Robot Kontrolü için Mantıksal Akıl Yürütme

Ozan Çaldıran[*] *Kadir Haspalamutgil*[*] *Abdullah Ok*[*] *Can Palaz*[*]
Esra Erdem[†] *Volkan Patoğlu*[†]

Sabancı Üniversitesi
Mühendislik ve Doğa Bilimleri Fakültesi
Tuzla, 34956, İstanbul, Turkey

[*]{cozan, kadirhas, abdullah, canpalaz}@su.sabanciuniv.edu
[†]{esraerdem, vpatoglu}@sabanciuniv.edu

Özetçe

Bu makalede, birden fazla robotun kontrolü için, eylem tanımlama dili $\mathcal{C}+$'ı ve otomatik akıl yürütücü CCALC'ı kullanan, mantığa dayalı bir sistem sunulmaktadır. Yeni yöntemler ve algoritmalar içeren bu sistemin uygulanılırlığı ve etkinliği, LEGO NXT MINDSTORMS robotları üzerinde gösterilmektedir. Özellikle, eş zamanlı eylemleri içeren bazı planlama problemleri, robot kontrolünü gözönünde bulundurarak ve plan uzunluğunu optimize ederek, otomatik olarak CCALC ile çözülüp robotlar tarafından doğru ve verimli bir şekilde uygulanabilmektedir.

1. Giriş

Robot teknolojisinin fabrika ortamlarını aşıp gündelik hayat içerisinde daha fazla yer almaya başlamasıyla beraber, robotlardan talep edilen özellikler, hız ve hassasiyetten, güvenlik ve bilişsel yetilere doğru kaymıştır. Nitekim, insanların da yer aldığı dinamik ortamlarda çalışan robotlarda, daha ileri seviyede esnekliğe, hünere, gürbüzlüğe ve bilişsel kapasiteye ihtiyaç duyulmaktadır. Bu tür ihtiyaçlar, iki farklı robotik yaklaşımının yakınlaşmasını gerektirmektedir: Geleneksel robotikte amaç, kazancı yüksek konum kontrolü altında çalışan katı robotlar tasarlayıp inşa etmektir; bilişsel robotikte [1] amaç, (hedefler, algı, eylemler, diğer etmenlerle işbirliği içerisinde görev icrası gibi) yüksek seviyede bilişsel fonksiyonları yerine getirebilen robotlar tasarlamak ve böylece robotların dinamik bir dünyada yüksek seviye karar alabilmelerini sağlamaktır. Bu çalışmada, robot kontrolü için mantıksal akıl yürütme kullanılarak, geleneksel ve bilişsel robotik alanların arasında bir köprü kurulması hedeflenmiştir.

Literatürde, iki alanı yaklaştırmayı hedeflemiş çalışmalar mevcuttur. Bu çalışmalarda, çeşitli biçimsel diller kullanılarak, eylemler ve değişim üzerinde mantıksal akıl yürütme gerçekleştirilmiştir. Örneğin, [2]'de durum analizi (Situation Calculus) [3, 4] tabanlı yüksek seviyede robot kontrol dili GOLOG [5] kullanılarak, LEGO MINDSTORMS RIS robotları programlanmıştır. Benzer şekilde [6]'da GOLOG and RHINO kontrol yazılımları kullanılarak RWI B21 and B14 gezgin robotları üzerinde plan icra takibi gerçekleştirilmiştir. [7]'de GOLOG'un bir başka sürümüyle futbol oynayan robotların koordinasyonu sağlanmıştır. WITAS insansız hava aracı projesinde, bir helikopter kontrol sisteminin [8] parçaları olan eylemler ve olaylar, zamansal eylem mantığı [9], öznitelikler ve zamansal değişenler (fluents) [10] ve bilişsel robot mantığı [11] vasıtasıyla ifade edilmiştir. [12]'da ise olay analizi (Event Calculus) [13, 14] bir Khepera robotun mantıksal akıl yürütmesi için uyarlanmıştır. Zamansal değişen analizi (Fluent Calculus) [15] tabanlı etmen programlama dili FLUX [16] da bazı robotlarda hayata geçirilmiştir. Örneğin, [17]'de FLUX, Pioneer 2 gezgin robotunun plan icra takibi için kullanılmıştır.

Bu çalışmada daha önceki çalışmalarda da olduğu gibi yüksek seviye mantıksal akıl yürütme ile robot kontrolü bir araya getirilmiştir. Öte yandan, daha önceki uygulamalardan farklı olarak,

- değişik bir biçimsel dil (eylem tanımlama dili $\mathcal{C}+$ [18]),
- değişik bir mantıksal otomatik akıl yürütücü (CCALC[1]),
- ve değişik robotlar (LEGO MINDSTORMS NXT)

kullanılmaktadır. Ayrıca birden fazla etmenin, eş zamanlı eylemleri gerçekleştirmesini gerektiren bir eylem uzayı ele alınmıştır. Detaylandırmak gerekirse, bir yükün iki gezgin robot tarafından, daha önceden tanımlanmış düzlemsel bir labirent üzerinde belirtilen bir başlangıç konumundan, seçilen bir hedef konumuna engellere çarpmadan taşınmasının planlanması ve bu planın uygulanması üzerinde çalışılmıştır. Buradaki ana fikir, bir planın otomatik olarak ortaya çıkarılması ve birden fazla etmen tarafından eş zamanlı olarak icra edilmesidir (Şekil 1).

Bu çalışmanın katkıları iki ana başlık altında özetlenebilir.

- Eylem tanımlama dilleri [19], eylemlere ve değişikliklere dair mantıksal akıl yürütme problemlerinin çözümünde sıkça kullanılmıştır. Öte yandan, diğer formalizasyonların aksine, eylem tanımlama dillerinin geleneksel robotik uygulamalarına uygunluğu ve akıl yürütme için kullanılabilirliği fiziksel robotlar üzerinde henüz gösterilmemiştir. Bu bağlamda bu çalışma eylem tanımlama dillerinin bilişsel robotik alanındaki kullanımının ilk örneğini teşkil etmektedir.
- Ek olarak, robot kontrolü için mantıksal akıl yürütmenin, göreceli olarak ucuz ve kullanımı kolay olan LEGO MINDSTORMS NXT robotlarına uygulanmış oluşu,

[1] http://www.cs.utexas.edu/users/tag/cc

bu robot mimarisinin robotik konusunda kısıtlı bilgiye sahip olup bilişsel robotik üzerinde çalışan araştırmacılar tarafından kullanımının yolunu açmıştır. Önerilen sistem mimarisi, araştırmacıların kendi biçimsel dillerini deneyebilecekleri bir ortam olmakla beraber, özellikle, eğitim amaçlı çalışmaların fiziksel robotlar üzerinde denenebilmesinin önünü açmaktadır.

Bildirinin devamında, öncelikle Şekil 1'de gösterilen genel sistem mimarisi detaylandırılmıştır. Bildiride ele alınan eylem tanım kümesi ve planlama problemleri açıklanıp, problemlerin biçimlendirilmesine kısaca değinilmiştir. Sonrasında, akıl yürütücü CCALC tarafından hesaplanan planların LEGO MINDSTORMS NXT robotlarınca icra edilişi anlatılmıştır. Bildiri, yapılan çalışmaların sonuçlarının özetlenmesi ve gelecekte ele alınacak konuların tartışılması ile sonlandırılmıştır.

2. Sistemin Genel Mimarisi

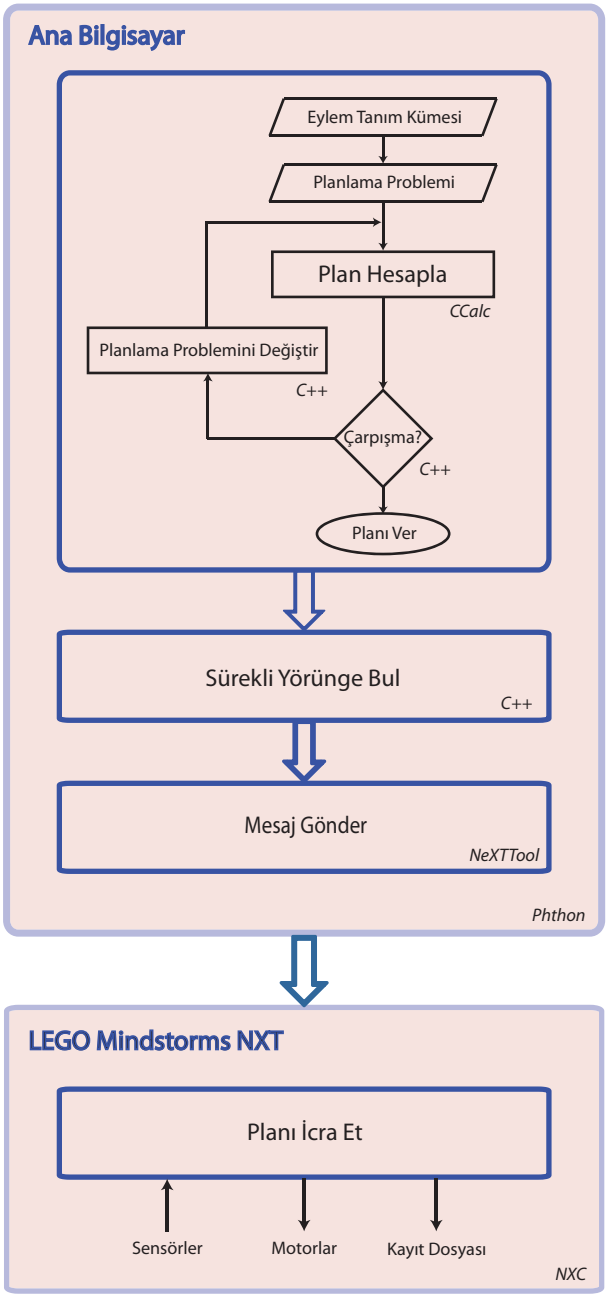
Robot kontrolünün akıl yürütme ile biraraya getirildiği sistemimizin genel mimarisi Şekil 1'de gösterilmiştir.

Öncelikle, eylemlerin tanım kümesi, bir eylem tanımlama dili olan $\mathcal{C}+$ [18] kullanılarak tanımlanır. Buradaki amaç, yapılmış olan tanıma uygun olarak, LEGO NXT MINDSTORMS robotlarının ortak bir hedefi gerçekleştirmek üzere icra etmesi gereken eylemleri planlamaktır. Bu tür bir planın hesaplanması için otomatik akıl yürütücü CCALC kullanılabilir. CCALC, verilmiş olan başlangıç durumuna ve hedef koşullarına bağlı olarak, hedeflenen bir duruma ulaşmak için bir plan ve bu planın içerdiği bütün eylemlerin ve durumların bir özetini (plan geçmişini) hesaplar. Bu eylem ve durum geçmişi, ve ters kinematik denklemler kullanılarak, robotların motor açılarının yörüngeleri bir C++ programı ile elde edilir; ve bu motor açıları NeXTTool adlı bir program ile Bluetooth iletişimi aracılığıyla robotlara gönderilir. Bu çalışmada bütün bu işlemler bir ana bilgisayar üzerinde Python diliyle yazılmış bir program tarafından otomatik olarak gerçekleştirilmektedir.

LEGO MINDSTORMS NXT'nin beyni, ARM7 mikroişlemcisi kullanan NXT adlı bir atanmış kontrolördür. NXT, dahili Bluetooth aracılığıyla, bilgisayarla iletişim kurabilme ve kendi için özel olarak tasarlanmış üç adet motoru sürebilme özelliklerine sahiptir. Bu çalışmada iki adet motor robotun doğrusal hareketi için, ve diğer motor ise robot kolunun hareketini sağlamak için kullanılmıştır. Yükün robotlar tarafından kavranması için fazladan bir serbestlik derecesine olan ihtiyacı ortadan kaldırmak amacıyla, sonlandırıcı olarak sürekli mıknatıslar ve yük olarak ise metal uçlu bir çubuk seçilmiştir. NXT programlamak için, çeşitli method ve programlama dilleri arasından, göreli kullanım kolaylığı ve dokümantasyonu nedeniyle NXC seçilmiştir. NXC diliyle yazılmış olan bir program, ana bilgisayardan gönderilen mesajların (yani motor referanslarının) okunması ve bu referanslar aracılığıyla robot motorlarının kontrolünün gerçekleştirilmesi için kullanılmıştır.

3. Örnek: İki Robotun Bir Yükü Birlikte Taşıması

İki robot ve bir yükün üzerinde bulunduğu bir platform düşünelim. Yükün bir demir çubuk olduğunu, ve yükü ucundan tutabilmesi için her bir robotun her iki sonlandırıcısında da mıknatıs bulunduğunu farzedelim. Ayrıca, hiçbir robotun yükü tek başına taşıyamadığını, ve yükün hareketi için iki robotun da yükü tutuyor olması gerektiğini farzedelim. Üzerinde çalıştığımız planlama problemi, yükün platform üzerinde belirtilen bir yerden başka bir yere taşınmasını amaçlamaktadır.



Şekil 1: Sistemin genel mimarisi

Bu problemi çözmek için öncelikle eylemlerin daha sonra da problemin tanımlanması gerekmektedir. Bu tanımlar için, platformun (koordinat düzleminde) bir labirent olduğunu, robotların sonlandırıcılarının ve yükün uç noktalarının labirent üzerinde noktalar ile gösterildiğini, ve engellerin ise çizgiler olarak tanımlandığını farzedelim.

Bu örnekte, robotların sonlandırıcıları dört yönde (yukarı,

aşağı, sağ ve sol) tam sayılık adımlarla hareket edebilirler. Bu tür hareketler, eylem tanımlama dili $\mathcal{C}+$'da başlıca dört kısımda, nedenselliğe dayanan kurallarla tanımlanır: eylemin icrası için gereken ön koşullar, eylemin icrası sonrasında doğrudan olan etkileri, herhangi bir durumun tanımı, ve durumlar ve eylemler ile ilgili kısıtlar. Örneğin, yükün bir yerden başka bir yere taşıma eylemini gözönünde bulunduralım. Bu eylemin bir ön koşulu, robotların yükü her iki ucundan tutmalarıdır. Bu tür ön koşullar, bu eylemin tanımının ilk kısmını oluşturmaktadır. Ön koşulları yerine getirildiği takdirde bu eylem icra edilebilir, ve bu eylem icra edildiği takdirde yükün pozisyonu değişir; eylemin icrasına doğrudan bağlı olan bu tür durum değişiklikleri ikinci kısımda tanımlanır. Herhangi bir durumda, robotun bir sonlandırıcısı yükün bir ucunun üstündeyse, mıknatıstan dolayı, robot yükü o ucundan tutuyor pozisyonundadır; eyleme doğrudan bağlı olmayan bu tür kurallar üçüncü kısımda tanımlanabilir. En son kısımda ise kısıtlar tanımlanır: örneğin, yük taşınmadığı sürece yer değiştirmez; yük platformdaki engellerle üstüste gelemez.

Robotların hareketleri boyunca engellere çarpamayacağı $\mathcal{C}+$ dilinde bir ön koşul olarak belirtilmiştir. Yükün herhangi bir durumda engel ile üstüste gelemeyeceği de bir kısıt olarak tanımlanmıştır. Burada tanımlanan çarpışma durumları ayırıcı eksen teorisi kullanılarak tespit edilebilmiştir. Bu çarpışma durumlarının haricinde, yükün hareketi sırasında engellere temas etme ihtimali de bulunmaktadır. Bu çarpışmaların tespit edilebilmesi için, bir durumdan diğerine geçilirken, yükün her iki ucunun da pozisyonları zamana bağlı birer fonksiyon olarak yazılarak, yükün her an için pozisyon ve oryantasyonunu belirten bir doğru denklemi çıkarılmıştır. Bu doğru denklemi kullanılarak engellerin uç noktalarının hareket sırasında yük ile çakışıp çakışmadığı tespit edilebilir.

Yukarıda açıklanan eylem tanım kümesinin $\mathcal{C}+$'daki tanımını [20]'da görebilirsiniz.

Her ne kadar $\mathcal{C}+$'da kısıtlar tanımlanarak hareket sırasında çarpışma oluşmasını garanti etmek teorik olarak mümkün olsa da ([20]'da açıklandığı gibi); bu kısıtların eklenmesi hem çözüm süresini hem de hafıza ihtiyacını gerçekleştirilebilir seviyenin dışına taşımaktadır. Bu nedenle, mantıksal akıl yürütücü CCALC'tan elde edilen planlarda bu çarpışma durumu kontrol edilmemektedir. Bu tür çarpışmaları engellemek için Algoritma 1'da gösterilen algoritma tanımlanmıştır. Bu algoritmaya göre, CCALC'ın bulduğu planda bu tip çarpışmaların gerçekleşip gerçekleşmeyeceği C++ dilinde yazılmış başka bir program ile kontrol edilir; çarpışma olacaksa, çözülen planlama problemi çarpışma yaratacak olan eylemlerin ve durumların yasaklayacak şekilde değiştirilir; ve tekrardan CCALC'a çözüme gönderilmektedir. Bu döngü C++ programının herhangi bir çarpışmaya yol açmayacak bir plan hesaplanana kadar devam eder. Bu algoritmanın nasıl işlediği, örneklerle [20]'da gösterilmiştir.

Algoritma 1 PLAN

Girdi: Eylem tanım kümesi $\mathcal{D}$, planlama problemi $\mathcal{P}$
Çıktı: Uzunluğu en fazla n olan ve çarpışma içermeyen bir plan P (varsa)

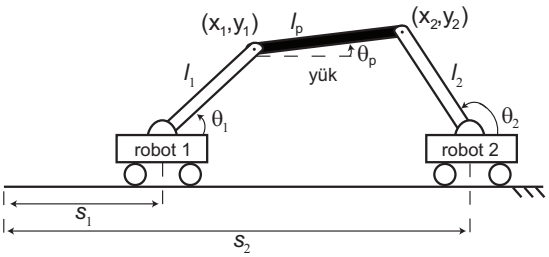
```
plan := false;    // çarpışma içermeyen bir plan bulunmadı

while ¬plan do
  plan, P, H ← CCALC'ı kullanarak, uzunluğu en fazla n olan bir plan P (varsa) ve bu planın geçmişini H hesapla;

  if plan then
    collision := false;    // çarpışma bulunmadı
    i := 0;

    while ¬collision AND i ≤ |P| do
      Δ ← i ve i + 1'inci adımlarda, robot sonlandırıcılarının konumlarını belirten parametreleri CCALC'ın bulduğu plan geçmişinden elde et;

      // eğer çarpışma tespit edilmişse, i'inci adımda icra edilen eylemi ve yükün konumunu CCALC'ın bulduğu plan geçmişinden elde et;
      collision, L, A ← trajectoryCollision(Δ);

      i++;
    end while

    if ¬collision then
      return P
    else
      P ← Planlama problemini, yük L konumunda iken A eylemini icra etmeyecek şekilde değiştir;
      plan := false;
    end if
  end if
end while
```

4. Planın LEGO Robotları Üzerinde Gerçekleştirilmesi

Eylem tanım kümesinde, eylemler robotların hareketlerinin birer doğru boyunca olmasını öngörecek şekilde tanımlanmıştır. Bu nedenle, CCALC tarafından oluşturulan planın takibi için, plan geçmişinde yer alan noktalar arasında robot sonlandırıcılarının doğrusal hareketini sağlayacak motor açılarının hesaplanmasına ihtiyaç vardır. Ancak, NXC dilinin kullandığı yerleşik yazılımda sadece tam sayıların tanımlı olması ve NXT'nin gerçek zamanda doğrusal yörüngenin hesaplanması için gerekli olan işlem yükünü karşılayamaması nedeniyle bu hesaplama gerçekleştirilememektedir. Bunun yerine, hesaplanan plan geçmişinde yer alan noktalar arasında eş aralıklarla ara değerler tanımlanarak doğrusal harekete olabildiğince yakın bir hareket hedeflenmiştir. Aradeğerleme için, Şekil 2'de temsili olarak gösterilmiş olan sistem için, ters kinematik denklemleri çıkartılmış ve bu denklemler kullanılarak motor açıları hesaplanmıştır. Hesaplanan motor açıları Bluetooth iletişimi aracılığıyla LEGO robotlarına gönderilip, robotların istenen yörüngeyi takibi için orantısal geri-beslemeli kontrol uygulanmıştır.

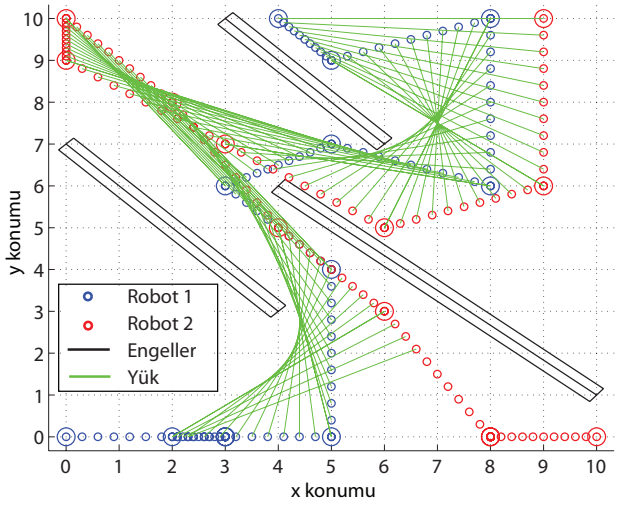
5. Deneysel Sonuçlar

Örnek bir planlama probleminin, yukarıda anlatılan sistem ile çözülmesiyle elde edilen planların uygulaması aşağıdaki



Şekil 2: Yük tasıyan iki robota ait temsili şema

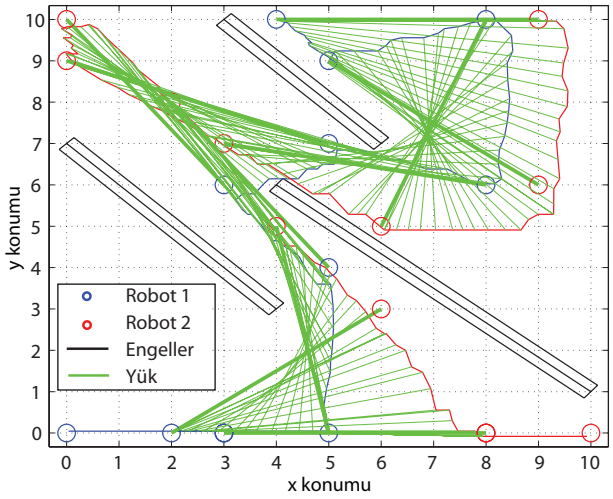
şekillerde görülebilir. Şekil 3, planda elde edilen ve robotların sonlandırıcılarını gösteren noktaların aradeğerleme ile çoğaltılması ile elde edilen referans yörüngeyi göstermektedir. Şekil 4, robotların bu referans yörüngeyi izlerken motor kayıtlarından (düz kinematik denklemler çözülerek) elde edilen, robot sonlandırıcılarının ve yükün pozisyonlarını ve oryantasyonlarını göstermektedir. Bu şekillerde, yeşil çizgiler yükü, siyah çizgiler engelleri, kırmızı ve mavi çemberler robotların istenen pozisyonlarını, ve kırmızı ve mavi çizgiler ise robotların gerçekte takip ettiği yörüngeleri göstermektedir. Şekil 5 ise, bulunan planın robotlar tarafından icra edilmesi sırasında, her bir durumun gerçekleştiği ekran kopyalarını göstermektedir. Farklı örnekler için [20]'a bakınız.



Şekil 3: Yörünge aradeğerlemesi

6. Sonuç

Bu makale, mantığa dayalı otomatik akıl yürütme gösterimlerinin ve sistemlerinin, robot kontrolü için nasıl kullanılabileceği konusunda yeni yöntemler ve algoritmalar sunmaktadır. Bu yöntemler ve algoritmaların uygulanabilirliği ve etkinliği, LEGO MINDSTORMS NXT robotları üzerinde bazı örnek planlama problemleriyle gösterilmiştir. Geliştirilen sistemin, plan icraatının takibini de içerecek şekilde, genelleştirilmesi ve farklı robotlarda uygulanması üzerinde çalışmalarımız devam etmektedir.



Şekil 4: Deneysel olarak gerçekleşen yörünge

7. Kaynakça

[1] Levesque, H., Lakemeyer, G.: Cognitive robotics. In: Handbook of Knowledge Representation, Elsevier (2007)

[2] Levesque, H.J., Pagnucco, M.: Legolog: Inexpensive experiments in cognitive robotics. In: Proc. of CogRob. (2000) 104–109

[3] McCarthy, J.: Situations, actions, and causal laws. Technical report, Stanford University (1963)

[4] Levesque, H.J., Pirri, F., Reiter, R.: Foundations for the situation calculus. ETAI **2** (1998) 159–178

[5] Levesque, H.J., Reiter, R., Lin, F., Scherl, R.B.: GOLOG: A logic programming language for dynamic domains. JLP **31** (1997)

[6] Hähnel, D., Burgard, W., Lakemeyer, G.: GOLEX - bridging the gap between logic (GOLOG) and a real robot. In: Proc. of KI. (1998) 165–176

[7] Ferrein, A., Fritz, C., Lakemeyer, G.: Using GOLOG for deliberation and team coordination in robotic soccer. Künstliche Intelligenz **1** (2005)

[8] Doherty, P., Granlund, G., Kuchcinski, K., Sandewall, E., Nordberg, K., Skarman, E., Wiklund, J.: The WITAS unmanned aerial vehicle project. In: Proc. of ECAI. (2000) 747–755

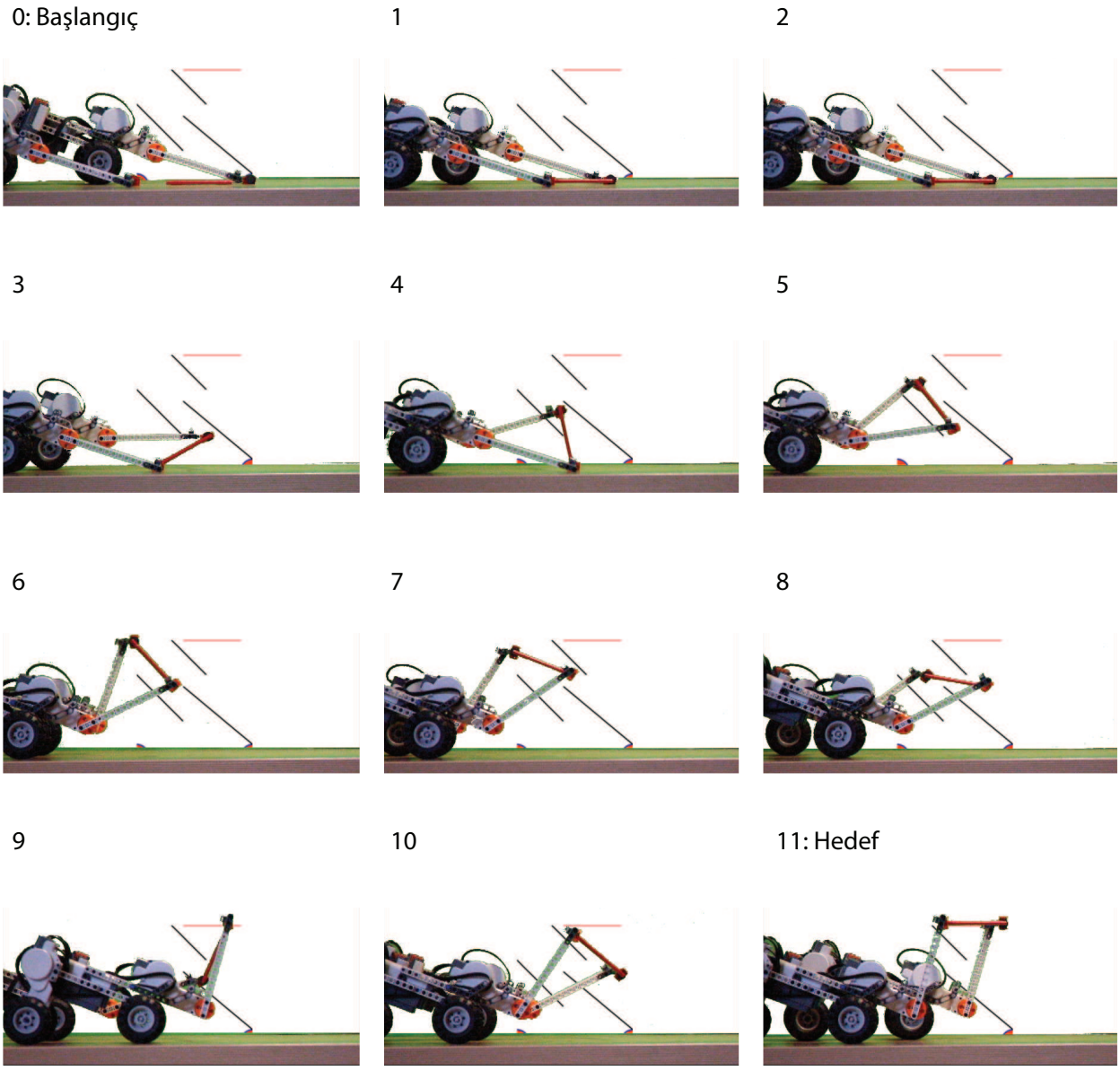
[9] Doherty, P., Gustafsson, J., Karlsson, L., Kvarnström, J.: TAL: Temporal action logics language specification and tutorial. ETAI **2** (1998) 273–306

[10] Sandewall, E.: Features and Fluents: A Systematic Approach to the Representation of Knowledge about Dynamical Systems. Oxford University Press (1994)

[11] Sandewall, E.: Cognitive robotics logic and its metatheory: Features and fluents revisited. ETAI **2** (1998) 307–329

[12] Shanahan, M., Witkowski, M.: High-level robot control through logic. In: ATAL. (2000) 104–121

[13] Kowalski, R., Sergot, M.: A logic-based calculus of events. New Gen. Comput. **4**(1) (1986) 67–95



Şekil 5: Plandaki durumları gösteren ekran kopyaları

[14] Miller, R., Shanahan, M.: The event calculus in classical logic - alternative axiomatisations. ETAI **3**(A) (1999) 77–105

[15] Thielscher, M.: Introduction to the fluent calculus. ETAI **2** (1998) 179–192

[16] Thielscher, M.: FLUX: A logic programming method for reasoning agents. TPLP **5**(4-5) (2005) 533–565

[17] Fichtner, M., Großmann, A., Thielscher, M.: Intelligent execution monitoring in dynamic environments. In: Proc. of Workshop on Issues in Designing Physical Agents for Dynamic Real-Time Environments: World modeling, planning, learning, and communicating. (2003)

[18] Giunchiglia, E., Lee, J., Lifschitz, V., McCain, N., Turner, H.: Nonmonotonic causal theories. AIJ **153** (2004)

[19] Gelfond, M., Lifschitz, V.: Action languages. ETAI **2** (1998) 193–210

[20] Caldiran, O., Haspalamutgil, K., Ok, A., Palaz, C., Erdem, E., Patoglu, V.: Bridging the gap between high-level reasoning and low-level control. In: Proc. of LPNMR. (2009) To appear.