Push Down Automata

Eylül 15, 2009 · Şadi Evren ŞEKER · algoritma analizi (teory of algorithms)/ Automata (otomatlar, özdevinirler) · 6 Comments

Yazan : Şadi Evren ŞEKER

Aşağı sürüklemeli otomatlar (push down automaton) yapı olarak birer otomat makineleridir. ~~Normal bir sonlu otomattan farkı, belirli (deterministic) olması ve ilave bir yığın (stack) bulundurmasıdır. Yani makinemiz basitçe her adımda ne yapacağından tam olarak emindir (belirli ,deterministict) ve veri depolamak için hafızada bulunan bir yığından (stack) istifade edebilir.~~ Düzeltme (Tarık Bey'e teşekkürler): PDA'ler için kullanılan otomata (automaton) göre belirli (deterministic) veya belirsiz (nondeterministic) olma ihtimali vardır. Yani kullanılan otomat belirliyse (Deterministic) bu PDA de belirli aşağı sürüklemeli otomat (Deterministic Push Down Automaton DPDA) olarak isimlendirilir. Şayet tersine kullanılan otomat belirsizse (non deterministic) bu durumda otomat, belirsiz aşağı sürüklemeli otomat (NonDeterministic PushDownAutomata NPDA) olarak isimlendirilir. Bir sonlu otomattan (finite state machine) farkı ise yığın (stack) kullanılmasıdır.

Bilindiği üzere yığın (stack) yapısının temel iki fonksiyonu bulunur. Koyma (push) ve alma (pop) işlemleri isimlerinden de anlaşılacağı üzere yığına koyma ve yığından bir veriyi alma işlemini gerçekleştirir. PDA içerisinde bu fonksiyonlar aynen bulunur. Buna ilave olarak bir pda'i açıkça tanımlayabilmek için 6 bilgi gerekir. Bu bilgiler aşağıdaki şekilde gösterilebilir:

$(Q,\Sigma,\Gamma, \delta ,q_0,F)$

Yukarıdaki satır bir aşağı sürüklemeli otomat için kabul edilen en standart gösterim şeklidir. Kaynaklarda aynı bilgiyi ifade etmek için farklı semboller kullanılmaktadır ancak semboller değişse de pda için bu bilgiler gerekir:

Q: makinemizde (otomatımızda ,automaton) bulunan durumları (states) gösterir.

Σ: makinemizin kabul ettiği girişte kullanılabilecek alfabenin (alphabet) kümesidir.

Γ: yığında (stack) kullanılabilecek alfabenin (alphabet) kümesidir.

δ: Q ile gösterilen durumlar (states) arasındaki geçişlerin kümesidir.

q_0: başlangıç durumudur (initial state)

F: Bitiş durumudur (final state)

Yukarıdaki bu akademik gösterim ile neyin kastedildiğini bir örnek üzerinden anlamaya çalışalım. Örnek olarak çok klasik bir makine olan 0^n1^n probleminin çözüm makinesini pda olarak göstermek isteyelim. Diğer bir deyişle makinemiz bir giriş kelimesini alacak ve bu kelimedeki 0'ların sayısı 1'lerin sayısına eşitse ve o'lar 1'lerden önce geliyorsa bu kelimeyi kabul edecek, şayet 0'ların sayısı ve 1'lerin sayısı eşit değil veya sıralamada bir hata varsa bu girdiyi kabul etmeyecek.

Çözmeye çalıştığımız problemi daha iyi anlayabilmek için bir iki girdinin kabul edilip edilmeyeceğini inceleyelim:

λ : kabul, n= 0 için doğru (burada λ sembolü ile boş girdi kastedilmiştir)

01 : kabul , n = 1 için doğru

10 : ret , sıralama hatası, 0'lar 1'lerin önünde olmalı

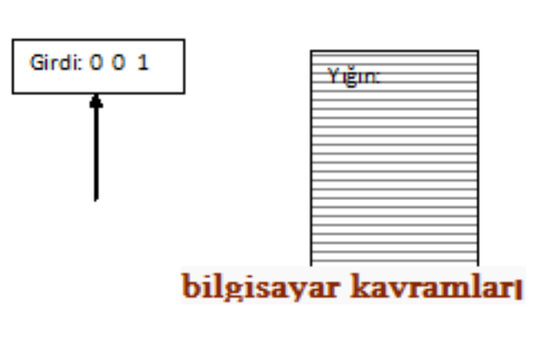
0011: kabul n = 2 için doğru

0101 : ret, sıralama hatası , bütün 0'lar, 1'lerin önünde olmalı

00011: ret, 0'ların sayısı ile 1'lerin sayısı tutmuyor

Şimdi yukarıdaki problemin çözümü olan aşağı sürüklemeli otomatımızı tasarlayalım. Yukarıdaki tanımda 6 unsurun bulunması gerektiğinden bahsetmiştik. Sırayla bunlara cevap arayalım. Önce makinemizi tasarlayarak başlayalım. Makinenin tasarımı için bir yığın (stack) kullanacağız. Biliyoruz ki şayet makineye gelen 0'ları sırasıyla koyarsak (push) ve 0'lar bittikten sonra gelen her 1 için yığından bir eleman alırsak (pop) bu durumda makine girdi kelimesi bittiğinde yığını boş bulursa (yığındaki harfler ile girdideki harfler aynı anda biterse) kelimeyi kabul edecek aksi halde reddedecektir.

Bu tasarımı bir iki örnek ile anlamaya çalışalım. Örneğin girdimiz 0011 olsun.



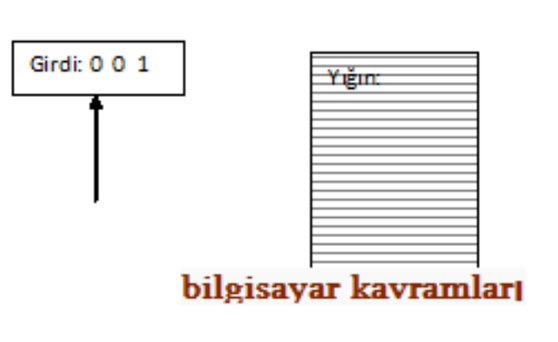
İlk durumda makinemiz girdinin ilk harfi olan 0'ı okuyacak ve 0 gördükçe yığına koyacak (push)

Yığına 0'ları koyduktan sonra her gördüğü 1 için yığından bir 0 çıkaracak (pop):

Görüldüğü üzere 2 adet 1 harfi için 2 adet 0 harfi yığından çıkarılmıştır (pop) ve sonuçta girdi kelimenin sonuna ulaşılmış ve yığında aynı anda boşalmıştır. Dolayısıyla 0011 kelimesini kabul ederiz.

Bu yazı şadi evren şeker tarafından yazılmış ve bilgisayarkavramlari.com sitesinde yayınlanmıştır. Bu içeriğin kopyalanması veya farklı bir sitede yayınlanması hırsızlıktır ve telif hakları yasası gereği suçtur.

Benzer makinemiz için bu sefer reddedilecek bir girdiyi tecrübe edelim. Misal girdimiz 001 olsaydı ne olurdu?

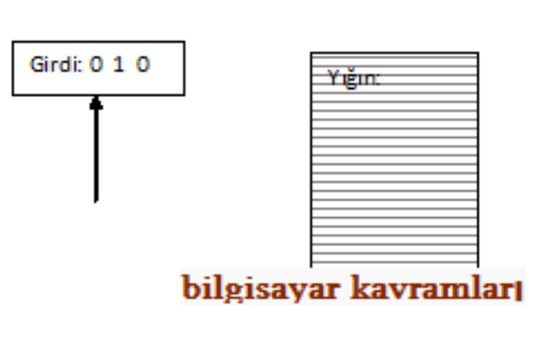


İlk durumda makinemiz girdinin ilk harfi olan 0'ı okuyacak ve 0 gördükçe yığına koyacak (push)

Yığına 0'ları koyduktan sonra her gördüğü 1 için yığından bir 0 çıkaracak (pop):

Görüldüğü üzere girdi kelimesinin sonuna gelindiğinde hâlâ yığında bir harf bulunmakta bu durumda girdiyi reddetmek zorundayız.

Farklı bir örnek olarak 0101 girdisini tecrübe edelim:

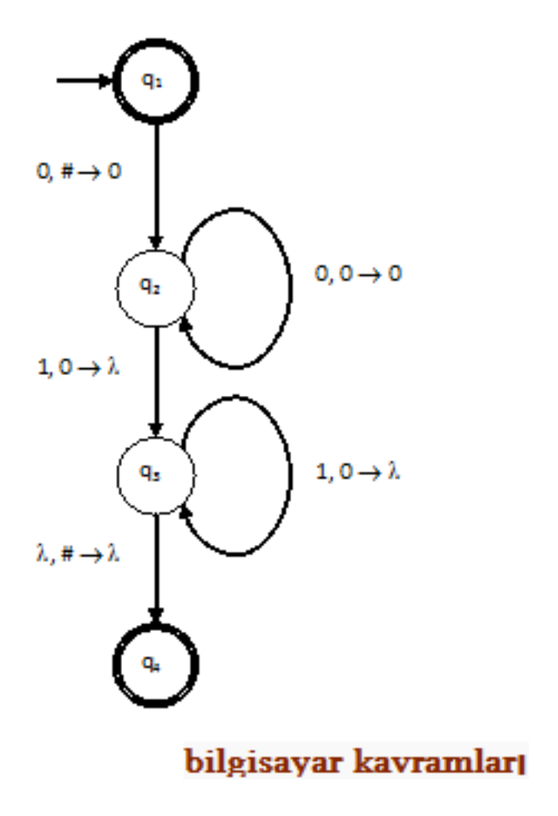


İlk durumda makinemiz girdinin ilk harfi olan 0'ı okuyacak ve 0 gördükçe yığına koyacak (push)

Yığına 0'ları koyduktan sonra her gördüğü 1 için yığından bir 0 çıkaracak (pop):

Bu defa sıfırları yığına koyup ardından her 1 için bir çıkarma (pop) işlemi yaptıktan sonra hâlâ girdi kelimesinin bitmediğini görüyoruz. Bu durumda girdi kelimemizi kabul etmiyoruz.

Tasarımını ve testini yaptığımız yukarıdaki makinemizi bir sonlu durum makinesi olarak çizmek istersek aşağıdaki şekli elde ederiz:



Yukarıdaki makine tasarımımızı kısaca gözden geçirecek olursak. Kabul edilen durumlar q_1 ve q_4 durumlarıdır. Yani λ (boş kelime) durumunu kabul için q_1 diğer kabul edilir durumlar için de q_4 durumu (state) tasarlanmıştır. Tasarımda bulunan q_2 durumu geçiş durumudur. Yani q_2 durumunda bulunduğu sürece makine girdiden bir harf okumakta ve bu harf 0 olmaktadır. Okunan harf 0 olduğu sürece de bu değer yığına (stack) konulmaktadır (push). Bu durum (yani q_2 durumu) girdiden bir harf olarak 1 gelmesinde bozulur. Şayet girdiden 1 harfi okunursa bu defa durum değiştirilerek q_3 durumuna geçilir ve bu q_3 durumunda da girdiden 1 harfi okundukça yığından 0 harfi çıkarılır (pop).

Son durumda şayet girdi boşsa ve yığın da boşsa q_4 durumuna yani kabul durumuna geçilir. Bunun dışındaki ihtimallerde q_2 yada q_3 gibi kabul edilmeyen bir duruma takılır ve makinemiz girdiyi reddeder.

Bu yazı şadi evren şeker tarafından yazılmış ve bilgisayarkavramlari.com sitesinde yayınlanmıştır. Bu içeriğin kopyalanması veya farklı bir sitede yayınlanması hırsızlıktır ve telif hakları yasası gereği suçtur.

Yukarıdaki makinemiz görüldüğü üzere başarılı bir şekilde çalışıyor. Bu makineyi bir aşağı sürüklemeli otomat (push down automaton) olarak yazacak olursak aşağıdaki tanımları yapmamız gerekir:

Σ={0,1} olacaktır çünkü girişte ya 0 ya da 1 gelebilmektedir. Bunun dışında bir harfin gelmesi söz konusu değildir.

Γ={0,#} olacaktır. Burada yığında olabilecek harfler kümesi gösterilirken dikkat edilirse yığına sadece 0 harfini koyuyoruz (istenildiği kadar). Dolayısıyla 1 harfi bu kümede bulunmaz. Bu kümede bulunan # sembolü ise yığının boş olduğunu ifade için konulmuştur. Yani yığın boş olduğunda bunu da bir şekilde göstermemiz gerekir. Bu boş durumu temsilen # sembolü kullanılmıştır. Yine farklı kaynaklarda farklı geçen bir semboldür ancak anlamsal olarak bütün PDA gösterimlerinde böyle bir sembol bulunmalıdır.

Q= { q_1, q_2, q_3, q_4 } olarak yazılabilir çünkü yukarıdaki makine tasarımında da görüldüğü üzere 4 durum bulunmaktadır.

F = { q_1, q_4 } olarak yazılabilir çünkü yukarıdaki makinede kabul edilen 2 durum vardır ve bunlar da q_1 ve q_4 durumlarıdır.

q_0 = { q_1 }olarak yazılabilir. Burada şekilden de anlaşılacağı üzere başlangıç durumumuz q_1 durumudur.

Son olarak sonlu durum makinemizin (finite state machine) tasarımını göstermemiz gerekmektedir. Bu gösterim için yine farklı kaynaklarda farklı yöntemler kullanılmaktadır. Örneğin yukarıdaki şekilde bir sonlu durum makinesi çizilmesi yeterli görülebilirken bazı kaynaklarda aşağıdaki gösterim kullanılmıştır. Hangi gösterim kullanılırsa kullanılsın sonuçta bu adımda anlatılan şey bir sonlu durum makinesidir ve bu makinede bulunan ve makineyi bir PDA yapan ise bir yığın (stack) kullanılmasıdır.

Giriş	0			1			λ		
Yığını	0	#	λ	0	#	λ	0	#	λ
q_1			q_2,0						
q_2				q_3, λ					
q_3				q_3, λ				q_4, λ	
q_4									

Yukarıdaki gösterim aslında şekil olarak çizilen sonlu durum makinesini tablo olarak göstermekten başka bir işe yaramaz. Bu tabloda 9 sütün bulunmaktadır. Bunlar sırasıyla 0,1 ve λ durumlarından yine 0,1 ve λ durumlarına geçişleri gösterir. Örneğin tablonun ilk satırı 0'dan 0 durumuna geçişi son sütünü ise λ'dan λ durumuna geçişi gösterir.

Tablodaki 4 satır ise makinemizdeki durumları gösterir. Makinenin anlamlı olabilmesi için başlangıç durumunun { q_1 } olduğunu ve kabul edilir bitiş durumlarının { q_1, q_4 } olduğunu bilmek gerekir (ki bu bilgiyi zaten veriyoruz).

Yukarıdaki tablonun okunmasını daha net anlamak için bir örnek durumu inceleyelim. Örneğin makinemiz girdide bulunan bütün 0'ları okumuş ve yeni bir harf olarak 1 ile karşılaşmış olsun. Bu durumda q_2 durumundan q_3 durumuna geçiş yapması gerekir. Şekilde görüldüğü üzere bunu yapacağı tek durum bir 1 okunması halinde q_3 geçmesini söyleyen satırdır.

İlgili Yazılar

- Ters Parça Algoritması (Reverse Factor Algorithm)
- Smith Waterman Dizgi Yaslama (String Alignment) Algoritması
- Tarak Sıralaması (Comb Sort)
- Branch and Bounding (Dallanma ve Sınırlandırma Yaklaşımı)
- Permütasyon Sıralaması (Permutation Sort)
- Gnome Sıralaması (Gnome Sort)

Yorumlar

çağatay — Reply
Ekim 3, 2009 at 12:09 am

Fazla Türkçe açıklamalı kaynak bulamamıştım. PDA'nın temelini anlamak için çok yararlı oldu. Emeğinize sağlık 🙂

tarık-ege.üni — Reply
Aralık 12, 2009 at 1:53 am

Tek Sayıda Sembollü Palindrome ve çift Sayıda Sembollü Palindrome için PDA çizilebilir.bu ikisi de non deterministiktir.ondan dolayı pda ,deterministik olduğu gibi non deterministikte olabilir…girişte hata olabilir…

Şadi Evren ŞEKER Article Author — Reply
Aralık 12, 2009 at 2:02 pm

evet halısınız, pda'ler belirli (deterministic) veya belirsiz (nondeterministic) olabilir, kullanılan otomata (automaton) göre tipi belirlenir. Yazı aslında belirli otomatlar hedef alınarak yazılmıştı ancak uyarınız sonucunda gerekli düzetlmeyi yapıyorum.

Teşekkürler

kudret — Reply
Mayıs 1, 2012 at 3:33 pm

cok faydalı oldu dersim için çok tesekkur ederim.

Emir — Reply
Aralık 9, 2012 at 12:38 am

ilk örnekte 0011 örneği verilmek istenmiş ama 001 olmuş.

Büşra — Reply
Ocak 6, 2016 at 11:03 pm

Türkçe kaynak sıkıntısı çekerken bu siteye denk gelmek çok iyi oldu.Çok faydalı oldu Evren hocam emeğiniz için teşekkürler

Bir Cevap Yazın

E-posta hesabınız yayımlanmayacak. Gerekli alanlar * ile işaretlenmişlerdir

İsim *

E-posta *

İnternet sitesi

Yorum gönder

Kategoriler

- algoritma analizi (teory of algorithms
- Automata (otomatlar, özdevinirler
- bilgisayar felsefesi
- Bilgisayar Grafiği (Computer Graphics
- Bilgisayar Kavramları
- Bilgisayar Matematiği
- Bilgisayar Standartları
- C/C++
- Derleyiciler
- Doğal Dil İşleme (NLP
- Donanım (Hardware
- Dosya Organizasyonu (File Organisation
- graf teorisi (graph theory, çizge kuramı
- işletim sistemleri
- JAVA
- Kod Örnekleri
  - Scheme (Lisp
- Kuantum Hesaplama
- Mantık Devreleri (Logic Circuits
- MIS (Yönetim Bilişim Sistemleri
- Nesne Yönelimli Programlama
- Network(Ağ
  - Web Teknolojileri
- Programlama Dilleri
- Resim İşleme (Image Processing
- Scheme (lisp
- Sınavlar
- Sistem Programlama (System Programming
- Son Kullanıcı
- Temel Bilimler
- Uncategorized
- Veri Güvenliği(Cryptography
- Veri Madenciliği (Data Mining
  - Metin Madenciliği (Text Mining
- Veri Madenciliği (Data Mining
- Veri Sıkıştırma (Data Compression
- Veri Tabanı (Database
- veri yapıları
- yapay zeka (artificial intelligence
  - Yapay Sinir Ağları (Artificial Neural Networks
- Yazılım Mühendisliği (Software Engineering

Son Eklenenler

- Rapid Miner ile K-NN Uygulaması
- Kemmi Normalleştirme (Quantile Normalization)
- Ters Parça Algoritması (Reverse Factor Algorithm)
- Genetik Programlama (Genetic Programming)
- JAVA dilinde WEKA ile kod geliştirilmesi
- Eş-Eğitim (Co-Training)
- Metin Madenciliği (Text Mining)
- BT'nin İşletmelere Etkisi
- Ana Üretim Planlaması (Master Manufacturing Planning)
- BOM (Bill of Materials) Ürün Reçetesi

Meta

- Giriş
- Yazılar RSS
- Yorumlar RSS
- WordPress.org