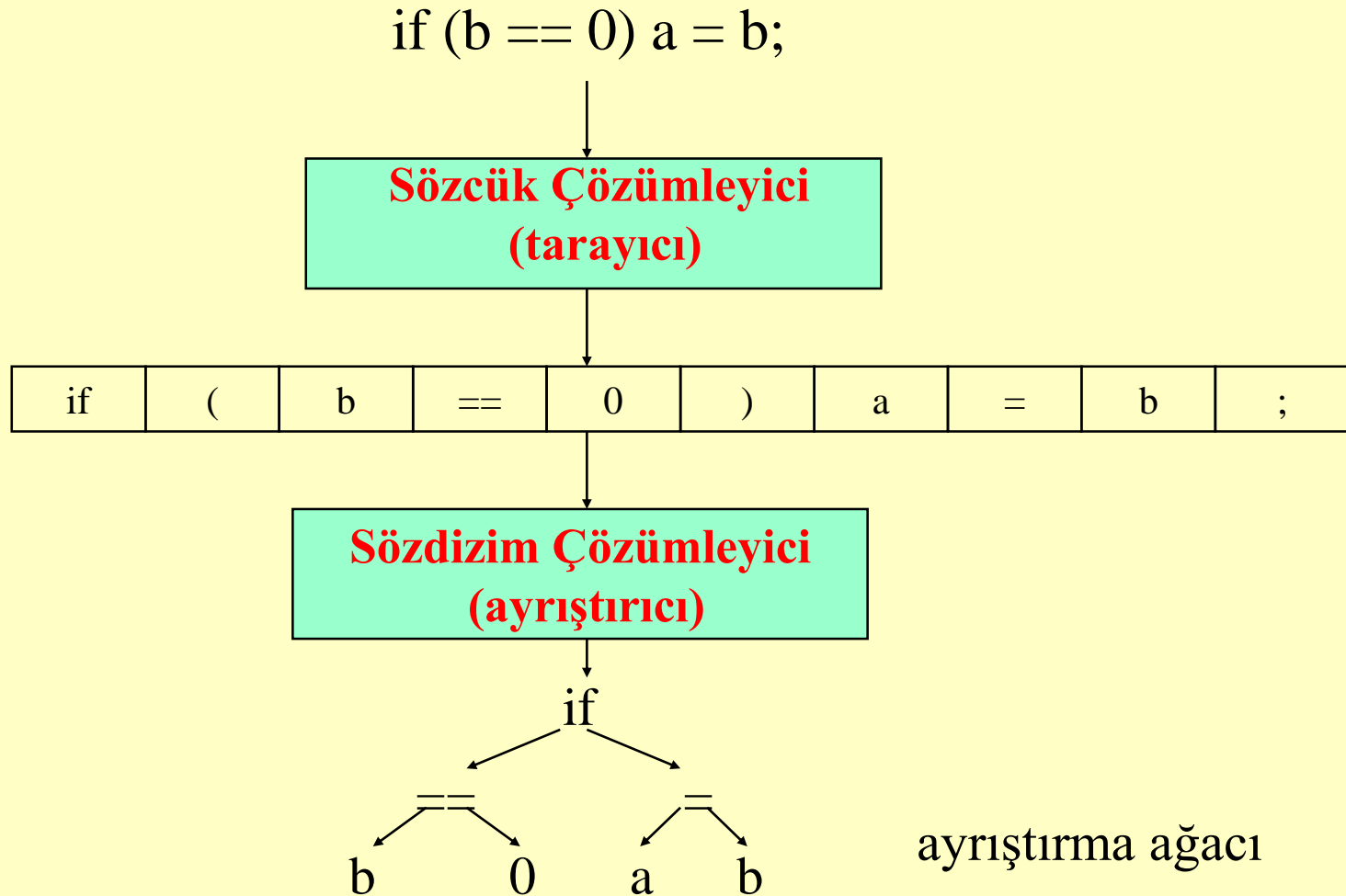


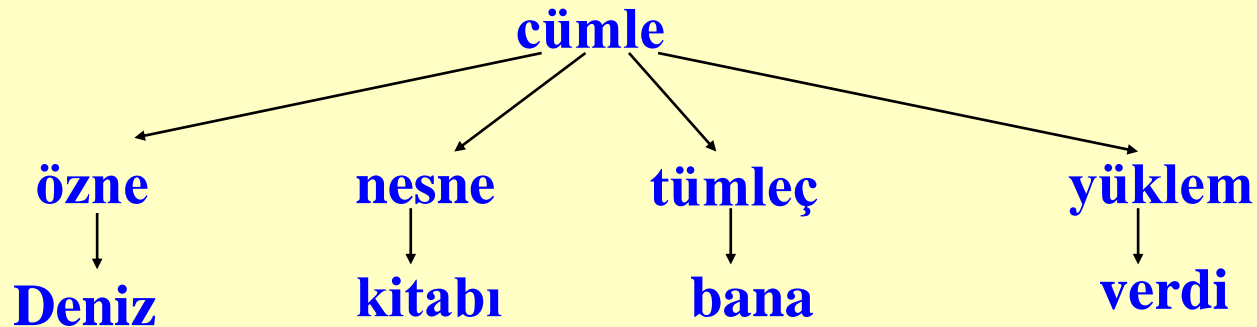
# SÖZDİZİM ÇÖZÜMLEME

# Sözdizim Çözümlemenin Zamanı



# Ayrıştırma Örneği

- Doğal dillerde sözdizim çözümlemesi
  - Cümle yapısının dilbilgisi kurallarına uygunluğunu araştır
  - Her sözcüğün görevini belirle



**“Deniz kitabı bana verdi”**

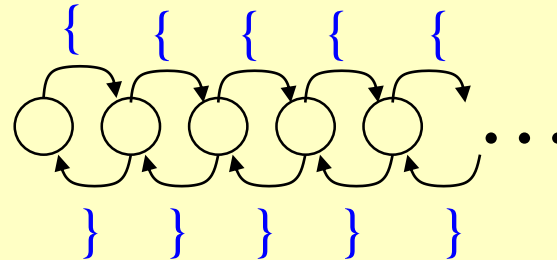
# Sözdizim Çözümlemenin Amacı

- Amaç– Kaynak programı oluşturan sözcüklerin programlama dilinin gramer kurallarına uygun bir sıralamada olup olmadıklarını belirleme
- Bu amaca ulaşmak için neye gerek duyulur?
  - Dilin gramer kurallarını (sentaksını) kesin ve açık bir şekilde tanımlamak için bir notasyon
  - Giriş sözcük katarının dilin sentaksına uygunluğunu belirlemek için bir yöntem
  - Bu gereksinimler karşılanırsa, herhangi bir dil için bir ayrıştırıcı **otomatik** olarak üretilebilir

# Düzgün İfadeler Kullanılabilir Mi?

- Düzgün ifadeler sozcükleri tanımlamak için kullanılıyor ve DFA ile kolayca gerçekleştirilebiliyor. O halde dilin sentaksını belirlemek için de kullanılabilir mi?
  - **HAYIR!** – karmaşık kuralların tanımında yetersiz kalırlar
  - Örnek – İç içe yapılar (bloklar, ifadeler, deyimler) için dengeli parantezler:

{ {} {} {} {} }



# Bağlamdan Bağımsız Gramerler (BBG)

- 4 bileşenden oluşurlar:
  - Terminal simgeler (uç sözcükler) = sözcük veya  $\varepsilon$
  - Non-terminal simgeler = sözdizimsel sınıf veya değişken
  - Başlangıç simgesi  $S$  = özel bir non-terminal
  - $SOL \rightarrow SAĞ$  şeklindeki türetim kuralları
    - $SOL$  = bir tek non-terminal
    - $SAĞ$  = terminal veya non-terminal katarı
    - Non-terminallerin açılımlarını tanımlar
- Gramerin ürettiği dil başlangıç simgesinden yola çıkıp türetim kurallarını tekrar tekrar uygulayarak elde edilen sözcük katarları kümesidir
  - $L(G) = G$  gramerinin ürettiği dil

$S \rightarrow a S a$
$S \rightarrow T$
$T \rightarrow b T b$
$T \rightarrow \varepsilon$

# BBG - Örnek

- Dengeli parantezler dili için gramer
  - $S \rightarrow ( S ) S$
  - $S \rightarrow \varepsilon$ 
    - 1 adet non-terminal:  $S$
    - 2 adet terminal: “(”, “)”
    - Başlangıç simgesi:  $S$
    - 2 adet türetim kuralı
- Eğer bir sözcük katarı, dilin türetim kuralları uygulanarak elde edilebiliyor ise, o katar dilin gramer kurallarına (sentaksına) uygun yapıdadır
  - Örnek katar - “(())”
  - $S = (S) \quad S = (S) \varepsilon = ((S) S) \quad \varepsilon = ((S) \varepsilon) \quad \varepsilon = ((\varepsilon) \varepsilon) \quad \varepsilon = (())$

# BBG Gösterimi

- Nonterminaller genellikle şöyle gösterilir:
  - Küçük harfle isimler – deyim, ifade, vs.
  - Büyük harfler
  - S harfi – gramerin başlangıç simgesi
- Terminaller genellikle şöyle gösterilir:
  - Küçük harfler – a,b,x, vs
  - Özel işaretler – +, \*, (, ), 2,5, vs.
  - Özel sözcükler – if, while, for, vs.
- Yunan alfabesinden harfler, ( $\alpha$ ,  $\beta$ ,  $\gamma$ , vs.) terminal ve/veya nonterminallerden oluşan bir gramer simgeleri katarını gösterir
  - $A \rightarrow \alpha$



# BBG Gösterimi (2)

- $A \rightarrow \alpha_1, A \rightarrow \alpha_2, \dots, A \rightarrow \alpha_k$ 
  - A'nın birden fazla açılımı var ise, şöyle gösterilir  
 $A \rightarrow \alpha_1 \mid A \rightarrow \alpha_2 \mid , \dots , \mid A \rightarrow \alpha_k$
  - “ $\alpha_i$ ” A nonterminalinin açılım “seçenekleridir”
- Aksi belirtilmedikçe, ilk türetimde tanımlanan nonterminal gramerin başlangıç simgesidir
- **Türetim işlemi** – Başlangıç simgesinden başlayarak türetim kurallarının ard arda uygulanması işlemi
- **Kabul** – Verilen sözcük katarını üreten bir türetim işlemleri dizisinin varlığının belirlenmesi

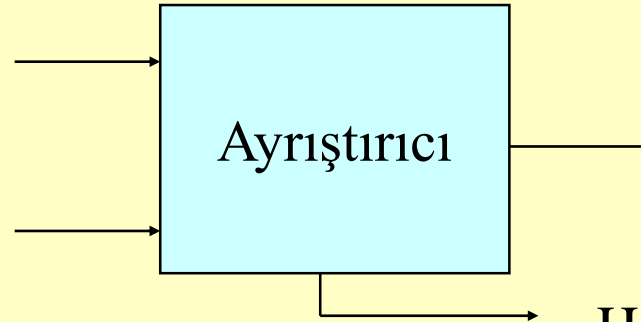
# Ayrıştırıcı

Bağlamdan

Bağımsız

Grameri,  $G$

Sözcük katarı,  $s$   
(tarayıcıdan)



## Sözdizim Çözümleyici (ayrıştırıcı)

- Bir  $G$  gramerini ve “ $s$ ” sözcük katarını giriş bilgisi olarak alır, ve eğer “ $s$ ”  $G$ ’nin tanımladığı dile ait bir katar ise, bu katarı üreten türetim adımlarını verir.
- Aksi durumda, hata mesajları (sentaks hataları) üretir

# Dİ, BBG'nin bir altkümesidir

Her Dİ (düzgün ifade) için bir G grameri tanımlanabilir

$\varepsilon$	$S \rightarrow \varepsilon$
$a$	$S \rightarrow a$
$R1 R2$	$S \rightarrow S1 S2$
$R1 \mid R2$	$S \rightarrow S1 \mid S2$
$R1^*$	$S \rightarrow S1 S \mid \varepsilon$

$G1 = R1$  için tanımlanan,  $S1$  başlangıç simgesine sahip gramer

$G2 = R2$  için tanımlanan,  $S2$  başlangıç simgesine sahip gramer

# Örnek gramer: aritmetik ifade (toplama)

## ■ Gramer

- $S \rightarrow E + S \mid E$
- $E \rightarrow \text{sayı} \mid (S)$

## ■ Genişletilmiş gramer

- $S \rightarrow E + S$
- $S \rightarrow E$
- $E \rightarrow \text{sayı}$
- $E \rightarrow (S)$

4 türetim kuralı

2 non-terminal simge (S,E)

4 terminal simge: “(”, “)”, “+”, sayı

başlangıç simgesi: S

# Türetim İşlemi

- S başlangıç simgesiyle başla
- Türetim kurallarını kullanarak sözcük katarını türet
- $\alpha, \beta, \gamma$  gramer simgeleri katarları ve bir  $A \rightarrow \beta$  türetim kuralı var ise, bir türetim adımı şöyle olacaktır:

○  $\alpha A \gamma \Rightarrow \alpha \beta \gamma$  (A yerine  $\beta$  yerleştir)

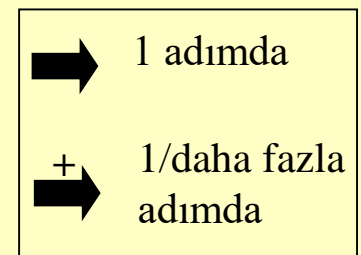
○  $S \xrightarrow{+} \alpha$  ise,  $\alpha$  G gramerinin bir cümlesidir

- Örnek

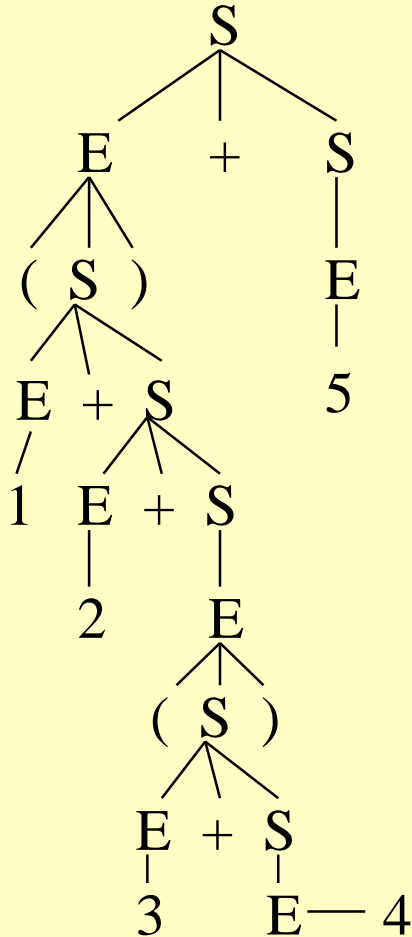
- türetim kuralı:  $S \rightarrow E + S$

- (S + E) + E  $\rightarrow$  (E + S + E) + E

- [S nonterminalini (S) açılımıyla (E + S) yer değiştir ]



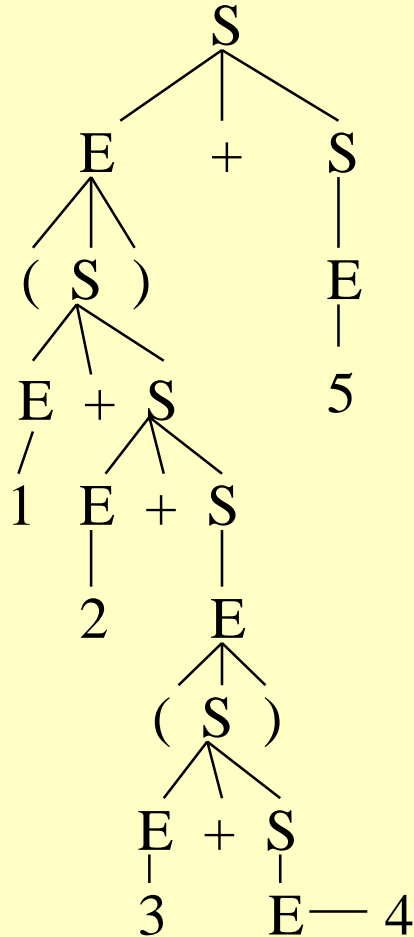
# Ayrıştırma Ağacı



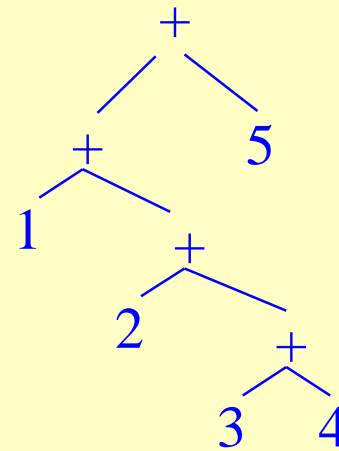
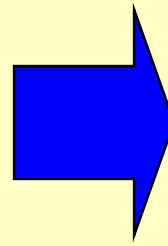
- **Ayrıştırma Ağacı** = Türetim adımlarının bir ağaç yapısı üzerinde gösterilimi
- Ara düğümlerde non-terminal simgeler
- Yapraklarda terminal simgeler (sözcük)
- Ancak türetim adımlarının sırası hakkında bilgi içermez

# Ayrıştırma Ağacı / Soyut Sentaks Ağacı

Ayrıştırma Ağacı (Parse Tree)



Soyut Sentaks Ağacı (Abstract Parse Tree)



SSA içinde gereksiz bilgiler taşınmaz

# Türetim Sırası

- **Türetme adımı:** türetim kuralları herhangi bir sırada seçilebilir, nonterminal simge yerine açılımı (türetim kuralının sağ tarafı) yerleştirilir
- İki standart türetim sırası vardır:
  - En-sağdan türetim
  - En-soldan türetim
- En-soldan türetim
  - Katar içinde **en solda yer alan** nonterminal simgeyi belirle ve bir türetim kuralı uygula
  - $E + S \rightarrow 1 + S$
- En-sağdan türetim
  - Benzer şekilde ancak **en sağda yer alan** nonterminal simgeyi seç
  - $E + S \rightarrow E + E + S$



# Türetim Örnekleri

- $S \rightarrow E + S \mid E$

- $E \rightarrow \text{sayı} \mid (S)$

- En-soldan türetim:  $(1 + 2 + (3 + 4)) + 5$

$$\begin{aligned} S &\rightarrow E + S \rightarrow (S) + S \rightarrow (E + S) + S \rightarrow (1 + S) + S \rightarrow (1 + E + S) + S \rightarrow \\ &(1 + 2 + S) + S \rightarrow (1 + 2 + E) + S \rightarrow (1 + 2 + (S)) + S \rightarrow (1 + 2 + (E + S)) + S \rightarrow \\ &(1 + 2 + (3 + S)) + S \rightarrow (1 + 2 + (3 + E)) + S \rightarrow (1 + 2 + (3 + 4)) + S \rightarrow \\ &(1 + 2 + (3 + 4)) + E \rightarrow (1 + 2 + (3 + 4)) + 5 \end{aligned}$$

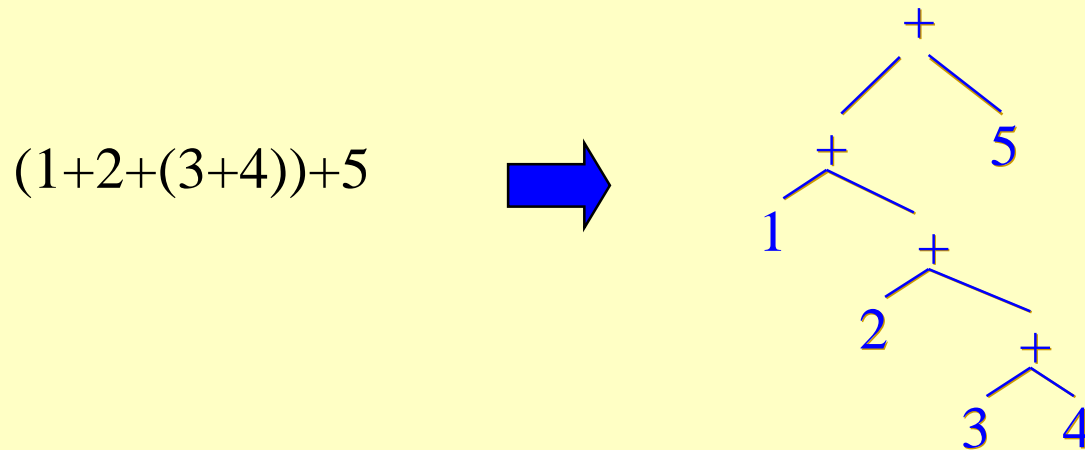
- En-sağdan türetim:  $(1 + 2 + (3 + 4)) + 5$

$$\begin{aligned} S &\rightarrow E + S \rightarrow E + E \rightarrow E + 5 \rightarrow (S) + 5 \rightarrow (E + S) + 5 \rightarrow (E + E + S) + 5 \rightarrow \\ &(E + E + E) + 5 \rightarrow (E + E + (S)) + 5 \rightarrow (E + E + (E + S)) + 5 \rightarrow \\ &(E + E + (E + E)) + 5 \rightarrow (E + E + (E + 4)) + 5 \rightarrow (E + E + (3 + 4)) + 5 \rightarrow \\ &(E + 2 + (3 + 4)) + 5 \rightarrow (1 + 2 + (3 + 4)) + 5 \end{aligned}$$

Sonuç: türetim sırası farklı ancak aynı ayrıştırma ağacı elde edildi

# Belirsiz Gramerler

- Aritmetik toplama ifadesi için, en-sağdan ve en-soldan türetimler aynı ayrıştırma ağacını oluşturmuştur
- Türetim sırasından bağımsız olarak, + operatörü ayrıştırma ağacında sağ taraf ile ilişkilendirilir



# Belirsiz Gramerler (2)

- + sağ taraf ile ilişkilendirilir çünkü türetim kuralı **sağ-rekürsif** özelliğe sahiptir:  $S \rightarrow E + S$
- Başka bir örnek gramer
  - $S \rightarrow S + S \mid S * S \mid \text{say}_1$
  - Hem sağ-rekürsif, hem sol-rekürsif
- **Belirsiz gramer** = aynı cümlenin farklı türetimleri farklı ayrıştırma ağaçları oluştururlar
  - Eğer, aynı cümle için, **iki farklı** en-sağdan/en-soldan türetim var ise, G belirsizdir

# Belirsiz Gramer - Örnek

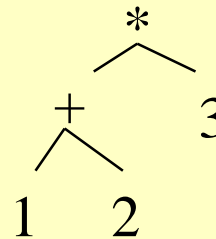
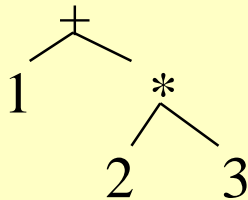
$$S \rightarrow S + S \mid S * S \mid \text{say}_1$$

Aritmetik ifade:  $1 + 2 * 3$

Türetim 1:  $S \rightarrow S + S \rightarrow$   
 $1 + S \rightarrow 1 + S * S \rightarrow 1 + 2 * S \rightarrow$   
 $1 + 2 * 3$

Türetim 2:  $S \rightarrow S * S \rightarrow$   
 $S + S * S \rightarrow 1 + S * S \rightarrow 1 + 2 * S \rightarrow$   
 $1 + 2 * 3$

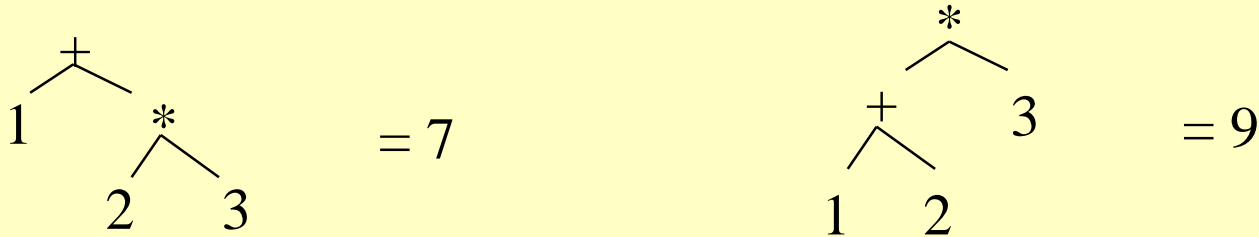
2 en-soldan türetim



Ancak, farklı ayrıştırma ağaçları!

# Belirsizliğin Etkisi

- Farklı ayrıştırma ağaçları farklı değerler üretirler!  
**Sonuç:** programın anlamında tanımsızlık

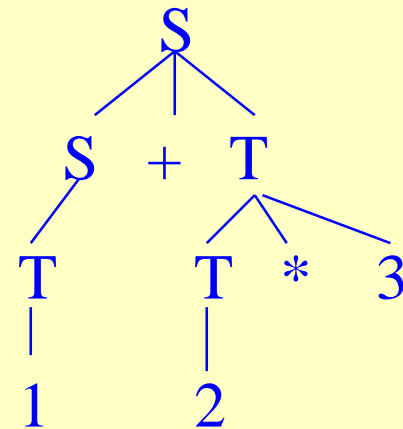


Bir ayrıştırıcı ancak “belirsiz olmayan bir gramer” ile oluşturulabilir!

# Belirsizlikten Kurtulma

- Belirsizlikten kurtulmak için, yeni nonterminal simgeler eklenir ve sadece sol-rekürsif veya sağ-rekürsif yapıya izin verilir

- $S \rightarrow S + T \mid T$
- $T \rightarrow T * \text{sayı} \mid \text{sayı}$



- $T$  non-terminali işlem önceliğini belirler
- Sol-rekürsif yapı; sol taraf ile ilişkilendirme

# Belirsizlikten Kurtulma (2)

- İşlem önceliğinin belirlenmesi
  - Her öncelik düzeyi için ayrı bir nonterminal tanımla
    - T nonterminal simgesi
  - Verilen bir öncelik düzeyinin operatörleri türetim kuralının sağ tarafı olarak belirlenirler
    - $T \rightarrow T * \text{sayı}$
  - Yüksek öncelikli operatörlere erişim bir önceki düzeydeki öncelik non-terminali üzerinden gerçekleşir

# İlişkilendirilme

- Bir operatör sol/sağda yer alan operand ile ilişkilidir veya bu tür ilişki yoktur (yok)
  - Sol:  $a + b + c = (a + b) + c$
  - Sağ:  $a \wedge b \wedge c = a \wedge (b \wedge c)$
  - Yok:  $a < b < c$  tanımsızdır
- Rekürsif özelliğin operatöre göre konumu ilişkilendirilme özelliğini belirler
  - Sol-rekürsif yapı; sol taraf ile ilişkilendirme
  - Sağ-rekürsif yapı; sağ taraf ile ilişkilendirme
  - yok: Rekürsif yapıyı kullanma. Operatörün her iki tarafında bir yüksek öncelik düzeyine ait non-terminal simgesine yer ver