.J., 1981.

[73] Marsan, M. A., Balbo, G., Conte, G., Donatelli, S. and Francesshinis, G., Modelling with Generalized Stochastic Petri Nets, John Wiley and Sons Ltd., Turin, 1995.

[74] Krogh, B. H., Controlled Petri Nets and Maximally Permissive Feedback Logic, 25th. Annnual Allerton Conference, USA, 1987.

[75] Ichikawa, A. and Hiraishi, K., Analysis and Control of Discrete Event System Represented by Petri Nets, IIASA Conference Sopron, Hungry, 1987.

[76] Holloway, L. E. and Krogh, B. H., Controlled Petri Nets: a Tutorial Survey, Lecture Notes in Control and Information Sciences, 199, Springer Verlag, London, 158-168, 1994.

[77] Uzam, M., Gelen, G. and Dalcı, R., An Effective Method for Detection and Elimination of Avalanche Effect Problem", 2010 IEEE International Conference on Mechatronics and Automation (ICMA 2010), Xi'an, China, 1988-1993, August 4-7, 2010.

[78] Fabian, M. and Hellegren, A., PLC-based Implementation of Supervisory Control for Discrete Event Systems, 37th IEEE Conference on Decision and Control, Tampa, Florida, USA, 3305-3310, 1998.

[79] Lauzon, C., Mills, J. K. and Benhabib, B., An Implementation Methodology for the Supervisory Control of Flexible Manufacturing Workcells, Journal of Manufacturing Systems, 16, 2, 91-101, 1997.

[80] Brandin, B.A., The Real-Time Supervisory Control of an Experimental Manufacturing Cell, IEEE Trans. Robotics and Automatation, 12, 1, 1–14, 1996.

[81] Ramirez-Serrano, A., Zhu, S. C. and Benhabib, B., Moore Automata for the Supervisory Control of Robotic Manufacturing Workcells, Autonomous Robots, 9, 59-69, 2000.

[82] De Querioz, M. H. and Cury, J. E. R, Modular Supervisory Control of Large Scale Discrete Event System, Workshop on Discrete Event System, 2000.

[83] Gouyon, D., Petin, J. F. and Gouin, A., Pragmatic Approach for Modular Synthesis and Implementation, International Journal of Production Research, 42, 14, 2839-2858, 2004.

[84] Liu, J. and Darabi, H., Ladder Logic Implementation of Ramadge-Wonham Supervisory Controller, 6th International Workshop on Discrete Event Systems, Zaragosa, Spain, 383-389 2002.

[85] Moniruzzaman, M. and Gohari, P., Implementing Supervisory Control Maps with PLC, American Control Conference, Newyork City, USA, 3594-3599, 2007.

[86] Manesis, S. and Akantziotis, K., Automated Synthesis of Ladder Automation Circuits Based on State-Diagrams, Advances in Engineering Software, 36, 225-233, 2005.

[87] Vieria, A. D. Cury, J. E. R. and De Queiroz, M. H., A Model for Implementation of Supervisory Control of a Discrete Event Systems, IEEE Conference on Emerging Technologies and Factory Automation, ETFA'2006, 225-232, 2006.

[88] Dalcı, R., Ramadge-Wonham Yöntemi Kullanılarak Elde Edilen Bir Gözeticinin PLC'lere Yüklenmek Üzere Kontrol Koduna Dönüştürülmesi İçin Genel Bir Yöntem, Yüksek Lisans Tezi, Niğde Üniversitesi Fen Bilimleri Enstitüsü, 2010.

[89] Hasdemir, İ.T., Kurtulan, S. and Gören, L., An Implementation Methodology for Supervisory Control Theory, International Journal of Advanced Manufacturing Technology, 36, 373-385, 2008.

[90] Gelen, G. and Uzam, M., Novel Analysis of Petri Net Based Controllers by Means of TCT Implementation Tool of Supervisory Control Theory, Maejo International Journal of Science and Technology, 4, 3, 360-396, 2010.

[91] Guia, A., and DiCesare, F., Supervisory Design Using Petri Nets, 30th Conf. on Decision and Control, Brighton-England, 92 – 97, 1991.

[92] Guia, A., DiCesare, F. and Silva, M., Generalized Mutual Exclusion Constraints on Nets with Uncontrollable Transitions, IEEE International Conference on Systems, Man and Cybernetics, Chicago, Illiois, USA, 974-979, 1992.

[93] Moody, J. O. and Antsaklis P. J., Petri Net Supervisors for Discrete Event Systems with Uncontrollable and Unobservable Transitions, IEEE Trans. Automatic Control, 45, 462-476, 2000.

[94] Uzam, M., On Suboptimal Supervisory Control of Petri Nets in the Presence of Uncontrollable Transitions via Monitor Places, Int. J. Advanced Manufacturing Technology, 47, 567-579, 2010.

[95] Iordache, M. V. and Antsaklis P. J., Supervisory Control of Concurrent Systems, A Petri Net Structural Approach, Birkhauser, Boston, P. 21, 2006.

[96] Darondeau, P., Equality of Languages Coincides with Isomorphism of Reachable State Graphs for Bounded and Persistent Petri Nets, Inform. Process. Lett., 94, 241-245, 2005.

[97] Droste, M. and Short, R. M., From Petri Nets to Automata with Concurrency, Appl. Categor. Struct., 10, 173-191, 2002.

[98] Egri- Nagy A. and Nehaniv C. L., Algebraic Properties of Automata Associated to Petri Nets and Applications to Computation in Biological Systems, BioSystems, 94, 135-144, 2008.

[99] Su, R. and Wonham W. M., Supervisor Reduction for Discrete Event Systems, Discrete Event Dynamic Systems Theory and Application, 14, 1, 31-53, 2004.

[100] Wong, K. C. and Wonham W. M., Hierarchical Control of Discrete Event Systems, Discrete Event Dynamic Systems Theory and Application, 241-273, 1996.

[101] Leduc, R. J., Lawford, M. and Wonham, W. M., Hierarchical Interface Based Supervisory Control-Part I: Serial Case, IEEE Trans. On Automatic Control, 50, 9, 1322-1335, 2005.

[102] Leduc, R. J., Lawford, M. and Wonham W. M., Hierarchical Interface based Supervisory Control-Part II: Parallel Case, IEEE Trans. on Automatic Control, 50, 9, 1336-1348, 2005.

[103] Schimidth, K., Reger, J. and Moor T., Hierarchical Control of structural DES, Workshop on Discrete Event Systems, 2004.

[104] Schimidth, K., Moor, T. and Perk, S., A Hierarchical Architecture for Nonbloking Control of Discrete Event Systems, Mediterranean Conference on Control and Automation, 2005.

[105] Rudie, K. and Wonham W. M., Think Globally, Act locally: Decentralized Supervisory Control, IEEE Trans. On Automatic Control, 37, 11, 1992.

[106] Lee, S-H. and Wong, K. C, Structural Decentralized Control of Concurrent DES, European Journal of Control, 35, 1125-1134, 2002.

[107] Wu, W., Dong, L., Wang, X., Su, H. and Chu J., Combined Petri Net Controller for Discrete Event Systems, Acta Automatica Sinica, 29, 5, 681–688, 2003.

[108] Luo, J., Jin, F. and Huang C., Combined Supervisor Synthesis for Enforcing GMECs on a Controlled Petri Net, 2007 IEEE International Conference on Integration Technology, Shenzhen, China, 633-38, 2007.

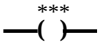
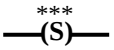
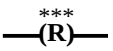
[109] Holloway, L.E., Krogh B. H. and Guia A., A Survey of Petri Net Methods for Controlled Discrete Event Systems, Discrete Event Dynamic Systems: Theory and Applications, 7, 151–190, 1997.

[110] Boel, R. K., Ben-Naoum, L. and Breusegem, V. V., On Forbidden-State Problems for a Class of Controlled Petri Pets, IEEE Transactions on Automatic Control, 40, 10, 1717–1731, 1995.

[111] Zhou, M. C. and Jeng, MD, Modeling, Analysis, Simulation, Scheduling, and Control of Semiconductor Manufacturing Systems: a PN Approach, IEEE Trans. Semicon. Manuf., 11, 3, 333-357, 1998.

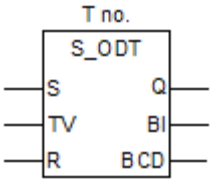
[112] Lecture notes on Control of Discrete Event Systems, Abo Akademi University, Finland shttps://www.abo.fi/student/en/tkfrtcdes, 2010.

EK-A
IEC 61131-3 Merdiven Diyagram Standardı Sembolleri

Sembol	Açıklama
I	*Giriş bölgesi*
Q	*Çıkış bölgesi*
M	*Hafıza bölgesi*
	*Normalde açık kontak* Eğer ilişkilendirilmiş Boole değişkeninin (*** ile gösterilen) durumu ON ise sol taraftaki bağlantının durumu sağ traftaki bağlantıya aktarılır. Diğer durumlarda sağ taraftaki bağlantının durumu OFF olur.
	*Normalde kapalı kontak* Eğer ilişkilendirilmiş Boole değişkeninin (*** ile gösterilen) durumu OFF ise sol taraftaki bağlantının durumu sağ traftaki bağlantıya akatarılır. Diğer durumlarda sağ taraftaki bağlantınun durumu OFF olur.
	*Positif geçiş algılama kontağı* Bu kontağın bir değerlendirilmesinden bir sonraki değerlendirilmesine kadar sol taraftaki bağlantı ON ise ilişkilendirilmiş Boole değişkeninin (*** ile gösterilen) durumunun OFF'tan ON'a değişiminin algılanması ile aynı zamanda sağ taraftaki bağlantının durumu ON olur. Diğer tüm durumlarda sağ taraftaki bağlantının durumu OFF olur.
	*Negatif geçiş algılama kontağı* Bu kontağın bir değerlendirilmesinden bir sonraki değerlendirilmesine kadar sol taraftaki bağlantı ON ise ilişkilendirilmiş Boole değişkeninin (*** ile gösterilen) durumunun ON dan OFF a değişiminin algılanması ile aynı zamanda sağ taraftaki bağlantının durumu ON olur. Diğer tüm durumlarda sağ taraftaki bağlantının durumu OFF olur.
	*Bobin* Sol taraftaki bağlantının durumu ilişkilendirilmiş Boole değişkenine (*** ile gösterilen) kopyalanır.
	*Bobini SET et (latch – kilitle) komutu* Sol taraftaki bağlantının drumunun ON olduğu an ilişkilendirilmiş (*** ile gösterilen) Boole değişkeninin durumu ON yapılır. Reset bobini ile OFF yapılana kadar ON durumunda kalır.
	*Bobini RESET et (unlatch – kilidini aç) komutu* Sol taraftaki bağlantının durumunun ON olduğu an ilişkilendirilmiş (*** ile gösterilen) Boole değişkeninin durumu OFF yapılır. Set bobini ile ON yapılana kadar OFF durumunda kalır.

ON = 1 (HIGH), OFF = 0 (LOW)

PLC merdiven diyagramı kodlarında kullanılan Siemens S7-300 PLC'nin on delay timer elementi Şekil E1.1'de görülmektedir.



Şekil E.1.1 Siemens S7 300 PLC'nin düz zaman rölesi (On Delay Timer-ODT) elementi

ODT, başlama (S) girişine bir sinyalin yükselen kenarı geldiğinde çalışmaya başlar. Zamanlayıcının etkinleştirilmesinde her zaman için bir sinyal değişimi gereklidir. S girişindeki sinyal durumunun 1 olması boyunca TV girişinde belirtilen zaman aralığını geçirmek için zamanlayıcı sayar. Belirtilen zaman geçtiğinde ve herhangi bir hata olmaksızın S girişindeki sinyal durumu '1' olarak durması halinde Q çıkışındaki sinyal durumu 1 olmaktadır. Zamanlayıcı sayarken S girişindeki sinyal durumu '1'den '0'a değişirse zamanlayıcı duracaktır. Bu durumda Q çıkışının sinyal durumu '0' olacaktır. Zamanlayıcı saymaktayken zamanlayıcıyı yeniden kurma girişi (R) '0'dan '1'e değişirse, bundan önce sayılan zaman değeri sıfırlanır. Q çıkış sinyali de '0'a kurulur. Sayılan zaman değeri BI ve BCD çıkışlarından gözlenebilir. BI'da zaman değeri ikilik sayı sistemiyle (binary kod), BCD'de BCD kodunda gözlenir.

EK-B
"Av_effect_detector.exe" Programı Kaynak Kodu

```
   #include <stdio.h>
   #include <iostream>
   #include <math.h>
   #include <list>
   #include <conio.h>
   using namespace std;

int main()
{
   FILE *fp,*fp1;
   printf( "\n");
   printf("-------------------------------------------------------------------------\n ");
   printf("     DETECTION of EVENTS THAT CAUSE AVALANCHE EFFECT \n");
   printf("              Copyright 2010 Gokhan GELEN\n\n");
   printf("-------------------------------------------------------------------------\n ");

start:
   int n_ssts=0,n_trans=0,num_cev=0;
   int linecount=0,i=0,j=0,k=0,crr_events=0,Crr_s_sts=0,Crr_t_sts=0,Crr_event,a=0;
   int q=0,w=0,r=0;
   char c,filename[80];
   int In_state[100],Out_state[100];
   int aval=0;
   int Events[100];
   int C_events[100],CP_MARK[100],all_events[100],inds=0;
   char control;

   for (int x=0; x<100; x++){
      In_state[x]=0;
      Out_state[x]=0;
      Events[x]=0;
      C_events[x]=0;
      CP_MARK[x]=0;
      all_events[x]=0;}

   printf( "\n");
   printf("Please enter the name of .ads file: ");
   printf("\n--> ");
         gets( filename );
   strcat (filename,".ads");

         fp = fopen( filename, "r" );

         while ( fp == NULL ){
                  printf("\n!!! Cannot open %s for reading !!! \n", filename );
                  printf("Please enter again the name of .ads file:  ");
                  printf("\n--> ");
                  gets( filename );
                  strcat (filename,".ads");
      fp = fopen( filename, "r" );}

   c = getc( fp ) ;
         while ( c != EOF){
         c = getc( fp ) ;
         if (c=='\n'){
            linecount=linecount+1;
   if (linecount == 6){
```

```
        fscanf(fp,"%d" , &n_ssts);}
    else if (linecount >= 27){
                        fscanf(fp,"%d %d %d",&q,&w,&r);
        all_events[inds]=w;
        inds=inds+1;
        if(q!=r){
        In_state[i]=q;
        Events[i]=w;
        Out_state[i]=r;
        i=i+1;}}}
    fclose( fp );
    n_trans=i-1;

    list<int> mylist (all_events,all_events+20);

    mylist.sort();
    mylist.unique();

    for (list<int>::iterator it=mylist.begin(); it!=mylist.end(); ++it){
        if(*it!=0){
        C_events[num_cev]= *it;
        num_cev++;}
    }

    i=0;
    k=0;
    j=0;

labelx:
    Crr_event=C_events[i];
labelxxx:
    if (Crr_event==Events[j]){
    Crr_s_sts=In_state[j];
    Crr_t_sts=Out_state[j];
labelxx:
        if(((Crr_s_sts==Out_state[k])||(Crr_t_sts==In_state[k]))&&(Crr_event==Events[k])){
        CP_MARK[a]=C_events[i];
        a=a+1;
        goto research;}
    k=k+1;
    if (k<n_trans) goto labelxx;
}
    j=j+1;
    if (j<n_trans) goto labelxxx;
research:
    k=0;
    j=0;
    i=i+1;
    if (i<num_cev) goto labelx;

//---------- OUTPUS to SCREEN

    printf("\n\nLABELS of ALL EVENTS: \n\n ");
    for (int x=0; x<num_cev; x++){
        printf("%3d",C_events[x]);
    }
    printf("\n\nLABELS of EVENTS THAT CAUSE AVALANCHE EFFECT: \n\n ");
    if(a==0)
        printf( " no event\n ");
    else
```

```
    for (int x=0; x<a; x++){
       printf("%3d",CP_MARK[x]);
    }

    fp1 = fopen( "logfile.txt", "a+" );

    fprintf(fp1, "%s",filename);
    fprintf(fp1, "-->\n");
    fprintf(fp1, "All events={");
    for (int x=0; x<num_cev; x++){
       fprintf(fp1, "%3d",C_events[x]);
    }
    fprintf(fp1, " }\n");

    fprintf(fp1, "Events that cause avalanche effect={");
    if(a==0)
       fprintf(fp1, " no event ");
    else
    for (int x=0; x<a; x++){
       fprintf(fp1,"%3d",CP_MARK[x]);
    }
    fprintf(fp1, " }\n\n");
    fclose(fp1);

    loop:
    printf(" \n\n Press R for new computation");
    printf(" \n Press Q for new quit");
    printf("\n--> ");
    //scanf ("%s",&control);
    control=getch();
    if (control=='r'|control=='R')
    goto start;
    else if (control=='q'|control=='Q')
    exit(1);
    else
    goto loop;

}
```

EK-C
“cp.exe” Programı Kaynak Kodu

```
   #include <stdio.h>
   #include <iostream>
   #include <math.h>
   #include <list>
   using namespace std;
   #include <conio.h>

int main()
{
          int WS[100][100],TRANS[100][3],TRANS1[100][3];
   int CD[100][100],CDT[100][100];
   int CP[100][100];
   int state[100][7];
   int Events[100];
   int C_events[100],Con_events[100],CP_MARK[100],IN_MRK[100];

          FILE *fp,*fp2,*fp3,*fp4;
   printf( "\n");
   printf("--------------------------------------------------------------------------\n ");
   printf("   SYNTHESIS of CONTROL POLICY for CONTROLLED PETRI NETS\n");
   printf("                Copyright 2010 Gokhan GELEN\n\n");
   printf("--------------------------------------------------------------------------\n ");

start:

          int d1=0,d2=0,d3=0,d4=0,d5=0,d6=0,d7=0,d8=0,d9=0,d10=0;
          int n_ssts=0,n_trans=0,n_cevnt=0,num_cev=0,n_events=0;
   int s0=0,s1=0,s2=0,s3=0,s4=0,s5=0,s6=0,s7=0;
          int linecount=0,linecount2=0,linecount3=0,ai=0,i=0,a=0,b=0,d=0,temp=0,kontrol=0;
   int q=0,w=0,r=0,k=0,n=0,j=0,g=0,q1=0,w1=0,r1=0;
   int index=0,index2=0;
          char c=0,c1=0,c2=0,ctemp=0,filename[80], reply[40];
   char filename1[80],filename2[80];
   char control,string[100];

   for (int x=0; x<100; x++){
      C_events[x]=0;
      Con_events[x]=0;
      CP_MARK[x]=0;
      IN_MRK[x]=0;
      Events[x]=0;
   for (int y=0; y<100; y++){
      WS[x][y]=0;
      CD[x][y]=0;
      CDT[x][y]=0;
      CP[x][y]=0;}
   for (int z=0; z<7; z++){
      state[x][z]=0;
   for (int g=0; g<3; g++){
      TRANS[x][g];
      TRANS1[x][g]; }}
   }

   printf("\nPlease enter the name of Event List file:");
   printf("\n--> ");
          gets( filename );
   strcat (filename,".ads");
```

```
        fp4 = fopen( filename, "r" );

        while ( fp4 == NULL ){
                printf("\n!!! Cannot open %s for reading !!! \n", filename );
                printf("Please again enter the file name:  ", filename );
   printf("\n--> ");
   gets( filename );
   strcat (filename,".ads");
   fp4 = fopen( filename, "r" );
   }
c = getc( fp4 ) ;
        while ( c != EOF){
        c = getc( fp4 ) ;
        if (c=='\n'){
           linecount3=linecount3+1;
if (linecount3 >= 27){
                fscanf(fp4,"%d %d %d",&q1,&w1,&r1);
   Events[n_events]=w1;
   n_events=n_events+1;}
   }
   }
fclose(fp4);
n_events=n_events-1;

printf("\nPlease enter the name of Supervisor file:");
printf("\n--> ");
        gets( filename );
strcat (filename,".ads");
        fp = fopen( filename, "r" );

        while ( fp == NULL ){
                printf("\n!!! Cannot open %s for reading !!! \n", filename );
                printf("Please again enter the file name:  ", filename );
   printf("\n--> ");
   gets( filename );
   strcat (filename,".ads");
   fp = fopen( filename, "r" );
   }
c = getc( fp ) ;
        while ( c != EOF){
        c = getc( fp ) ;
        if (c=='\n'){
           linecount=linecount+1;
if (linecount == 6){
   fscanf(fp,"%d" , &n_ssts);}
else if (linecount >= 27){
                fscanf(fp,"%d %d %d",&q,&w,&r);
   TRANS[ai][1]=q;
   TRANS[ai][2]=w;
   TRANS[ai][3]=r;
   ai=ai+1;}
   }
   }
        strcpy(filename1,filename);
fclose(fp);

for (int i=0; i<ai-1; i++){
   if(TRANS[i][1]!=TRANS[i][3]){
   WS[TRANS[i][1]][d10]=-1;
   WS[TRANS[i][3]][d10]=1;
```

```
   TRANS1[d10][1]=TRANS[i][1];
   TRANS1[d10][2]=TRANS[i][2];
   TRANS1[d10][3]=TRANS[i][3];
   d10=d10+1;}
   }
   n_trans=d10;

for(int d1=0;d1<n_events; d1++){
   if (Events[d1]%2==1){
   C_events[num_cev]=Events[d1];
   num_cev++;
}
}

printf("\nPlease enter the name of Control Data file:");
printf("\n--> ");
gets( filename );
strcat (filename,".pdt");
        fp2 = fopen( filename , "r" );

        while ( fp2 == NULL ){
                printf("\n!!! Cannot open %s for reading !!! \n", filename );
                printf("Please again enter the file name:", filename );
   printf("\n--> ");
   gets( filename );
   strcat (filename,".pdt");
   fp2 = fopen( filename, "r" );
   }

c1 = getc( fp2 ) ;
        while ( c1 != EOF){
        c1 = getc( fp2 ) ;
        if (c1=='\n'){
           linecount2=linecount2+1;
if (linecount2 >= 10 ){
  for (int d6=0; d6<7; d6++){
  fscanf(fp2,"%d%c",&temp,&c1 );
   if (c1 ==':'&& kontrol==0){
   state[d5][d6]=temp;
   temp=0;
   kontrol=1;}
   else if (c1 ==':'&& kontrol==1){
   d6=0;
   d5=d5+1;
   state[d5][d6]=temp;
   temp=0;}
   else if (c1 !=':'){
   state[d5][d6]=temp;
   temp=0;}
   }
  }
 }
        }
        strcpy(filename2,filename);
fclose( fp2);
for (int x=0; x<d5+1;x++){
s0=state[x][0];
for (int z=1; z<7;z++){
s1=state[x][z];
for (int y=0; y<num_cev; y++){
```

```
        if (s1==C_events[y]){
           CD[y][s0]=1;}}
    }

     for(int i=0;i<num_cev;i++)
     {
     for(int j=0;j<n_trans;j++)
     {
      for(k=0;k<n_ssts;k++){
         CP[i][j] = CP[i][j] + CD[i][k]*WS[k][j];}
     }
    for(int i=0;i<num_cev;i++){
     for(int j=0;j<n_trans;j++)
     {
      CP[i][j] = (-1)*CP[i][j];}
     }}

for(int i=0;i<num_cev;i++){
   if(CD[i][0]==1){
      CP_MARK[i]=0;}
   else if (CD[i][0]==0){
   CP_MARK[i]=1;}
   }

//---------- OUTPUTS to SCREEN
   printf("\n----------------------------------------------------------------------------\n");
   printf("Computation for Supervisor:%s  and Control data:%s\n",filename1,filename2);
   printf("----------------------------------------------------------------------------\n");
   printf("\n-->Transition Coding for Supervisor:%s \n",filename1);
   printf( "%c" ,'\n');
   printf("Transition name       TCT Coding\n");
   printf("----------------      ----------\n");
   for (int x=0; x<n_trans; x++){
   printf("ts%d              -->  [%d %d %d]\n",x+1,TRANS1[x][1],TRANS1[x][2],TRANS1[x][3]);
   }
   printf("\n-->Incedence Matrix (WS) of Supervisor:%s \n",filename1);
   printf( "%c" ,'\n');
   for (int x=0; x<n_ssts; x++){
      for (int y=0; y<n_trans;y++){
      printf("%2d ",WS[x][y]);
   }
    printf( "%c" ,'\n');
   }
   printf("\n-->Control Data (%s) in Matrix Form\n\n",filename2);
   for (int x=0; x<num_cev; x++){
      for (int y=0; y<n_ssts;y++){
      CDT[y][x]=CD[x][y];
      }
   }

   for (int x=0; x<n_ssts; x++){
      for (int y=0; y<num_cev;y++){
      printf("%2d ",CDT[x][y]);
      }
    printf("%c",'\n');
   }
   printf("\n-->Control Policy \n\n");
   for (int x=0; x<num_cev; x++){
      for (int y=0; y<n_trans;y++){
      printf("%2d ",CP[x][y]);
```

```
        }
     printf("%c",'\n');}

    printf("\n\n-->Initial Marking of Control Places \n\n");
    printf("Control Place     Number of tokens\n ");
    printf("%c",'\n');
    for (int x=0; x<num_cev; x++){
       printf( "      c%d        %10d\n", C_events[x], CP_MARK[x]);}

    printf("\n-->Control Policy in Informal Form\n\n");
    for (int x=0; x<num_cev; x++){
    printf("Control Place     :   c%d",C_events[x]);
    printf("\n");
    printf("Pre Transitions   :   ");
       for (int y=0; y<n_trans;y++){
       if (CP[x][y]==1)
       printf( "ts%d ",y+1);
    }
       printf( "\n");

    printf("Post Transitions  :   ");
       for (int y=0; y<n_trans;y++){
       if (CP[x][y]==-1)
       printf( "ts%d ",y+1);
       }
    printf( "\n");
    printf("Initial Marking   :   ");
    printf( "%d", CP_MARK[x]);
    printf( "\n\n");
    }

printf("--------------------------------------------------------------------------------");
//---------- OUTPUTS to LOGFILE
    fp3 = fopen( "logfile.txt", "a+" );
    fprintf(fp3,"\n--------------------------------------------------------------------------------\n");
    fprintf(fp3,"Computation for Supervisor:%s  and Control data:%s\n",filename1,filename2);
    fprintf(fp3,"--------------------------------------------------------------------------------\n");
    fprintf(fp3,"\n-->Transition Coding for Supervisor:%s \n",filename1);
    fprintf(fp3,"%c" ,'\n');
    fprintf(fp3,"Transition name       TCT Coding\n");
    fprintf(fp3,"----------------      ----------\n");
    for (int x=0; x<n_trans; x++){
    fprintf(fp3,"ts%d              -->  [%d %d %d]\n",x+1,TRANS1[x][1],TRANS1[x][2],TRANS1[x][3]);
    }
    fprintf(fp3,"\n-->Incedence Matrix (WS) of Supervisor:%s \n",filename1);
    fprintf(fp3,"%c" ,'\n');
    for (int x=0; x<n_ssts; x++){
       for (int y=0; y<n_trans;y++){
       fprintf(fp3,"%2d ",WS[x][y]);
    }
     fprintf(fp3,"%c" ,'\n');
    }
    fprintf(fp3,"\n-->Control Data (%s) in Matrix Form\n\n",filename2);
    for (int x=0; x<n_ssts; x++){
       for (int y=0; y<num_cev;y++){
       fprintf(fp3,"%2d ",CDT[x][y]);
       }
     fprintf(fp3,"%c",'\n');
    }
```

```
    fprintf(fp3,"\n-->Control Policy \n\n");
    for (int x=0; x<num_cev; x++){
       for (int y=0; y<n_trans;y++){
       fprintf(fp3,"%2d ",CP[x][y]);
       }
     fprintf(fp3,"%c",'\n');
    }
    fprintf(fp3,"\n\n-->Initial Marking of Control Places \n\n");
    fprintf(fp3,"Control Place     Number of tokens\n ");
    fprintf(fp3,"%c",'\n');
    for (int x=0; x<num_cev; x++){
       fprintf(fp3,"      c%d        %10d\n", C_events[x], CP_MARK[x]);
    }
    fprintf(fp3,"\n-->Control Policy in Informal Form\n\n");
    for (int x=0; x<num_cev; x++){
    fprintf(fp3,"Control Place     :   c%d",C_events[x]);
    fprintf(fp3,"\n");
    fprintf(fp3,"Pre Transitions   :   ");
       for (int y=0; y<n_trans;y++){
       if (CP[x][y]==1)
       fprintf(fp3,"ts%d ",y+1);
    }
fprintf(fp3,"\n");
      fprintf(fp3,"Post Transitions  :   ");
       for (int y=0; y<n_trans;y++){
       if (CP[x][y]==-1)
       fprintf(fp3,"ts%d ",y+1);
       }
    fprintf(fp3,"\n");
    fprintf(fp3,"Initial Marking   :   ");
    fprintf(fp3,"%d", CP_MARK[x]);
    fprintf(fp3,"\n\n");
    }
fprintf(fp3,"--------------------------------------------------------------------------------");
fclose( fp3 );
loop:
    printf(" \n\n Press R for new computation");
    printf(" \n Press Q for new quit");
    printf("\n--> ");
    control=getch();
    if (control=='r'|control=='R')
    goto start;
    else if (control=='q'|control=='Q')
    exit(1);
    else
    goto loop;
}
```

EK-D

Halka Toplama Kolu Kapasitesinin 1 Olduğu Durumda Elde Edilen Yekpare Gözetici İçin Hesaplanan Kontrol Poliçesine Ait Program Çıktısı

```
--------------------------------------------------------------------------------
Computation for Supervisor:super.ads  and Control data:supdat.pdt
--------------------------------------------------------------------------------

-->Transition Coding for Supervisor:super.ads

Transition name       TCT Coding
---------------      ----------
ts1             -->  [0 22 1]
ts2             -->  [1 12 0]
ts3             -->  [1 61 2]
ts4             -->  [2 32 3]
ts5             -->  [3 22 4]
ts6             -->  [3 71 5]
ts7             -->  [4 12 3]
ts8             -->  [4 71 6]
ts9             -->  [5 22 6]
ts10             -->  [5 42 7]
ts11             -->  [6 12 5]
ts12             -->  [6 42 8]
ts13             -->  [7 22 8]
ts14             -->  [7 52 0]
ts15             -->  [8 12 7]
ts16             -->  [8 52 1]
ts17             -->  [8 61 9]
ts18             -->  [9 32 10]
ts19             -->  [9 52 2]
ts20             -->  [10 22 11]
ts21             -->  [10 52 3]
ts22             -->  [11 12 10]
ts23             -->  [11 52 4]

-->INCEDENCE MATRIX (WS) of Supervisor:super.ads

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

-->CONTROL DATA (supdat.pdt) in MATRIX FORM

 1  1
 0  1
 0  1
 1  0
 1  0
 1  0
```

1 0
1 1
0 1
0 1
1 1
1 1

-->CONTROL POLICY

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

-->INITIAL MARKING of CONTROL PLACE

Control Place Number of tokens

c61 0
c71 0

-->CONTROL POLICY in INFORMAL FORM

Control Place : c61
Pre Transitions : ts1 ts12 ts13
Post Transitions : ts2 ts4 ts15 ts18
Initial Marking : 0

Control Place : c71
Pre Transitions : ts4 ts21 ts23
Post Transitions : ts10 ts12
Initial Marking : 0

EK-E

Halka Toplama Kolu Kapasitesinin 2 Olduğu Durumda Elde Edilen Yekpare Gözetici İçin Hesaplanan Kontrol Poliçesine Ait Program Çıktısı

```
--------------------------------------------------------------------------------
Computation for Supervisor:sup.ads  and Control data:condat.pdt
--------------------------------------------------------------------------------

-->Transition Coding for Supervisor:sup.ads

Transition name       TCT Coding
---------------       ----------
ts1             -->  [0 22 1]
ts2             -->  [1 12 0]
ts3             -->  [1 61 2]
ts4             -->  [2 32 3]
ts5             -->  [3 22 4]
ts6             -->  [3 82 5]
ts7             -->  [4 12 3]
ts8             -->  [4 82 6]
ts9             -->  [5 22 6]
ts10             -->  [5 71 7]
ts11             -->  [6 12 5]
ts12             -->  [6 61 8]
ts13             -->  [6 71 9]
ts14             -->  [7 22 9]
ts15             -->  [7 42 10]
ts16             -->  [8 32 11]
ts17             -->  [8 71 12]
ts18             -->  [9 12 7]
ts19             -->  [9 42 13]
ts20             -->  [9 61 12]
ts21             -->  [10 22 13]
ts22             -->  [10 52 0]
ts23             -->  [11 22 14]
ts24             -->  [11 71 15]
ts25             -->  [12 32 15]
ts26             -->  [12 42 16]
ts27             -->  [13 12 10]
ts28             -->  [13 52 1]
ts29             -->  [13 61 16]
ts30             -->  [14 12 11]
ts31             -->  [14 71 17]
ts32             -->  [15 22 17]
ts33             -->  [15 42 18]
ts34             -->  [16 32 18]
ts35             -->  [16 52 2]
ts36             -->  [17 12 15]
ts37             -->  [17 42 19]
ts38             -->  [18 22 19]
ts39             -->  [18 52 3]
ts40             -->  [18 82 20]
ts41             -->  [19 12 18]
ts42             -->  [19 52 4]
ts43             -->  [19 82 21]
ts44             -->  [20 22 21]
ts45             -->  [20 52 5]
ts46             -->  [21 12 20]
```

ts47 --> [21 52 6]
ts48 --> [21 61 22]
ts49 --> [22 32 23]
ts50 --> [22 52 8]
ts51 --> [23 22 24]
ts52 --> [23 52 11]
ts53 --> [24 12 23]
ts54 --> [24 52 14]

-->INCEDENCE MATRIX (WS) of Supervisor:sup.ads

```
-1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 1 -1 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 1 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 1 -1 -1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 1 0 -1 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 1 0 0 -1 -1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 1 1 0 -1 -1 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 1 0 0 0 -1 -1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 -1 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 -1 -1 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 -1 -1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 -1 -1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 -1 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 -1 -1 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 -1 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 -1 -1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 -1 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 -1 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 -1 -1 -1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 -1 -1 -1 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 -1 -1 1 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 -1 -1 -1 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 -1 -1 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 -1 -1 1 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 -1 -1
```

-->CONTROL DATA (condat.pdt) in MATRIX FORM

```
1 1
0 1
0 1
1 1
1 1
1 0
0 0
1 0
0 0
0 0
1 1
1 0
0 0
0 1
1 0
1 0
0 1
1 0
1 1
1 1
1 1
0 1
0 1
1 1
1 1
```

-->CONTROL POLICY

```
1 -1 0 -1 0 0 0 1 1 0 -1 0 0 1 0 -1 0 -1 0 0 1 0 0 0 -1 0 -1 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 1 1 0 -1 0 0 -1 0 0 0 0
0 0 0 0 0 1 0 1 0 0 0 0 0 0 -1 0 0 0 -1 0 0 0 0 0 0 -1 0 0 0 0 0 0 -1 0 0 0 -1 0 0 0 0 0 0 0 1 0 1 0 0 1 0 1 0 1
```

-->INITIAL MARKING of CONTROL PLACE

Control Place Number of tokens

c61	0
c71	0

-->CONTROL POLICY in INFORMAL FORM

Control Place : c61
Pre Transitions : ts1 ts8 ts9 ts14 ts21 ts43 ts44
Post Transitions : ts2 ts4 ts11 ts16 ts18 ts25 ts27 ts34 ts46 ts49
Initial Marking : 0

Control Place : c71
Pre Transitions : ts6 ts8 ts45 ts47 ts50 ts52 ts54
Post Transitions : ts15 ts19 ts26 ts33 ts37
Initial Marking : 0

EK-F

Halka Toplama Kolu Kapasitesinin 1 Olduğu Durumda Hesaplanan İndirgenmiş Yekpare Gözeticinin Kontrol Poliçesine Ait Program Çıktısı

```
--------------------------------------------------------------------------
Computation for Supervisor:red_sup.ads  and Control data:rsupdat.pdt
--------------------------------------------------------------------------

-->Transition Coding for Supervisor:red_sup.ads

Transition name       TCT Coding
---------------      ----------
ts1             -->  [0 12 4]
ts2             -->  [0 22 1]
ts3             -->  [0 32 2]
ts4             -->  [0 42 6]
ts5             -->  [0 52 3]
ts6             -->  [1 12 0]
ts7             -->  [1 32 4]
ts8             -->  [1 52 0]
ts9             -->  [1 61 0]
ts10             -->  [2 22 3]
ts11             -->  [2 71 4]
ts12             -->  [3 12 2]
ts13             -->  [3 71 0]
ts14             -->  [4 22 0]
ts15             -->  [4 42 5]
ts16             -->  [4 52 2]
ts17             -->  [5 22 6]
ts18             -->  [5 52 0]
ts19             -->  [6 12 5]
ts20             -->  [6 52 1]
ts21             -->  [6 61 1]

-->INCEDENCE MATRIX (WS) of Supervisor:red_sup.ads

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

-->CONTROL DATA (rsupdat.pdt) in MATRIX FORM

 1  1
 0  1
 1  0
 1  0
 1  1
 1  1
 0  1

-->CONTROL POLICY

 0  1  0  1  0 -1 -1 -1 -1  0  0  0  0  0  0  0  1  0 -1  0  0
 0  0  1  0  1  0  0  0  0  0 -1  0 -1  0  0  1  0  0  0  0  0
```

-->INITIAL MARKING of CONTROL PLACE

Control Place Number of tokens

c61 0
c71 0

-->CONTROL POLICY in INFORMAL FORM

Control Place : c61
Pre Transitions : ts2 ts4 ts17
Post Transitions : ts6 ts7 ts8 ts9 ts19
Initial Marking : 0

Control Place : c71
Pre Transitions : ts3 ts5 ts16
Post Transitions : ts11 ts13
Initial Marking : 0

EK-G
Halka Toplama Kolu Kapasitesinin 2 Olduğu Durumda Hesaplanan İndirgenmiş Yekpare Gözeticinin Kontrol Poliçesine Ait Program Çıktısı

```
----------------------------------------------------------------------------
Computation for Supervisor:redsup.ads  and Control data:rcdat.pdt
----------------------------------------------------------------------------


-->Transition Coding for Supervisor:redsup.ads

Transition name       TCT Coding
----------------      ----------
ts1              -->  [0 12 2]
ts2              -->  [0 22 1]
ts3              -->  [0 32 2]
ts4              -->  [0 42 5]
ts5              -->  [0 52 3]
ts6              -->  [0 82 4]
ts7              -->  [1 12 0]
ts8              -->  [1 42 7]
ts9              -->  [1 52 4]
ts10             -->  [1 61 0]
ts11             -->  [2 22 0]
ts12             -->  [2 42 8]
ts13             -->  [2 52 6]
ts14             -->  [2 82 3]
ts15             -->  [3 12 6]
ts16             -->  [3 22 4]
ts17             -->  [3 32 6]
ts18             -->  [3 71 0]
ts19             -->  [4 12 3]
ts20             -->  [4 61 3]
ts21             -->  [4 71 1]
ts22             -->  [5 12 8]
ts23             -->  [5 22 7]
ts24             -->  [5 32 8]
ts25             -->  [5 52 0]
ts26             -->  [5 82 1]
ts27             -->  [6 22 3]
ts28             -->  [6 71 2]
ts29             -->  [7 12 5]
ts30             -->  [7 52 1]
ts31             -->  [7 61 5]
ts32             -->  [8 22 5]
ts33             -->  [8 52 2]
ts34             -->  [8 82 0]

-->INCEDENCE MATRIX (WS) of Supervisor:redsup.ads

-1 -1 -1 -1 -1 -1  1  0  0  1  1  0  0  0  0  0  0  1  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  1
 0  1  0  0  0  0 -1 -1 -1 -1  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  1  0  0  0  1  0  0  0  0
 1  0  1  0  0  0  0  0  0  0 -1 -1 -1 -1  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  1  0
 0  0  0  0  1  0  0  0  0  0  0  0  0  1 -1 -1 -1 -1  1  1  0  0  0  0  0  0  1  0  0  0  0  0  0  0
 0  0  0  0  0  1  0  0  1  0  0  0  0  0  0  1  0  0 -1 -1 -1  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 -1 -1 -1 -1 -1  0  0  1  0  1  1  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  1  0  1  0  1  0  0  0  0  0  0  0  0  0 -1 -1  0  0  0  0  0  0
 0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0 -1 -1 -1  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  1  0  1  0  0  0  0  0  0  0 -1 -1 -1
```

-->CONTROL DATA (rcdat.pdt) in MATRIX FORM

```
1  1
0  1
1  1
1  0
0  0
1  1
1  0
0  1
1  1
```

-->CONTROL POLICY

```
0  1  0  0  0  1 -1  0  0 -1  0  0  0  0  0  1  0  0 -1 -1  0  0  1  0  0  1  0  0 -1  0 -1  0  0  0
0  0  0  0  1  1  0  0  1  0  0  0  1  1  0  0  0 -1  0  0 -1  0  0  0  0  0  0  0 -1  0  0  0  0  0  0
```

-->INITIAL MARKING of CONTROL PLACE

Control Place	Number of tokens
c61	0
c71	0

-->CONTROL POLICY in INFORMAL FORM

Control Place : c61
Pre Transitions : ts2 ts6 ts16 ts23 ts26
Post Transitions : ts7 ts10 ts19 ts20 ts29 ts31
Initial Marking : 0

Control Place : c71
Pre Transitions : ts5 ts6 ts9 ts13 ts14
Post Transitions : ts18 ts21 ts28
Initial Marking : 0

EK-H

Halka Toplama Kolu Kapasitesinin 1 Olduğu Durumda Hesaplanan Modüler Gözeticilerin Kontrol Poliçelerine Ait Program Çıktısı

```
---------------------------------------------------------------------------
Computation for Supervisor:modsup1.ads  and Control data:cdat1.pdt
---------------------------------------------------------------------------

-->Transition Coding for Supervisor:modsup1.ads

Transition name       TCT Coding
---------------      ----------
ts1              -->  [0 22 1]
ts2              -->  [1 12 0]
ts3              -->  [1 61 2]
ts4              -->  [2 32 3]
ts5              -->  [3 22 4]
ts6              -->  [3 42 0]
ts7              -->  [4 12 3]
ts8              -->  [4 42 1]

-->INCEDENCE MATRIX (WS) of Supervisor:modsup1.ads

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

-->CONTROL DATA (cdat1.pdt) in MATRIX FORM

 1
 0
 0
 1
 1

-->CONTROL POLICY

 1 -1  0 -1  0  0  0  1


-->INITIAL MARKING of CONTROL PLACE

Control Place     Number of tokens

     c61               0

-->CONTROL POLICY in INFORMAL FORM

Control Place     :   c61
Pre Transitions   :   ts1 ts8
Post Transitions  :   ts2 ts4
Initial Marking   :   0

---------------------------------------------------------------------------
```

```
--------------------------------------------------------------------------
Computation for Supervisor:modsup2.ads  and Control data:cdat2.pdt
--------------------------------------------------------------------------

-->Transition Coding for Supervisor:modsup2.ads

Transition name       TCT Coding
---------------      ----------
ts1             -->  [0 32 1]
ts2             -->  [1 71 2]
ts3             -->  [2 42 3]
ts4             -->  [3 32 4]
ts5             -->  [3 52 0]
ts6             -->  [4 52 1]

-->INCEDENCE MATRIX (WS) of Supervisor:modsup2.ads

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

-->CONTROL DATA (cdat2.pdt) in MATRIX FORM

 1
 0
 0
 1
 1

-->CONTROL POLICY

 1  0 -1  0  0  1


-->INITIAL MARKING of CONTROL PLACE

Control Place     Number of tokens

     c71               0

-->CONTROL POLICY in INFORMAL FORM

Control Place     :   c71
Pre Transitions   :   ts1 ts6
Post Transitions  :   ts3
Initial Marking   :   0

--------------------------------------------------------------------------
```

EK-I

Halka Toplama Kolu Kapasitesinin 2 Olduğu Durumda Hesaplanan Modüler Gözeticilerin Kontrol Poliçelerine Ait Program Çıktısı

```
---------------------------------------------------------------------------
Computation for Supervisor:modsup1.ads  and Control data:cdat1.pdt
---------------------------------------------------------------------------

-->Transition Coding for Supervisor:modsup1.ads

Transition name       TCT Coding
---------------      ----------
ts1             -->  [0 22 1]
ts2             -->  [1 12 0]
ts3             -->  [1 61 2]
ts4             -->  [2 32 3]
ts5             -->  [3 22 4]
ts6             -->  [3 82 0]
ts7             -->  [4 12 3]
ts8             -->  [4 82 1]

-->INCEDENCE MATRIX (WS) of Supervisor:modsup1.ads

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

-->CONTROL DATA (cdat1.pdt) in MATRIX FORM

 1
 0
 0
 1
 1

-->CONTROL POLICY

 1 -1  0 -1  0  0  0  1


-->INITIAL MARKING of CONTROL PLACE

Control Place     Number of tokens

     c61               0

-->CONTROL POLICY in INFORMAL FORM

Control Place     :   c61
Pre Transitions   :   ts1 ts8
Post Transitions  :   ts2 ts4
Initial Marking   :   0

---------------------------------------------------------------------------
```

```
--------------------------------------------------------------------------
Computation for Supervisor:modsup2.ads  and Control data:cdat2.pdt
--------------------------------------------------------------------------

-->Transition Coding for Supervisor:modsup2.ads

Transition name       TCT Coding
---------------      ----------
ts1             -->  [0 82 1]
ts2             -->  [1 71 2]
ts3             -->  [2 42 3]
ts4             -->  [3 52 0]
ts5             -->  [3 82 4]
ts6             -->  [4 52 1]

-->INCEDENCE MATRIX (WS) of Supervisor:modsup2.ads

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

-->CONTROL DATA (cdat2.pdt) in MATRIX FORM

 1
 0
 0
 1
 1

-->CONTROL POLICY

 1  0 -1  0  0  1


-->INITIAL MARKING of CONTROL PLACE

Control Place     Number of tokens

     c71               0

-->CONTROL POLICY in INFORMAL FORM

Control Place     :   c71
Pre Transitions   :   ts1 ts6
Post Transitions  :   ts3
Initial Marking   :   0

--------------------------------------------------------------------------
```

EK-J

Halka Toplama Kolu Kapasitesinin 5 Olduğu Durumda Hesaplanan Modüler Gözeticilerin Kontrol Poliçelerine Ait Program Çıktısı

```
---------------------------------------------------------------------------
Computation for Supervisor:modsup1.ads  and Control data:cdat1.pdt
---------------------------------------------------------------------------

-->Transition Coding for Supervisor:modsup1.ads

Transition name       TCT Coding
---------------      ----------
ts1             -->  [0 22 1]
ts2             -->  [1 12 0]
ts3             -->  [1 61 2]
ts4             -->  [2 32 3]
ts5             -->  [3 22 4]
ts6             -->  [3 82 0]
ts7             -->  [4 12 3]
ts8             -->  [4 82 1]

-->INCEDENCE MATRIX (WS) of Supervisor:modsup1.ads

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

-->CONTROL DATA (cdat1.pdt) in MATRIX FORM

 1
 0
 0
 1
 1

-->CONTROL POLICY

 1 -1  0 -1  0  0  0  1


-->INITIAL MARKING of CONTROL PLACE

Control Place     Number of tokens

     c61               0

-->CONTROL POLICY in INFORMAL FORM

Control Place     :   c61
Pre Transitions   :   ts1 ts8
Post Transitions  :   ts2 ts4
Initial Marking   :   0

---------------------------------------------------------------------------
```

```
--------------------------------------------------------------------------------
Computation for Supervisor:modsup2.ads  and Control data:cdat2.pdt
--------------------------------------------------------------------------------

-->Transition Coding for Supervisor:modsup2.ads

Transition name       TCT Coding
---------------      ----------
ts1             -->  [0 112 1]
ts2             -->  [1 71 2]
ts3             -->  [2 42 3]
ts4             -->  [3 52 0]
ts5             -->  [3 112 4]
ts6             -->  [4 52 1]

-->INCEDENCE MATRIX (WS) of Supervisor:modsup2.ads

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

-->CONTROL DATA (cdat2.pdt) in MATRIX FORM

 1
 0
 0
 1
 1

-->CONTROL POLICY

 1  0 -1  0  0  1


-->INITIAL MARKING of CONTROL PLACE

Control Place     Number of tokens

     c71               0

-->CONTROL POLICY in INFORMAL FORM

Control Place     :   c71
Pre Transitions   :   ts1 ts6
Post Transitions  :   ts3
Initial Marking   :   0

--------------------------------------------------------------------------------
```

ÖZGEÇMİŞ

Gökhan GELEN 04.02.1981 yılında Erzincan'da doğdu. İlk, Orta ve Lise öğrenimini Erzincan'da tamamladı. 1999 yılında girdiği İnönü Üniversitesi Elektrik-Elektronik Mühendisliği Bölümü'nden 2003'te mezun oldu. Yüksek Lisans eğitimine aynı yıl İnönü Üniversitesinde başladı. 2004 yılında Niğde Üniversitesi Elektronik Anabilim Dalı'na Araştırma Görevlisi olarak atandı. Yüksek Lisans eğitimini 2006 yılında Niğde Üniversitesi'nde tamamlayarak aynı yıl Doktora eğitimine başladı. İkinci lisans olarak Anadolu Üniversitesi İşletme Fakültesini 2008 yılında bitirdi. Halen Niğde Üniversitesi Elektrik-Elektronik Mühendisliği Bölümünde araştırma görevlisi olarak çalışmaktadır. Bilim dalındaki ilgi alanları Ayrık Olay Sistemleri ve Petri ağlarıdır.