ARA KOD ÜRETİMİ

1

Arakod Üretimi

- Sözdizim denetimi tamamlanan sözcük katarı için arakod üretilecektir
- Doğrudan makina kodu üretmektense, arakod üretmenin nedenleri şunlardır:
  - Basit, bu nedenle de üretilmesi kolay olması
  - İşlemci türünden bağımsız olması
  - Yine de işlemcinin yürüteceği makina koduna yakın olması
  - Eniyileme işlemleri için üzerinde rahat çalışılabilir bir kod oluşması

2

Arakod Üretimi(2)

- Ayrıştırıcı, verilen giriş katarının, bir gramerin tanımladığı dile ait bir cümle olup olmadığını belirler
- Ayrıca, bu giriş katarının yapısını da belirler- katar için bir ayrıştırma ağacı oluşturur
- Bu işlevlerine ek olarak, ayrıştırıcı, gramerin belirli türetim kuralları için, onlarla ilişkilendirilmiş olan işlem dizileri de yürütebilir
- Bu işlemler genellikle ayrıştırıcının en önemli çıktısını oluştururlar

3

Sözdizim ve Anlambilim(Syntax–Semantics)

- Dilin sözdizimi gramer kuralları ile tanımlanır
- Ancak, çeviri sırasında gerek duyulan bir çok bilgi türetim kuralları ile tanımlanamaz
- Türetim kuralları ile tanımlanamayan bu bilgiler "dilin anlamsal özellikleri" olarak bilinirler (semantic features) ve anlamsal kurallarla belirlenirler
- Sözdizim kurallarını tanımlamak için notasyonlar var olduğu halde (örneğin, BNF formu), anlamsal kurallar genellikle sözlü ifadelerle belirlenir

4

Anlamsal Kurallar ve Gramer Simgeleri Nitelikleri

- Anlamsal kurallar nasıl uygulanabilir?
- Ayrıştırma ağacındaki her düğüm bir gramer simgesine karşı düşer. Gramer simgeleri (ağaç düğümleri) kendileriyle ilişkilendirilen niteliklere sahip olabilirler
  - Nitelik belirli bir amaç için gerekli olan herhangi bir bilgi olabilir, örneğin tip, adres, bir sayı, bir işaretçi, vs
  - Ayrıştırma ağacı düğümünü bir kayıt yapısı olarak düşünürsek, her nitelik bilgisi bu yapının bir alt alanı olacaktır

5

Anlamsal Kurallar ve Gramer Simgeleri Nitelikleri(2)

- Anlamsal kuralları uygulamak için gramer simgeleri niteliklerinin sözdizimle yönlendirilmiş tanımlarından yararlanılır
- Her gramer kuralının kendisiyle ilişkilendirilmiş olan bir anlamsal kurallar kümesi bulunur.
- Anlamsal kurallar, o gramer kuralında yer alan simgelere ait niteliklerin ne şekilde hesaplanacağını belirler ve diğer işlemleri (sembol tablosuna bilgi girişi, arakod üretimi, vs)yönlendirirler

6

Ayrıştırma Ağacı Düğümlerinin Nitelikleri

- Gramer türetim kuralı: A→ X_1 X_2 ... X_n
- Ayrıştırma ağacında, A bir ara düğümde, X_i de bu düğümün alt düğümlerinde yer alacaklardır
- Bu türetimle ilişkili bir anlamsal kural, türetim içinde yer alan herhangi bir simgenin niteliğinin, diğer simgelerin niteliklerine bağlı olarak hesaplanmasını sağlar

7

Hesaplanmış ve Edinilmiş Nitelikler

- Gramer türetim kuralı: A→ X_1 X_2 ... X_n
- Hesaplanmış Nitelik (Synthesized Attribute)

  Bir anne düğümün niteliği, çocuk düğümlerinin nitelik bilgileri kullanılarak hesaplanır (A'nin niteliği, Xi simgelerinin o anda sahip oldukları nitelik değerlerine bağlı olarak belirlenir)
- Edinilmiş Nitelik (Inherited Attribute)

  Bir düğümün niteliği, anne ve/veya çocuk düğümlerinin nitelik bilgileri kullanılarak hesaplanır

8

Hesaplanmış ve Edinilmiş Nitelikler(2)

- Hesaplanmış nitelikler, türetimin açılımında yer alan tüm simgelerin nitelik bilgileri bilinir bilinmez hesaplanabilirler.
  - Bu işlem "aşağıdan yukarı" ayrıştırıcı (LR) için daha uygundur. Gerek duyulan tüm bilgiler, indirgeme işleminden hemen önce, yığında yer alır
- Edinilmiş nitelikler, türetim kuralının tanımladığı nonterminale ait bilgileri de gerektirir
  - Bu işlem "yukarıdan aşağı" ayrıştırıcı (LL) için daha uygundur

9

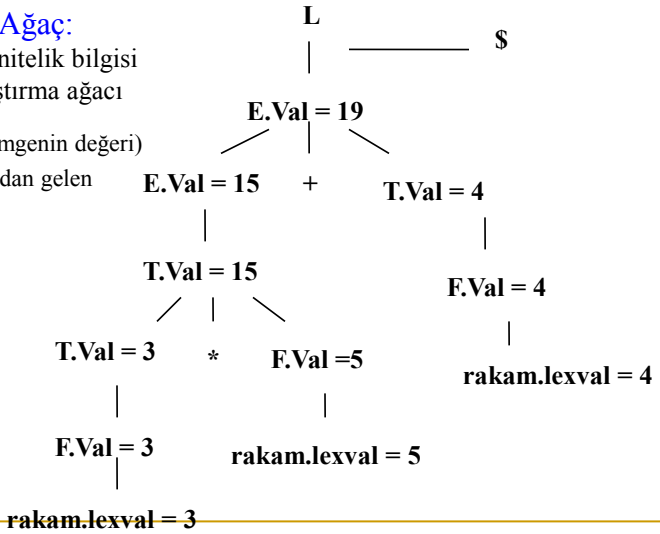
Örnek: Hesap Makinası

Türetim Kuralları	Anlamsal Kurallar
1. L →E $	1. print(E.val)
2. E → E1 + T	2. E.val := E1.val + T.val
3. E → T	3. E.val := T.val
4. T → T1 * F	4. T.val := T1.val * F.val
5. T → F	5. T.val := F.val
6. F → (E)	6. F.val := E.val
7. F → rakam	7. F.val := rakam.lexval

10

Örnek: 3*5+4$

- Donatılmış Ağaç:
  Düğümlerine nitelik bilgisi eklenmiş ayrıştırma ağacı
  - (val niteliği=simgenin değeri)
  - lexval=tarayıcıdan gelen değer



11

Örnek: Tip Bildirimi

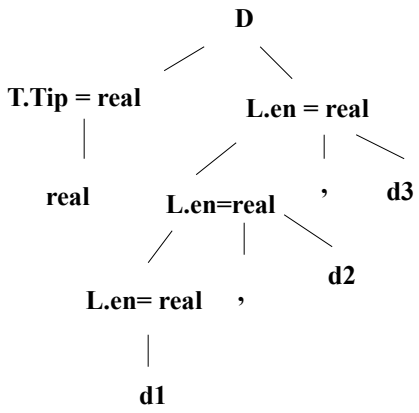
Türetim Kuralları	Anlamsal Kurallar
1. D → TL	1. L.en := T.Tip
2. T → int	2. T.Tip := integer
3. T → real	3. T.Tip := real
4. L → L1, d	4. {L1.en := L.en TipEkle (d, L.en)}
5. L → d	5. TipEkle (d, L.en)

- d: değişken
- en: edinilmiş nitelik
- TipEkle: sembol tablosunda, d kaydına tip bilgisi ekle

12

Örnek: “real id1, id2, id3”

Edinilmiş nitelikler kullanan tip bildirimi: tip bilgisi kökten yapraklara doğru ilerletilir (yukarıdan aşağı ayrıştırma)



13

Hesaplanmış Nitelikler

- Eğer tüm nitelikler hesaplanmış nitelikler ise, aşağıdan yukarı ayrıştırıcı ayrıştırma adımları sırasında nitelikleri de hesaplayabilir
  - Yığındaki nonterminallere nitelik alanları ekle
  - Her indirgeme işleminden önce, o türetimle ilişkilendirilmiş olan anlamsal kuralı yürüt

14

Yürütme Sırasında Yığın

Örnek:

Türetim Kuralı	Anlamsal Kural
A→ X Y Z	A.a = f(X.x, Y.y, Z,z)

İndirgeme adımından hemen önce yığının durumu

YIĞIN

simge	durum	nitelik
Z		Z.z
Y		Y.y
X		X.x
...		...

İndirgeme adımından hemen sonra yığının durumu

A		A.a
...		...

15

Arakod

- Arakod: ayrıştırma adımları sırasında üretilen kod
- Arakod türleri:
  - Postfix
  - Soyut Ayrıştırma Ağaçları (AST)
  - Üç Adresli Kod (dörtlükler, üçlükler)
- İşlemci mimarisinin temel özelliklerini taşır
  - Ardışıl yürütme, dallanma komutları, vs.
- Ancak mimariye özgü ayrıntılardan uzaktır
  - Saklayıcı kullanımı, koşullarını denetimi,vs.

16

Üç Adresli Kod

- Ardışıl yürütülen komutlar dizisi oluşturulur
- Her komutun bir operatör, en fazla iki operand ve bir sonuç alanı bulunur
  - x:=y op z (x,y,z:sabit, geçici değişkenler, değişken, vs.)
- Komutlara etiket eklenebilir
- Operand bir sembol tablosu kaydına işaretçidir

x:=y + z * w

t1:=z * w
t2:=y + t1
x:=t2

17

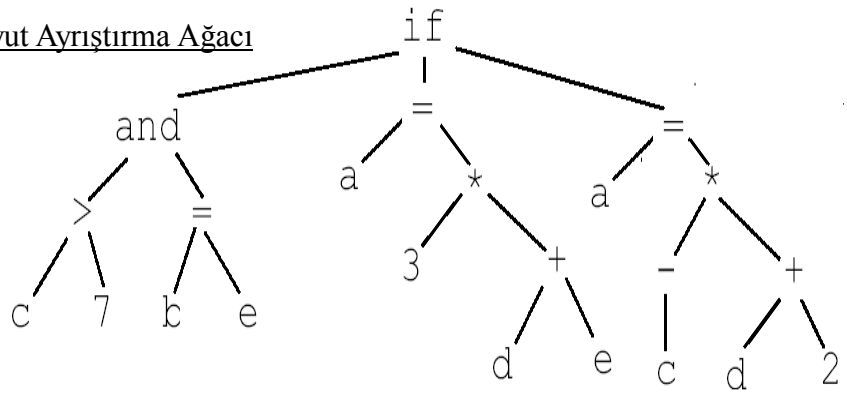
Üç Adresli Komut Türleri

- Atama Komutları
  - **A = B**
  - **A = B op C**
  - **A = op B**
- Altprogram çağrı Komutları
  - **param A**
  - **call P,N**
  - **return N**
- Dizi Erişimi
  - **A[I]**
  - **x:=y[i]**
  - **x[i]:=y**
- Dallanma Komutları
  - **goto L**
  - **if A relop B goto L**
- İşaretçi Komutları
  - **x:=&y**
  - **x:=*y**
  - ***x:=y**

18

Arakod Örneği

Soyut Ayrıştırma Ağacı



if ((c>7) && (b==e)) then a=3 *(d+e)
else a=-c*(d+2)

19

Üretilecek Olan Üç-adresli Arakod

```
      if c > 7 goto L1
      goto L2
L1: if b = e goto L3
L2: t1 := d + 2
      t2 := 3
      t3 := t1 * t2
      a := t3
      goto L4
L3: t4 := d + e
      t5 := -c
      t6 := t4 * t5
L4: a := t6
```

- Ardışıl kod
- Koşullu dallanmalar
- Tüm ara değerler (ağacın ara düğümleri) için geçici değişkenler
- Etkin olmayan kod

20

Üç-adresli Kodun Gerçeklenmesi

- Dörtlükler dizisi ile gösterilir (quadruples)

t_1=- c
t_2=b * t_1
t_3=- c
t_4=b * t_3
t_5=t_2 + t_4
a=t_5

	op	*arg1*	*arg2*	*sonuç*
(0)	t_eksi	c		t_1
(1)	*	b	t_1	t_2
(2)	t_eksi	c		
(3)	*	b	t_3	t_4
(4)	+	t_2	t_4	t_5
(5)	=	t_5		a

21

Sözdizimle Yönlendirilmiş Çeviri

- Sözdizimle yönlendirilmiş çeviri (syntax directed translation)-Ayrıştırma adımlarıyla birlikte yürütülür
- Gramer simgelerine nitelikler eklenir ve her indirgeme adımından önce anlamsal işlem (semantic actions) kümeleri yürütülür. Öteleme adımlarında işlem yoktur.
- Anlamsal işlemlerin görevleri
  - niteliklerin hesaplanması
  - sembol tablosuna girişler
  - gerekli anlamsal kuralların yürütülmesi
  - arakod üretilmesi

22

Atama Deyimlerinin Çevirisi

Gramer:
A → d = E
E → E + E | E * E | -E | (E) | d

Kullanılan nitelikler:

- place (E.place)- ifadenin değerini taşıyacak olan yerin adı (sembol tablosundaki kaydına işaretçi)
- d.place- değişkenleri birbirlerinden ayırdeder

- Kullanılan özel fonksiyonlar:
  - newtemp- Yeni bir geçici değişken yaratır ve geri getirir
  - gen- uygun parametrelerle çağrılır ve bir üç-adresli kod satırı üretir

23

Atama Deyimlerinin Çevirisi (2)

Türetim Kuralı	Anlamsal İşlemler
S → d := E	**{*gen*(d.*place* ‘=’ E.*place*) }**
E → E_1 + E_2	**{E.*place*= *newtemp* ;** *gen*(E.*place*‘:=’E_1.*place*‘+’E_2.*place*) }
E → E_1 * E_2	**{E.*place*= *newtemp* ;** *gen*(E.*place*‘=’E_1.*place*‘*’E_2.*place*) }
E → - E_1	**{E.*place*= *newtemp* ;** *gen*(E.*place* ‘=’ ‘t-eksi’ E_1.*place*) }
E → (E_1)	**{E.*place*= E_1.*place* ;}**
E → d	**{E.*place* = d.*place* ;}**

24

Örnek: A = -B*(C+D)

TÜRETİM	YIĞIN	PLACE	ÜRETİLEN KOD
A = -B * (C+D)			
= -B * (C+D)	d	A	
-B * (C+D)	d =	A --	
B * (C+D)	d = --	A -- --	
* (C+D)	d = - d	A -- -- B	
* (C+D)	d = - E	A -- -- B	T1 = -B
* (C+D)	d = E	A – T1	
(C+D)	d = E *	A – T1 --	
C+D)	d = E * (	A – T1 -- --	
+D)	d = E * (d	A – T1 -- -- C	
+D)	d = E * (E	A – T1 -- -- C	
D)	d = E * (E +	A – T1 -- -- C --	
)	d = E * (E + d	A – T1 -- -- C -- D	
)	d = E * (E + E	A – T1 -- -- C -- D	T2 = C + D
)	d = E * (E	A – T1 -- -- T2	
	d = E * (E)	A – T1 -- -- T2 --	
	d = E * E	A – T1 -- T2	T3 = T1 * T2
	d = E	A – T3	A = T3
	A	A	

A = -B*(C+D)

T1 = -B
T2 = C + D
T3 = T1 * T2
A = T3

25