ZAMANSAL MUHAKEME İLE DURUM TABANLI DAVRANIŞ ÜRETİMİ

M. Fatih Hocaoğlu

İstanbul Medeniyet Üniversitesi, Mühendislik ve Mimarlık Fakültesi, İstanbul
mfatih.hocaoglu@medeniyet.edu.tr

Agena Bilişim ve Savunma Teknolojileri, hocaoglu@agenabst.com

ÖZ

Bu bildiride birden fazla programlama paradigmasını durum tabanlı programlama paradigmasında tümleştiren Etmen tabanlı Simülasyon Sistemi EtSiS'in, zamansal muhakeme (temporal reasoning) davranış önermeleriyle model durum geçişlerini kullanarak davranış üretimi ele alınmıştır. Yapılan davranış üretimi, aynı simülasyon çevresini paylaşan simülasyon modellerinin birbirlerinin davranışlarını bir ikili (dual) model olarak oluşturmasını sağlar. Oluşturulan ikili model tanımı, modellerin öznitelik ve durumsallıkları ile oluşturulan dünyanın ötesine geçen, davranış semantiğini çözümleyen bir yapı oluşturur.

Zamansal muhakeme davranış şablonları, durum ve olay geçiş değişkenlerini ve aralarındaki mantıksal operatörleri içerir. Mantıksal çözümlemelerle elde edilen gramer kuralları, her bir davranış durum geçişlerini tanımlar ve davranış üretimi için temel bileşeni oluşturur. Gramer kurallarına uygun olarak oluşturulan davranışlar, bir etmenin diğer etmen davranışlarını ikili dünya olarak tanımladığı bir iç model tasviri olarak, model iç tanımında yeralır ve ilgili etmenin davranışlarının önceden kestiriminde önemli bir avantaj sağlar. Sağlanan yetenek, daha zeki simülasyon ortamları hazırlamada bir adım olarak görülmektedir.

Anahtar Kelimeler: Davranış üretimi, Durum tabanlı programlama, Etmen öğrenmesi, EtSiS, Zamansal muhakeme, Üretim kuralları

STATE BASED BEHAVIOR GENERATION BY TEMPORAL REASONING

ABSTRACT

In this study, how AdSiF, which combines multi-paradigm into state oriented paradigm, generates behaviors using state transitions and temporal reasoning behavioral premises is examined. Behavior generation provides simulation models, which share the same simulation environment, to create an inner dual world representation of each other. The

dual world representation constitutes a behavioral structure that is further from the world envisioned by model attributes and stative information.

Temporal reasoning behavioral patterns consist of state transitions, events, and logical operators between them. Grammar rules generated represent state transitions and define building blocks for behavior generation. The behaviors generated satisfying grammar rules are taken place in an agent inner structure as another agent's behavior representation and gives an advantage to forecast the behaviors of related agent. The opportunity is seen a step toward smarter simulation environments.

Keywords: AdSiF, Agent learning, Behavior Generation, Production Rules, State oriented programming, Temporal Reasoning

1. GİRİŞ

Durum tabanlı davranış üretiminde amaç, aynı simülasyon çevresini paylaşan etmen karakteristikli simülasyon modellerinin birbirlerine ait zamansal (temporal) muhakeme tabanlı davranış şablonları ile uyumlu davranış tanımlarını üretmektir. Davranış üretiminin girdileri, davranış üretimi yapılacak olan simülasyon modeline veya modellerine ait zaman etiketli durum geçişleri ve zamansal muhakeme tabanlı davranış şablonlarıdır. Davranış şablonları, değişken olarak tanımlanmış durum tanımlarını mantıksal operatörler ile ilişkilendiren ve kalitatif bir zaman sonra yürütecekleri mantıksal operatörlerle ilişkilendirilmiş durum geçişleri olarak tanımlayan zamansal formülasyonlar olarak tanımlanabilir. Bir şablon olması yönüyle, her bir davranış formülasyonu çözümlendiğinde birden fazla davranış üretimi gerçekleştirecektir. Bu bakışla, bir şablon tek bir davranış değil bir davranış ailesini temsil eder ve yalnızca bir modele ait durum tanımlamalarını değil, birden fazla modelin durum davranışları üzerine kurulu bir sonraki adım tahminini üretir. Bu yönüyle, bir grup modele ait kollektif davranışı üretir.

Çalışmada, tanımlanan davranış şablonlarının kullanımı ile bir modelin bir diğer modele ait davranış üretimi ve davranış diyagramları ile modele ait ikili (dual) davranış modeli kurması incelenmiştir. Oluşturulan ikili model, takip edilen modelin davranışlarının çözümlenmesiyle yapısal bir öğrenme sağlamaktadır.

Bölüm 2'de dil teorisi ve otomata teori ele alınmıştır. Bölüm 3'de çözümün ana çerçevesini oluşturan mantık programlama ve zamansal muhakemeden kısaca bahsedilmiş ve Bölüm 4'de oluşturulan çözümün model tasarımına ışık tutulmuştur. Bölüm 5'de bir hava savunma simülasyonu örneği ile doğrulama ve davranış üretimi gerçekleştirilmiştir. Sonuç bölümünde yöntemin sağladığı avantajlar ele alınmıştır.

2. DİL TEORİSİ VE OTOMATLAR

Dil ve otomata teorileri kesikli olay sistemlerinde formal bir işletim tanımlaması sağlar. Çıkış noktası, kesikli olay sistemlerinin kendileri ile ilişkili bir olay kümesi içeriyor

olması olarak ele alınır. E ile gösterilen olaylar kümesi dilin “alfabesi”ni ve olaylar ile üretilen karakter katarları ise dilin “kelimeleri”ni oluşturur. Bu kapsamda bir L dili, E olaylar seti ile formüle edilmiş karakter katarlarını içerir ve oluşturulan dil kesikli bir sistemin davranışlarını tanımlar. Bir dilin ürettiği kelimelerin geçerliliği gramer kuralları ile belirlenir. Gramer kuralları kelimenin üretilmesi ve harf tekrarlılıklarının geçerliliğini belirler ve her kelime bir nihai (final) durum ile sonuçlanır. Nihai durum sistemin, olay katarlarının (kelime) gerçeklediği bir seri durum geçişi tamamlayarak, bir eylemi yerine getirmesi ve nihai bir durum olarak ulaştığı bir kararlılık noktasıdır.

Gramer kuralları daha üst bir mantık operasyonu ile elde edilen çözümlemeler ile oluşturulur ve bu çalışmada sözkonusu mantık operasyonları zamansal muhakeme tabanlı olarak belirlenmiştir. Zira, sistem davranışları zamansal tanımlamalar içeren bir doğal dil ifadesi olarak belirtilmiştir.

Otomatlar belirli bir gramer kural setini sağlayan kelimeler üreten yapılar olarak tanımlanırlar. Sonlu durum otomatları sınırlı bir olay kümesine, bir başlangıç durumuna sahip bir durum kümesine, gramer kurallarına uygun durum geçişi sağlayan bir fonksiyona ve fonksiyonun ulaştıracağı nihai duruma sahiptir. Zamansal muhakeme tabanlı olarak belirlenen davranış önerme yapıları ile tanımlanan gramer kurallarının her biri bir sonlu durum otomatı karşılığına sahiptirler ve sistem davranışını gösterirler. Bu çerçevede, dil teorisi zamansal muhakeme tarafından tanımlanan model davranış tanımlamalarının tanımladığı gramer ve gramer kuralları ile üretilen her bir bileşenin simülasyon davranış doğrulaması [1], [2] ve davranış öğrenimi ile doğrudan ilgisi vardır.

3. MANTIK PROGRAMLAMA VE ZAMANSAL MUHAKEME

Mantıkta ana eksen tanımlı bir durumu doğru kabul eden veya aksi ispatlanmadığı için doğru varsayan bir önerme kümesini alan ve doğruluğu kabul edilen tanımlı şartlar altında doğru olması beklenen önermelerin neler olduğu belirlenir. Doğruluğu çıkarım yoluyla belirlenen önermeler, öncelikli olarak doğru kabul edilen önermelerin ima ettikleri ve bir anlamda gizli olarak doğruluğunu ileri sürdükleri duruma ilişkin önermelerdir ve mantığın amacı gizli olan doğrulukları açık ifadeler haline dönüştürmektir ve sonuç çıkarmaya dönük süreç “muhakeme” olarak tanımlanır [3].

Bir programlama paradigması olarak mantık programlama dünya tasvirinde, varlıklar arası ilişkiler, teorem ispat mekanizması ve doğruluğu kabul edilmiş önermeler seti ana unsurlar olarak görülebilir. Genel olarak mantık programlama bir problemin çözüm adımlarının tanımlanması yerine, problem tanımının tanımlanması ve teorem ispat yöntemleri ile çözüm aranması olarak anlaşılır. Yaygın olarak kullanılan mantık programlama yaklaşımı Horn-clause önerme yapılarının kullanımı ve problemin daha basit bileşenlere ayrılmasıdır. Problemin basit bileşenlere ayrılmasının kökü tümevarım muhakeme tekniğine dayanır [4]. Paradigma üç temel unsur üzerine kurulmuştur;

1. Değerler otomatik olarak üretilen değişkenlere atanır ve bu en genel tümleştirme olarak isimlendirilir (most general unifier),
2. Otomatik geri izleme (backtracking) mekanizması ile kontrol sağlanır,
3. Hesaplama tanımlı alfabe üzerinde gerçekleştirilir,

Önerme mantığı nesnelerin ve aralarında kurulan ilişkilerin varlık betimlemesi temellidir ve bu simülasyon modellemesine de yansıyan bir yaklaşımdır.

Zamansal muhakeme önerme mantığını bazı zamansal operatörler ekleyerek genişletir ve bu anlamda zamansal muhakeme bir tür biçimsel mantığın (modallogic) bir türüdür. Daha dar bir tanım ile zamansal muhakeme "daima (always)", "nihayetinde (eventually)" gibi ifadelerin doğruluk derecesini belirtmekte kullanılması ile kurulur [5]. Zamansal muhakeme ile bileşenlerin, protokollerin, nesnelerin, modüllerin, prosedürlerin ve fonksiyonların zaman ekseni üzerinde davranışlarının incelenmesi gerçeklenebilir. İnceleme önermelerin geçmiş, mevcut zaman ve geleceğe ilişkin davranış olarak üç fazı ile ilgilidir. Zamansal muhakeme program işletimi ile ilgili operatörler de sağlar. Bir işletim, bir seri durum geçişi olarak temsil edilir ve başlangıç durumu ile hesaplamaya başlanır. Geleceğe ilişkin hesaplamalar durum geçişleri ile belirlenir ve zamansal muhakeme önermeleri bu geçişlere ilişkin opratörler sunar [6]. EtSiS'in [7], [8] temel hesaplama paradigması olan durum tabanlı programlama paradigması, bu kapsamada, zamansal muhakeme ile güçlü bir uyum sağlar. Muhakeme p önermesi sağlayan s durumlar sıralaması $s \models p$ olarak gösterilir ve çalışmada kullanılan zamansal muhakeme gösterimi olarak tercih edilmiştir [9].

4. DAVRANIŞ ÖĞRENME VE MODEL TASARIMI

Davranış öğrenmede ana fikir, tanımlı zamansal muhakeme önerme şablonlarını sağlayan durum geçişlerinin bir davranış olarak belirlenmesidir. Geliştirilen zamansal muhakeme önerme şablonları Tablo 1'de görülmektedir. Önermelerin, A parçası mevcut durumu ifade ederken B parçası TFRP #1'de belirli zaman sonra ulaşılacak, TFRP #2 dışında olunacak ve TFRP #3'de ise ulaşılacak ve daimi olarak kalınacak olan tasvirleri gösterir. A ve B önerme bileşenleri durum tanımlarının mantıksal operatörler ile ilişkilendirilmesiyle oluşturulan mantıksal önerme bileşenleridir.Şablonlarda A ve B durum tanımlamaları birden fazla durum ifadesini içeren ve mantıksal operatörler ile ilişkilendirilmiş durum tanımları olabilir. Benzer şekilde, durum tanımları birden fazla modele ait tanımlamaların ilişkilendirilmesi ile gerçekleştirilebilir. Örnek uygulamada ele alınan ve savunma birimi varlıkları ile hücum varlıklarının her ikisini de içeren davranış yapıları buna örnek olarak görülebilir.

Tabloda yeralan zamansal önerme şablonlarının mantıksal operatörler ile çözümlenmesi sonucunda elde edilen gramer kuralları her önerme şablonunun alt kısmında, sonlu durum otomat gösterimleri ile yeralmaktadır. Sonlu durum otomatlarında içleri siyah

olarak doldurulmuş olan durumlar nihai durumu göstermektedir. Gramer kurullarında aralarında "ve" operatörü bulunan kurullar ardışık olay durum geçişlerini, "veya" operatörü olanlar ise alternatif (farklı şartlarda dallanılan) durum geçişlerini göstermektedir.

Bir etmenin diğer etmenin davranış modelini çözümlemesi (öğrenmesi) paylaştıkları çevrede gelişecek olaylara göstereceği reaksiyonu önceden üretmesi adına oldukça önemli bir avantaj sağlayacaktır. Hedef merkezli (Goal-Oriented) etmen tasarımlarında, başarılacak bir amacın, ulaşılacak bir amacın olması durumunda daha fazla önem kazanan bir nitelik olarak karşımıza çıkmaktadır [10].

Tabloda Bölüm 2'de ele alınan dil teorisi gramer çözümlemeleri ve bunların otomat karşılıkları görülmektedir. Tüm zamansal davranış şablonlarında S0 başlangıç durumunu göstermektedir. Her bir gramer kuralının gösterdiği durum geçişleri sağlandıktan sonra bir nihai durumda sistem kararlı olarak sonlanır. Örnek olarak, TFRP #1'de Kural 2 ile belirtilen gramer kuralında, S0 başlangıç durumundan, ¬A durum geçişi ise S0'da sonlanma veya A ve bir sonraki zaman noktasında B durum geçişi yürütülerek (A◊B) yine S0 durumunda sonlanır. Gramer kurallarının tümünde sonlanmanın S0'da olduğu görülür ve S0 nihai durum olarak gözlemlenir. TFRP #2 ve #3'de S0 ve S1'in nihai durum olduğu görülmektedir. Her iki zamansal davranış şablonunda da, S0 durumundan S1'e geçiş gerçekleştirildikten sonra tekrar S0'a geçişin mümkün olmadığı görülmektedir. TFRP #3'ün doğal dil ifadesi "S0 durumunda bulunan bir sistem, A geçişi yaptıktan sonra ulaştığı S1 durumunda sonsuz zaman için kalır". Bunun doğal bir sonucu olarak zaman ekseni üzerinde TFRP #3 ile tanımlı bir davranış, A geçişi yaptıktan bir zaman sonra tekrar S0'e geçişi bir davranış tasarım hatasını gösterir.

Model A durumuna girdikten bir zaman sonra B durumuna geçiş yapar.

Kural 1: Bir başlangıç durumu tanımlanır.

Kural 2 ve 3: Model A durum geçişini sağlayan olaydan farklı bir olay geçişi yürütür veya A durum geçişinden sonraki bir zamanda B durumuna geçiş yapar. A durum geçişi bir kararlılık noktası değildir ve takip eden zamanda B durum geçişi mutlaka yaşanarak sistem kararlılığa ulaşır.

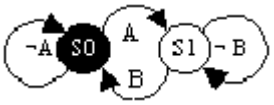
Kural 4: A geçişi ile S1 durumuna ulaşan bir sistem, B geçişi ile kararlılık noktası olan S0'a ulaşır veya B den farklı olan bir geçişi takip eden bir B geçişi ile S0'a ulaşır.

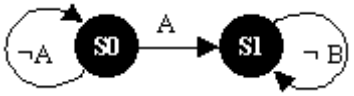
TFRP #2: φ:= □ (A→¬◊B) TFRP #1'in olumsuz (not) şartlı geçişini göstermekte olup, S0 ve S1 gibi iki karalılık durumuna sahip olduğu görülür.

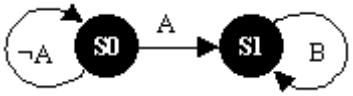
TFRP #3 bir süreklilik göstermektedir. Sistemin daima A ile tanımlı geçişi yürüterek S1 durumunda karalı kaldığı görülmektedir.

Tablo 1. Davranış Önerme Şablonları

TFRP #1: φ:= □ (A→◊B)	TFRP #2: φ:= □ (A→¬◊B)	TFRP #3: φ:= □ (A→□B)
Kural1: A→◊B, S ←φ	Kural 1: A→¬◊B, S ←φ	Kural 1: A→□B, S ←φ
Kural 2: A→◊B = ¬A+A(◊B)	Kural 2: A→¬◊B =¬A+A(¬◊B)	Kural 2: A→□B = ¬A+A(□B)
Kural 3: P:={S→¬A, S→A(◊B)}	Kural 3: P:={S→¬A, S→A(¬◊B)}	Kural 3: P:={S→¬A, S→A(□B)}
Kural 4: ◇B →B + ¬B(◇B)	Kural 4: ¬◇B →¬B(¬◇B)	Kural 4: □B →B(□B)
Kural 5: P:= P∪{◇B→B, ◇B→¬B(◇B)}	Kural 5: P:= P∪{◇B→¬B(¬◇B)}	Kural 5: P:= P∪{□B→B(□B)}







TFRP #1: φ:= □ (A→◊B) açıklaması aşağıdaki gibi ele alınır.

Tablonun hemen alt kısmında görülen durum diyagramları gramer kurallarının ürettiği sistem davranışını göstermektedir.

Gramer kuralları S0 ve S1 olarak iki durum tanımı ile bir şablon olarak gösterilmektedir. Bu en genel tümleştirme (most general unifier) prensibi ile uyumludur ve her iki durum tanımı değer olarak sisteme ait bir set durum değişkenini ve ilgili değerleri içerir. Tüm mümkün dünyanın iki durum olarak betimlenmesi, gramer kurallarının oluşturulmasında yürütülen mantık operatörlerinin "*not*" operatörü içermesidir. Operatör betimlenen durum uzayını iki farklı parçaya ayırır.

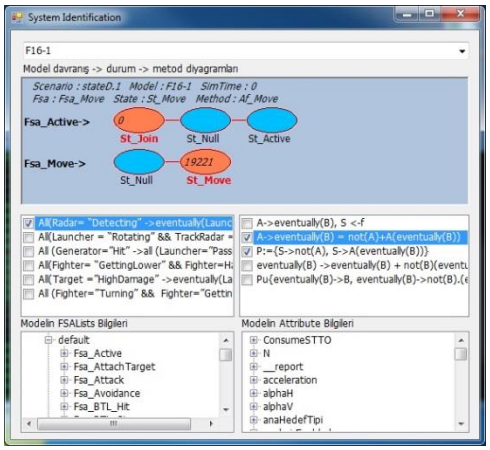
Davranış üretimi simülasyon işletim zamanında gerçekleştirilir ve aşağıdaki süreç adımları uygulanır;

1. İşletim simülasyon işletimi boyunca durum ve olay geçişlerinin takibi,
2. Elde edilen durum geçişlerinden şablon zamansal davranış gramer kuralları ile uyumlu durum geçişleri yakalama,
3. Sağlanan gramer kurallarından zamansal davranış formülü oluşturma,
4. Çift anlamlılıkları düzenleme; Bir durum dışında olma ifadesi (¬B) hesaplamalarda dışında olunacak durum dışındaki durumların oluşması ile gerçeklenir. Örnek olarak ¬(Launcher = "Working") tanımı Launcher'ınWorking durumu dışındaki durumların gerçeklenmesi olarak gözlemlenir. Working durumunun oluşmuyor olması çift anlamlı durumun bir kural olarak tespiti ile gerçekleştirilir.

Yazılım arayüzü Şekil 1'de görülmektedir. Davranış üretim algoritması aşağıda özet olarak belirtilen adımları içerir;

- Tüm durum geçişlerini takip et (Şekil 1); Modelin işletim esnasında yürüttüğü davranışlar (durum diyagram geçişleri) izlenir ve geçişlerin Tablo 1'de sunulan zamansal muhakeme davranış gösterim şablonlarına ait gramer kurallarını sağlayıp sağlamadığı izlenir.

- Her bir gramer kuralı için bir arama uzayı oluştur; Gramer kuralları için oluşturulan arama uzayı ile davranışın gramere uygunluğu değil, grameri sağlayan davranış şablonları oluşturulur. Bu modelin davranışlarını izleyen bir zeki etmenin, model davranışlarını daha üst bir gramerde tanımlayarak modelin davranış gramerini çözümlemesini, bir diğer ifade ile, davranış yapısını zamansal muhakeme tanımlamaları ile öğrenmesini sağlar. Buradaki öğrenme yapısal bir öğrenmedir. Alt adımları aşağıdaki gibidir;
    - Gramer kurallarının oluşturduğu zamansal formülü ile gramer kurallarını ilişkilendir,
    - Gramer kurallarını sağlayan durum geçişlerini belirle; Her bir gramer kuralını oluşturan durum geçişleri belirlenir.
    - TF şablonunu sağlayan gramer kurallarını belirle; sağlanan gramer kurallarından zamansal muhakeme ifadesinin birleştirilerek oluşturulması amaçlanır (şekilde sağlanan gramer kuralları görülmektedir).



Şekil 1. Davranış Üretim Modeli Arayüzü

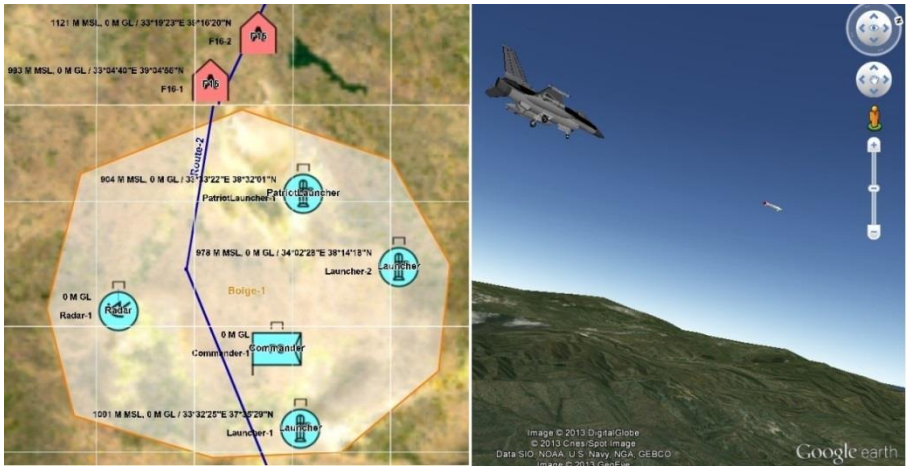
Algoritmanın temelinde arama algoritması ve geri izleme (backtracking) yaklaşımı esastır. Yaklaşım bir gramer kuralının sağlanıp sağlanmadığı araştırmak yerine, gramer kuralını sağlayan durum geçişlerini belirlemeyi sağlar.

5. ÖRNEK UYGULAMA: HAVA SAVUNMA SİMÜLASYONU

Seçilen örnek uygulamada havadan yere hücum gerçekleştiren uçaklara karşı savunma gerçekleştiren yer konuşlu füze savunma sistemleri mevcuttur. Yer savunma sistemleri hedef tespitlerini gerçekleştiren radar, tespit iletimlerini gerçekleştiren haberleşme cihazları ve komuta modeli ile silah sistemine ait lançer ve füzelerden oluşmaktadır.

Hücum unsuru olan uçaklar, yer savunma sistemi zamansal muhakeme tabanlı davranış formülasyonlarını gerçekleştirirler. Hücum ve savunma unsurları tarafından üretilen davranış tanımlamaları ayrı ayrı ele alınmıştır. Gözlemlenen durum geçişleri

ModelIsmi(Durum, DurumGirişZamanı) şeklinde bir format kullanılmıştır. Örnek birkaç gösterim Tablo 2'de görülmektedir.



Şekil 2. Yer Konuşlu Hava Savunma Sistemi ve Angajman Görünüşü

Tablo 2. Sistemlere Ait Durum Geçişleri

Hücum unsuru tarafından gözlemlenen savunma sistemi durum geçişleri

Radar(Detecting, 1.5)	TrackRadar(Tracking, 2.0)	Radar(Detecting, 2.8)
Radar(Detecting, 1.8)	Launcher(Rotating, 2.2)	Launcher(Fire,3.3)
TrackRadar(Tracking, 1.9)	Radar(Detecting, 2.3)	Generator(Hit,4.5)
Radar(Detecting, 1.9)	TrackRadar(Tracking, 2.5)	Launcher(Passive,4.5)
Launcher(Rotating, 2.0)	Radar(Detecting, 2.6)	Generator(Hit,5.0)
Radar(Detecting, 2.1)	Launcher(Rotating, 2.8)	Launcher(Passive,5.6)

Savunma unsuru tarafından gözlemlenen hücum varlığı durum geçişleri

Fighter(GettingLower, 1.0)	Fighter(AltStabilezed, 1.6)	Fighter(Approaching,3.0)
Fighter(GettingLower, 1.2)	Fighter(AltStabilezed, 1.7)	Fighter(Approaching,3.3)
Fighter(GettingLower, 1.3)	Fighter(Heading, 1.7)	Fighter(Approaching,3.5)
Fighter(GettingLower, 1.3)	Fighter(AltStabilezed, 2.1)	Fighter(ReleaseBomb,3.8)
Fighter(Heading, 1.3)	Fighter(Heading, 2.4)	Fighter(Turning,3.9)

Hücum unsurları yer savunma sistemi model bileşenlerine ait durum geçişlerini, savunma unsurları ise hücum eden savaş uçağı modeli durum geçişlerini takip ederek davranış üretimini gerçekleştirirler. Durum geçişlerinden gramer kurallarını sağlayan durum koleksiyonları kullanılarak üretilen zamansal davranışlar aşağıda sunulmuştur. Davranışların doğal dil açıklaması aşağıdaki şekildedir.

Savunma unsuru bir tespit yaptıktan sonra, lançerin tespit koordinatına doğru dönmeye başladığı gözlemlenir. Lançerin dönüşüyle beraber silah sistemine ait izleme radarının hedefi takibe başladığı görülür ve bu davranışı lançerin ateş etmesi takip eder. Hücum unsurunun jeneratörü hedef alması ve vurması durumunda silah sistemi işletilemez duruma geçer.

Hücum unsurunun alçalarak savunma sistemine yaklaştığı ve bomba yüklü olduğu gözlemlenir. Yapılan tespiti takiben, hücum eden uçağın yükseleceği veya bombasını

bırakacağı alternatifi bir taktik uygulama olarak bilinir ve durum geçişlerinden bombayı bıraktığı gözlemlenir. Burada hücum gerçekleştiren uçaklardan bir diğerinin yükselerek farklı bir taktik uyguladığı koşum kayıtlarından gözlemlenir. Bu sebeple oluşturulan zamansal davranış formülü "veya" operatörü içerir. Hücum eden uçağın bombasını bıraktıktan sonra manevra yapar ve uzaklaşır. Bu noktadan sonra savunma unsuru uçağı bir tehdit olarak görmez.

Hücum unsuru TFRP #1 için üretilen zamansal davranış

□ (Radar= "Detecting" →◊(Launcher="Rotating"))

□ (Launcher = "Rotating" &&TrackRadar = "Tracking" →◊(Launcher="Fire"))

Açıklama: Radar bir hedef tespit ettikten bir süre sonra lançer hedefe doğru döner.

Lançerin hedefe dönmesi ve izleme radarının hedefi izlemeye başlamasını lançerin hedefe ateş etmesi takip eder.

TFRP #3 için üretilen zamansal davranış

□ (Generator="Hit" →□ (Launcher="Passive"))

Açıklama: Jeneratör vurulursa bir süre sonra lançer işlevini yitirir.

Savunma unsuru TFRP #1

□ (Fighter= "GettingLower" && Fighter=HasBomb→◊(Fighter=GetHigh || Fighter=PopUpTheBomb))

□ (Target="HighDamage" →◊(Launher="StopEngagement"))

Açıklama: Savaş uçağı alçalıyor ve bombaya sahipse bir süre sonra yükselecek veya bomba bırakacaktır. Eğer hedef yüksek seviyede hasarlı ise lançer angajmanı sonlandıracaktır.

TFRP #3 için üretilen zamansal davranış

□ (Fighter="Turning" && Fighter="GettingFarther"→□ (Fighter="NotThreat"))

Açıklama: Savaş uçağı manevra yapıyorsa ve uzaklaşıyorsa artık bir hedef olarak değerlendirilmeyecektir.

Zamansal muhakeme davranış tanımlarından görülebileceği gibi, durum geçişlerinin gramer kuralları ile eşleştirilerek oluşturulan davranışlar ile modelin bir sonraki zaman adımında yürüteceği davranışı yapacağı seçimi önceden kestirmek mümkün olacaktır. Bu bir etmenin diğer bir etmen davranışını öğrenmesi olarak yorumlanabilir.

6. SONUÇ

Bildiride sunulan çözüm yaklaşımı, aynı simülasyon sahnesini paylaşan aktörlerin, etmen benzeri davranışları ile birbirlerine ait davranış iç yapılarını çözümlemeye ve bir

anlamda davranışlarını öğrenmeye yöneliktir. Aktörlerin davranış öğrenme tanımı, sahip oldukları zamansal formül şablonlarına değer atanmış, durum geçişleri ile uygulanır bir forma ulaştırılmış tanımlarına ulaşmasıdır. Yöntemin sağladığı önemli avantaj, bir meta- bilgi olan zamansal formül şablonlarına sahip etmenlerin reaktif bir öğrenme sağlaması, diğer aktörlerin olaylara karşı göstereceği tepkileri önceden iç model tanımı ile kestirebilmesi ve bir doğrulama amacı olarak da, modelin gerçekte ürettiği davranışın neler olduğunu görebilmektir. Doğrulama amacında yaklaşım, modelin yürütmesi gerektiği davranışları tanımlayarak, gerçekleşip gerçekleşmediğini kontrol etmek yerine, modelin yürütmekte olduğu davranışları, tanımlı gramer kurallarına uygun bir şekilde üretmesinin sağlanmasıdır.

Yaklaşımda önemli zorluk, gramer kuralını doğrulayan çift anlamlı (ambiguous) durum geçişleridir. Durum geçişlerinin filtrelenmesi ve uygun durum geçişlerinin birleştirilmeleri için kural tabanı geliştirilmiştir.

7. KAYNAKÇA

[1] Liu, C., Orgun, M. A., "Verification of Reactive Systems using Temporal Logic with Clocks", Theoretical Computer Science 220 377-408, (1999).

[2] Zeigler, B. P., Sarjoughian, H. S., (2002), "Implications of M&S Foundations for the V&V of Large Scale Complex Simulation Models", An Invited Paper for Session A6 of the Foundations for V&V in the 21st Century Workshop (Foundations '02) held at the Johns Hopkins University Applied Physics Laboratory, Laurel, Maryland (USA), October 22-24.

[3] Shapiro, S. C., (1992), "Encyclopedia of Artificial Intelligence", (v.2), John Wiley&Sons, Inc., New York.

[4] Russell, S. J.,Norvig, P., (1995), "Artificia Intelligence: A Modern Approach", Prentice-Hall International Inc., New Jersey.

[5] Garson, J. W., (2000), "Modal Logic", Stanford Encyclopedia of Philosophy, http://plato.stanford.edu/contents.html#l, pp. 1.

[6] Hailpern, B. T., (1982), "Verifying Concurrent Processes Using Temporal Logic", Springer-Verlag, Berlin.

[7] Hocaoğlu, M. F., (2005), "AdSiF: Agent Driven Simulation Framework", The Huntsville Simulation Conference 2005, 26-27 October.

[8] Hocaoğlu, M. F., (2011), "EtSiS: Etmen tabanlı Simülasyon Sistemi", 4. Ulusal Savunma Uygulamaları Modelleme ve Simülasyon Konferansı, USMOS'2011, Ankara, Türkiye.

[9] Manna, Z.,Pnueli, A., (1995), "TemporalVerification of ReactiveSystems", Springer-Verlag, New York.

[10] Hocaoğlu, M. F., ve diğerleri, (2002), "DEVS/RAP Agent-BasedSimulation", AI, Simulation & Planning in High Autonomy Systems-AIS'2002, April 7-10, Lisbon, Portugal.