

Bitti: BÖLÜM 1: FA (DFA,NFA), RE, PL

Buradayız: BÖLÜM 2: CFG, CFL, PDA

BÖLÜM 3: TM

Tekrar: Pumping Lemma ile RE İspat

- $\Sigma = \{0, 1\}$, $L = \{u0v : u, v \in \Sigma^* \text{ and } |u| = |v|\}$.
- Sözle ifade edersek; L orta sembolü 0 olan tek uzunluktaki katarlardır.
- L dilinin regular olup olmadığını PL yardımı ile gösteriniz.
- L diline ait $w=1^n01^n$ için PL uygularsak regular olmadığını kolayca gösterebiliriz.

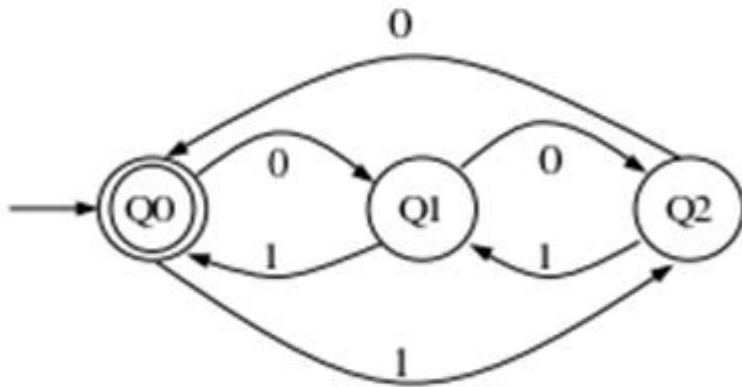
Tekrar: Pumping Lemma ile RE İspat

- Çelişki (contradiction) ile L 'nin düzenli dil olduğunu varsayalım. RE için PL'ye göre, L için bu lemmada ifade edilen özelliği karşılayan $n \geq 1$ bir pumping length bulunması gerekir.
 - $w \in L$ ve $|w| = 2n + 1 \geq n$ olmak üzere $w = 1^n 0 1^n$ alalım.
 - PL' 'den $w = xyz$ katarları için $x, y, z \in \Sigma^*$ olmak üzere
 - (i) $y \neq \varepsilon$, (ii) $|xy| \leq n$, ve (iii) $xy^i z \in L$ her $i \in \mathbb{N}$ için.
- $w = xyz = 1^n 0 1^n$ ve $|xy| \leq n$ olduğuna göre $y = 1^k$ $k \in \mathbb{N}$ olabilir çünkü önek xy tek 0 sembolünü içerirse (iii) sağlanmaz.
- Ayrıca, $y \neq \varepsilon$ olduğu için $k \geq 1$ olmalıdır. $i = 2$ için
$$xy^2 z = xyyz = 1^{n+k} 0 1^n \in L.$$
 - $k \geq 1$ için $1^{n+k} 0 1^n$ katarı L dilinin elemanı değildir; ya 0 ortada olmaz ya da uzunluğu çift olur. Bu yüzden L düzenli değildir.

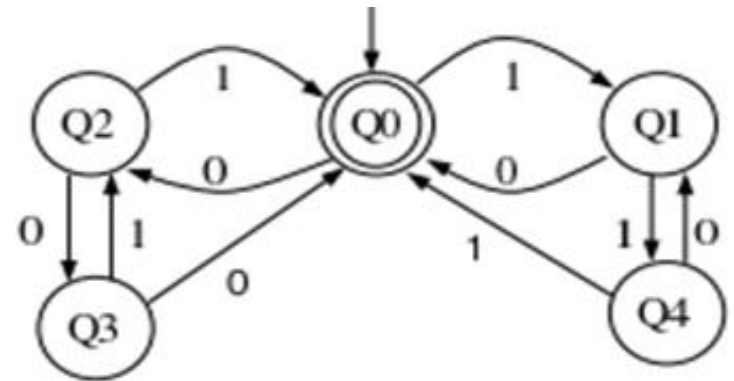
Tekrar: DFA ile RE İspat

- $d(w)$, $|n - m|$ değerini veriyor olsun. Burada n , w 'daki 0'ların sayısı ve m de 1'lerin sayısı olsun
- $A = \{w \in \{0, 1\}^* \mid d(w) = 0 \bmod 3\}$ düzenli midir? Cevabınızı kısaca anlatınız.

Tekrar: DFA ile RE İspat



Q0 - start and accept state (same number of 0s and 1s)
Q1 - $|x-y| \bmod 3$ is 1.
Q2 - $|x-y| \bmod 3$ is 2.
x - number of 1s
y - number of 0s



Q0 - start and accept state (same number of 0s and 1s)
Q1 - (number of 1s) mod 3 is 1.
Q4 - (number of 1s) mod 3 is 2.
Q2 - (number of 0s) mod 3 is 1.
Q3 - (number of 0s) mod 3 is 2.

CENG 306 Biçimsel Diller ve Otomatlar

Formal Languages and Automata

BÖLÜM 2

BAĞLAM DAN BAĞIMSIZ GRAMER
CONTEXT-FREE GRAMMAR

Konular

- Context-Free (bağlamdan bağımsız , içerikten bağımsız) Grammars (CFGs) and Languages (CFLs)
- Parse Trees (Türetme Ağacı) and Derivations

Context-Free Grammars and Languages

- Dil tanıyıcı cihaz bir dile ait geçerli string'leri kabul eder.
- Dil üretici cihaz bir dile ait string'leri oluşturur.
- Dil üretici cihazlar bir başlangıç işaretiyle bir string oluşturmaya başlarlar ve belirlenmiş kuralları kullanarak string oluştururlar.
- Regular expression bir dil üretici olarak kabul edilir.

Örnek:

$a(a^* \cup b^*)b$ regular expression ile önce bir a üretilir; ardından iki durumdan birisine göre devam edilir. istenen sayıda a üretilir veya b üretilir.

Son olarak da bir b üretilir.

Context-Free Grammars and Languages

- Context-free grammar ile regular ifadelerle göre çok daha karmaşık diller üretilebilir. Örnek:

- $a(a^* U b^*)b$ regular expression ön kısım, orta kısım ve son kısım olarak ayrıştırılabilir.

- S dilde bir string ve M ise orta kısım olmak üzere

☞ $S \rightarrow aMb$ şeklinde bir kural yazılabilir.

Burada M , a 'lardan veya b 'lerden oluşturulabilir.

$M \rightarrow A$ ve $M \rightarrow B$ yeni iki kural, A ve B dile ait stringlerdir.

$A \rightarrow e$, $A \rightarrow aA$ ve $B \rightarrow e$, $B \rightarrow bB$

- $a(a^* U b^*)b$ regular expression tarafından oluşturulan dil yukarıdaki kurallarla oluşturulabilir. **aaab** string'i aşağıdaki gibi oluşturulur;

$S \rightarrow aMb$, sonra $M \rightarrow A$ ile **aAb** $A \rightarrow aA$ ile **aaAb** yine $A \rightarrow aA$ ile **aaaAb** $A \rightarrow e$ ile **aaab** elde edilir.

Context-Free Grammars and Languages

- Kurallarla değiştirme işlemi, sadece değişecek sembol üzerinde yapılır ve önündeki ve ardındaki kısımlara bakılmaz.
- Bu yüzden context-free olarak adlandırılır.
- S, A, B, M non-terminal, a, b, e terminal olarak adlandırılır.
- Tüm string'ler sadece terminallerden oluşabilir.
- Kuralların uygulanması ve yeni bir string elde edilmesi regular expression'larda yeni bir string elde edilmesi gibidir.


Context-Free Grammars and Languages

Tanım:

- Bir context-free grammar $G = (V, \Sigma, R, S)$ şeklinde tanımlanır.

V alfabe (terminal ve non-terminaller)

Σ terminaller ($\Sigma \subseteq V$)

R kurallar ($(V - \Sigma) \times V^*$ )


S başlangıç sembolü ($S \in (V - \Sigma)$)

- $A \in V - \Sigma$ ve $u \in V^*$ için $(A, u) \in R$ ise $A \rightarrow_G u$ yazabiliriz.

- $u, v \in V^*$ için $u \Rightarrow_G v$ yazabiliriz sadece ve sadece

$x, y \in V^*$ ve $A \in V - \Sigma$ ve $u = xAy$ ve $v = xv'y$ ve $A \rightarrow_G v'$ ise

Context-Free Grammars and Languages

- \Rightarrow_G^* \Rightarrow_G ilişkisinin reflexive, transitive, closure 'udur.
- G grammar'ı tarafından oluşturulan dil
 $L(G) = \{w \in \Sigma^* : S \Rightarrow_G^* w\}$ şeklindedir. 
- Kurallar uygulanarak dile ait tüm string'ler elde edilebilir.
- Oluşturulan $L(G)$ dili **context-free language** olarak adlandırılır.
- $A \rightarrow_G u$ ve $u \Rightarrow_G v$ yerine $A \rightarrow u$ ve $u \Rightarrow v$ yazılabilir.
- $w_0 \Rightarrow_G w_1 \Rightarrow_G \dots \Rightarrow_G w_n$ **derivation** (türetme) olarak adlandırılır. w_n string'i w_0 'dan türetilmiştir.
- $w_0, \dots, w_n \in V^*$ olabilir. n **derivation length** (türetme uzunluğu) olarak adlandırılır.

Context-Free Grammars and Languages

Örnek:

$G = (V, \Sigma, R, S)$ grammar'i için

$V = \{S, a, b\}$, $\Sigma = \{a, b\}$ ve R kümesi $S \rightarrow aSb$ ve $S \rightarrow e$ olmak üzere iki tane kurala sahip olsun. Örnek bir derivation aşağıdaki gibi olabilir:

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb$$

ilk iki adımda $S \rightarrow aSb$ ve son olarak $S \rightarrow e$ uygulanmıştır. Oluşturulan dil

$L(G) = \{a^n b^n : n \geq 0\}$ olmuştur.

Bazı context-free diller regular değildir, ancak tüm regular diller context-free dildir.

Context-Free Grammars and Languages

Örnek:

$G = (W, \Sigma, R, S)$ grammar'i için $W = \{S, A, N, V, P\} \cup \Sigma$,

$\Sigma = \{Jim, big, green, cheese, ate\}$,

$R = \{P \rightarrow N,$

$P \rightarrow AP,$

$S \rightarrow PVP,$

$A \rightarrow big,$

$A \rightarrow green, N \rightarrow cheese, N \rightarrow Jim,$

$V \rightarrow ate\}$

$S = \text{sentence}$ $A = \text{adjective}$ $V = \text{verb}$

$N = \text{noun}$ $P = \text{phrase}$

$L(G)$ deki grammar olarak doğru bazı string'ler aşağıdadır;

Jim ate cheese

big Jim ate green cheese big cheese ate Jim

Bazı string'ler anlamsız olabilir;

big cheese ate green green big green big cheese green Jim ate green big Jim

Context-Free Grammars and Languages

- Herhangi bir bilgisayar programlama diliyle yazılmış bilgisayar programının, yazımının doğru olabilmesi için katı kurallara uyması gerekir.
- Konuşma dillerindeki tersine birçok programlama dilinin yazımının doğruluğunu kontrol etmek için context-free grammar'ler kullanılabilir.
- Özellikle programların syntax analizinde parse edilmesi aşamasında çok faydalıdır.
- Bir programlama dilinde bazı kısımlar regular expression'larla ifade edilebilir, program yapısı veya blok yapıları context-free grammar'ler tarafından ifade edilebilir.

Context-Free Grammars and Languages

Örnek: Bütün programlama dillerindeki ortak bir kısmı oluşturan bir dil tanımlayalım.

Bu dil doğru yazılmış aritmetik ifadeleri gösterebilir.

*id * (id * id + id)* yazımı doğru ancak ** id + (* ve *+ * id* yanlıştır.

(*id* değişken adlarıdır)

$$G = (V, \Sigma, R, E)$$

$$V = \{+, *, (,), id, T, F, E\},$$

$$\Sigma = \{+, *, (,), id\},$$

$$R = \{ E \rightarrow E + T, \quad (R1)$$

$$E \rightarrow T, \quad (R2)$$

$$T \rightarrow T * F, \quad (R3)$$

$$T \rightarrow F, \quad (R4)$$

$$F \rightarrow (E), \quad (R5)$$

$$F \rightarrow id \quad \} \quad (R6)$$

E = ifade (expression)

T = terim

F = öge (factor)

Context-Free Grammars and Languages

Örnek: (devam)

G grammar'ini $(id * id + id) * (id + id)$ string'ini aşağıdaki gibi oluşturur;

$$E \Rightarrow T \text{ (R2)}$$

$$\Rightarrow T * F \text{ (R3)}$$

$$\Rightarrow T * (E) \text{ (R5)}$$

$$\Rightarrow T * (E + T) \text{ (R1)}$$

$$\Rightarrow T * (T + T) \text{ (R2)}$$

$$\Rightarrow T * (F + T) \text{ (R4)}$$

$$\Rightarrow T * (id + T) \text{ (R6)}$$

$$\Rightarrow T * (id + F) \text{ (R4)}$$

$$\Rightarrow T * (id + id) \text{ (R6)}$$

$$\Rightarrow F * (id + id) \text{ (R4)}$$

$$\Rightarrow (E) * (id + id) \text{ (R5)}$$

$$\Rightarrow (E + T) * (id + id) \text{ (R1)}$$

$$\Rightarrow (E + F) * (id + id) \text{ (R4)}$$

$$\Rightarrow (E + id) * (id + id) \text{ (R6)}$$

$$\Rightarrow (T + id) * (id + id) \text{ (R2)}$$

$$\Rightarrow (T * F + id) * (id + id) \text{ (R3)}$$

$$\Rightarrow (F * F + id) * (id + id) \text{ (R4)}$$

$$\Rightarrow (F * id + id) * (id + id) \text{ (R6)}$$

$$\Rightarrow (id * id + id) * (id + id) \text{ (R6)}$$

Context-Free Grammars and Languages

Örnek: Düzgün dağılımlı sağ ve sol parantezleri üreten bir grammar oluşturalım.

$$G = (V, \Sigma, R, S)$$

$$V = \{S, (,)\},$$

$$\Sigma = \{ (,) \},$$

$$R = \{ S \rightarrow e, S \rightarrow SS, S \rightarrow (S) \}$$

Aşağıdaki türetmeleri G grammar'i oluşturur;

$$S \Rightarrow SS \Rightarrow S(S) \Rightarrow S((S)) \Rightarrow S(()) \Rightarrow \cancel{())}(())$$



$$S \Rightarrow SS \Rightarrow (S)S \Rightarrow ()S \Rightarrow ()(S) \Rightarrow \cancel{()}(())$$

Context-Free Grammars and Languages

Örnek: Bir $M = (K, \Sigma, \delta, s, F)$ DFA tarafından tanınan regular dil $L(M)$,

$G(M) = (V, \Sigma, R, S)$ grammar'i tarafından oluşturulabilir. Burada,

$$V = K \cup \Sigma,$$

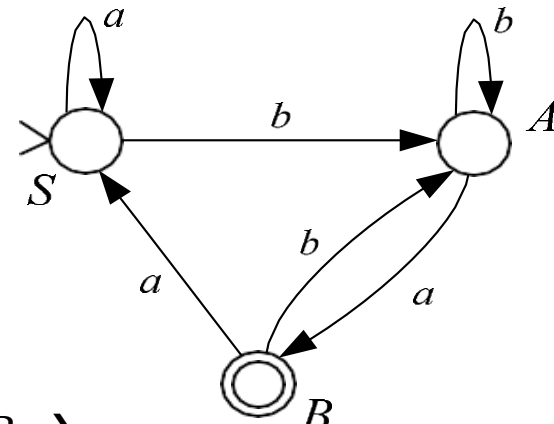
$$S = s,$$

$$R = \{q \rightarrow ap : \delta(q, a) = p\} \cup \{q \rightarrow e : q \in F\}$$

Nonterminaller otomatın durumları ve a girişi için yapılan q dan p 'ye geçiş R içinde $q \rightarrow ap$ şeklinde bir kural olarak alınır.

Yandaki otomat için oluşturulan kurallar;

$$S \rightarrow aS, S \rightarrow bA, A \rightarrow bA, A \rightarrow aB, B \rightarrow aS, B \rightarrow bA, B \rightarrow e.$$



Context-Free Grammars and Languages

- Regular olmayan context-free diller vardır.
- Ancak bütün regular diller context-free dildir.
- Regular olan ϕ ve $\{a\}$ basit context-free dillerdir. Kuralı olmayan ve sadece $S \rightarrow a$ şeklinde kuralı olan dillerdir.
- Context-free diller **union, concatenation ve Kleene star** işlemleri için kapalıdır.
- Context-free diller pushdown otomatlar tarafından tanınır. Pushdown otomatlar finite otomatların genelleştirilmiş şeklidir.
- Her finite otomat basit yapıdaki pushdown otomat olarak düşünülür. (Her PDA bir FA gibi çalışabilir ama tersi olmaz)

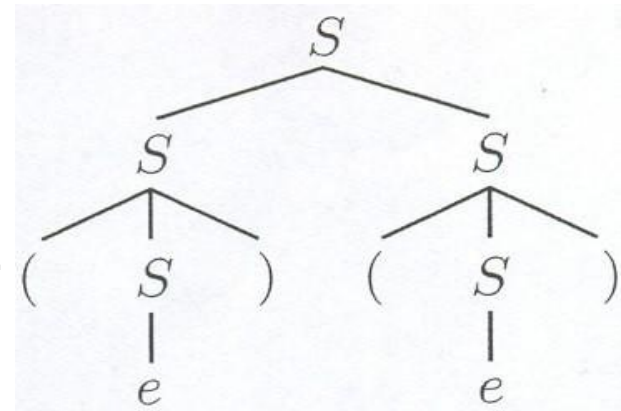
Parse Trees and Derivations

- G context-free grammar olsun. Bu grammar ile bir string'in oluşturulması farklı şekillerde olabilir.
- Dengeli parantez üreten bir context-free dil aşağıdaki farklı şekillerde aynı string'i üretir.

$$S \Rightarrow SS \Rightarrow S(S) \Rightarrow (S)(S) \Rightarrow (S)() \Rightarrow ()()$$

$$S \Rightarrow SS \Rightarrow (S)S \Rightarrow ()S \Rightarrow ()(S) \Rightarrow ()()$$

- iki türetmede yandaki şekildeki gibi gösterilebilir.



Bu şekil parse tree olarak adlandırılır.

- Her node V içinde bir nonterminal semboldür.
- Yapraklar (leaves) terminallerdir ve Σ içinde bir semboldür.
- Yapraklar soldan sağa concatenate edildiğinde oluşan string üretilir.

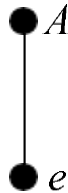
Parse Trees and Derivations

Bir context-free grammar $G = (V, \Sigma, R, S)$ için parsetree, roots, leaves aşağıdaki gibi tanımlanır.

1. $\circ a$

Her $a \in \Sigma$ için bu bir parse tree'dir. Tek node hem root hem de yapraktır ve a oluşturur.

2.

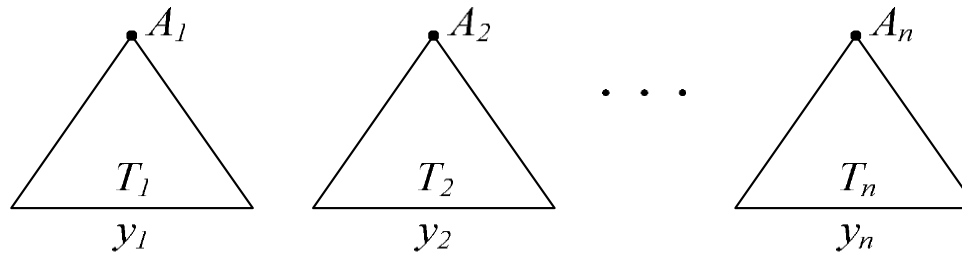


$A \rightarrow e$, R içinde bir kural ise bu bir parse tree'dir.

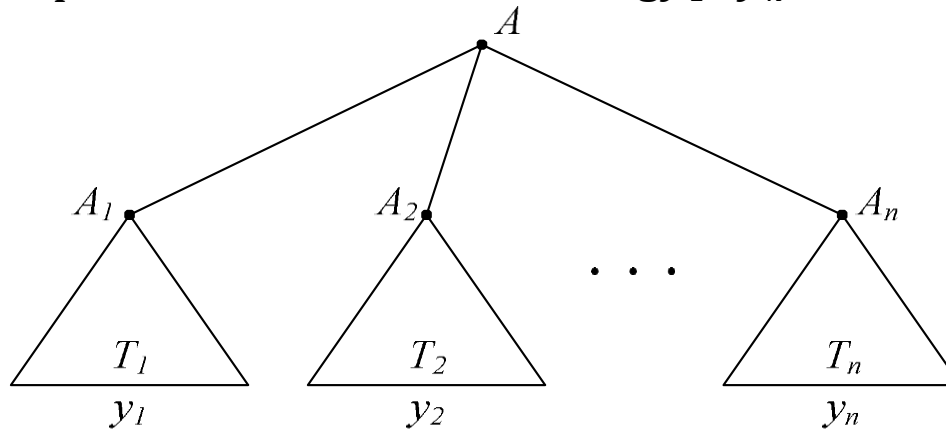
A root ve e yapraktır. Sadece e üretir.

Parse Trees and Derivations

3.



Hepsi parse tree'dir. $n \geq 1$ için A_1, \dots, A_n root ve y_1, \dots, y_n üretilir. $A \rightarrow A_1, \dots, A_n$ R içinde kural ise aşağıdaki parse tree'dir. Üretilen string $y_1 \dots y_n$ olur.



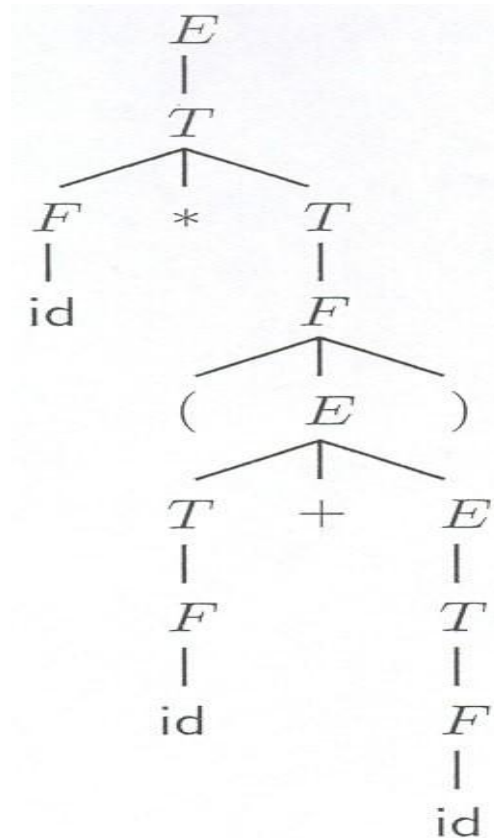
4. Bunların dışında hiçbir şey parse tree değildir.

Parse Trees and Derivations

Örnek: Aritmetik ifadeleri oluşturan grammar'ın

$\text{id} * (\text{id} + \text{id})$

için oluşturduğu parse tree aşağıdaki gibidir.



Parse Trees and Derivations

Bir context-free grammar $G = (V, \Sigma, R, S)$ için

$D = x_1 \Rightarrow x_2 \Rightarrow \dots \Rightarrow x_n$ ve $D' = x'_1 \Rightarrow x'_2 \Rightarrow \dots \Rightarrow x'_n$ *iki farklı türetmedir.*

Burada $x_i, x'_i \in V^$ ve $x_1, x'_1 \in V - \Sigma$ ve $x_n, x'_n \in \Sigma^*$*

iki türetmede bir nonterminalden terminal string'leri türetilir.

D türetmesi D' türetmesinden öncedir ve $D < D'$ şeklinde gösterilir **eğer;**

$n > 2$ için $1 < k < n$ olacak şekilde bir k değeri varsa ve;

1. Tüm $i \neq k$ için $x_i = x'_i$

2. $x_{k-1} = x'_{k-1} = uAvBw$ burada $u, v, w \in V^*$ ve $A, B \in V - \Sigma$

3. $x_k = uyvBw$, burada $A \rightarrow y \in R$

4. $x'_k = uAvzw$, burada $B \rightarrow z \in R$

5. $x_{k+1} = x'_{k+1} = uyvzw$

*En soldaki nonterminali önce değiştiren türetme diğerinden **önce gelir.***

Parse Trees and Derivations

Örnek: Herhangi bir grammar için aşağıdaki üç türetmenin önceliklerini çıkaralım.

$$D_1 = S \Rightarrow SS \Rightarrow (S)S \Rightarrow ((S))S \Rightarrow \textcolor{red}{(())}S \Rightarrow (())(S) \Rightarrow (())()$$



$$D_2 = S \Rightarrow SS \Rightarrow (S)S \Rightarrow ((S))S \Rightarrow \textcolor{red}{((S))}(\textcolor{red}{S}) \Rightarrow \textcolor{blue}{(())}(\textcolor{blue}{S}) \Rightarrow (())()$$

$$D_3 = S \Rightarrow SS \Rightarrow (S)S \Rightarrow ((S))S \Rightarrow ((S))(S) \Rightarrow \textcolor{blue}{((S))}(\textcolor{blue}{()}) \Rightarrow (())()$$



Burada $D_1 < D_2$ ve $D_2 < D_3$ olur.

$D_1 < D_3$ olamaz çünkü birden fazla ara string'te farklılık vardır. Bütün türetmeler aynı parse tree'ye sahiptir.

Parse Trees and Derivations

örnek: devam

$$D_1 = S \Rightarrow SS \Rightarrow (S)S \Rightarrow ((S))S \Rightarrow (())S \Rightarrow (())(S) \Rightarrow (())()$$

$$D_2 = S \Rightarrow SS \Rightarrow (S)S \Rightarrow ((S))S \Rightarrow ((S))(S) \Rightarrow (())(S) \Rightarrow (())()$$

$$D_3 = S \Rightarrow SS \Rightarrow (S)S \Rightarrow ((S))S \Rightarrow ((S))(S) \Rightarrow ((S))() \Rightarrow (())()$$

$$D_4 = S \Rightarrow SS \Rightarrow (S)S \Rightarrow (S)(S) \Rightarrow ((S))(S) \Rightarrow (())(S) \Rightarrow (())()$$

$$D_5 = S \Rightarrow SS \Rightarrow (S)S \Rightarrow (S)(S) \Rightarrow ((S))(S) \Rightarrow ((S))() \Rightarrow (())()$$

$$D_6 = S \Rightarrow SS \Rightarrow (S)S \Rightarrow (S)(S) \Rightarrow ((S))() \Rightarrow ((S))() \Rightarrow (())()$$

$$D_7 = S \Rightarrow SS \Rightarrow S(S) \Rightarrow (S)(S) \Rightarrow ((S))(S) \Rightarrow (())(S) \Rightarrow (())()$$

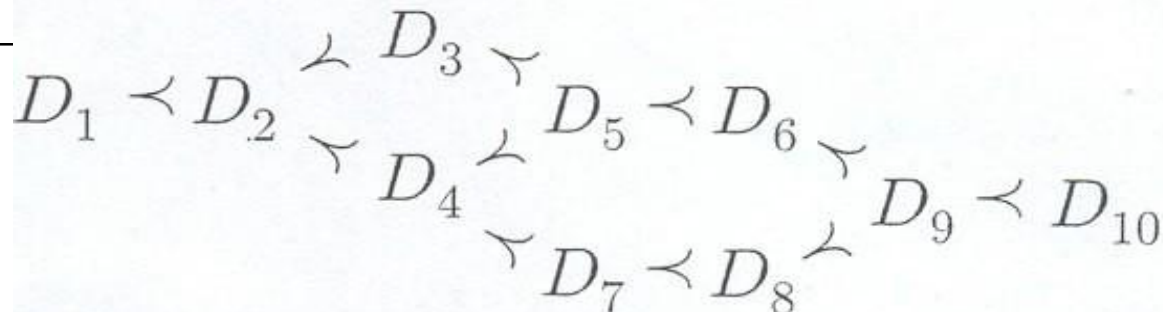
$$D_8 = S \Rightarrow SS \Rightarrow S(S) \Rightarrow (S)(S) \Rightarrow ((S))(S) \Rightarrow ((S))() \Rightarrow (())()$$

$$D_9 = S \Rightarrow SS \Rightarrow S(S) \Rightarrow (S)(S) \Rightarrow ((S))() \Rightarrow ((S))() \Rightarrow (())()$$

$$D_{10} = S \Rightarrow SS \Rightarrow S(S) \Rightarrow S() \Rightarrow (S)() \Rightarrow ((S))() \Rightarrow (())()$$



Buradaki tüm öncelik ilişkileri aşağıdaki şekilde gösterilebilir.



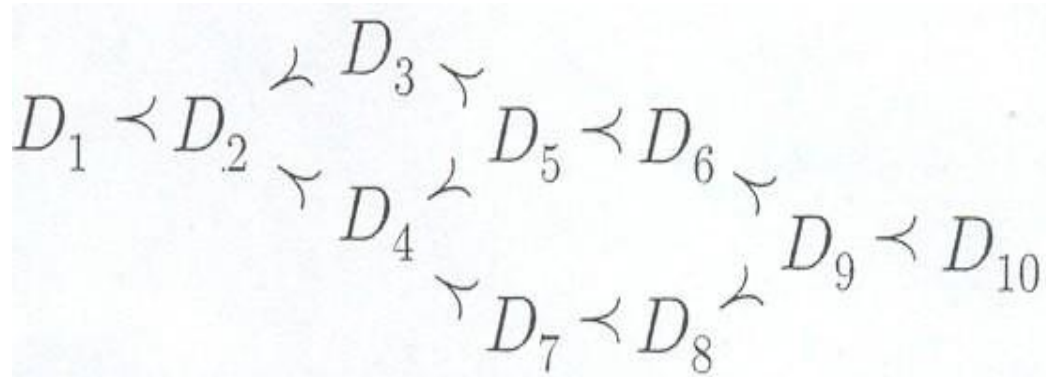
Parse Trees and Derivations

Bir parse tree üzerinde **leftmost derivation** ve **rightmost derivation** elde edilebilir.

Leftmost derivation için ağacın root node'undan başlanır ve **sürekli en soldaki nonterminal değiştirilir**.

Rightmost derivation için ağacın root node'undan başlanır ve **sürekli en sağdaki nonterminal değiştirilir**.

Önceki örnekte D_1 **leftmost** D_{10} ise **rightmost** derivation n türetmedir.



Parse Trees and Derivations

Leftmost derivation için $x \Rightarrow^L y$ ve rightmost derivation için $x \Rightarrow^R y$ kullanılır.

$x \Rightarrow^L y$ yazabiliriz eğer sadece ve sadece $x = wA\beta$, $y = w\alpha\beta$, ise ve burada $w \in \Sigma^*$ ve $\alpha, \beta \in V^*$ ve $A \in V - \Sigma$ ve $A \rightarrow \alpha$ kuralı grammar'de varsa.

$x_1 \Rightarrow^L x_2 \Rightarrow^L \dots \Rightarrow^L x_n$ tüm leftmost türetme sırasını ifade eder.

SORU

- $L1 = \{a^n b^m c d^m e^n \mid n, m \geq 0\}$
- $L2 = \{(ab)^n c d^m \mid n, m \geq 0\}$
- $L3 = \{a^n b(cd)^n e^n \mid n \geq 0\}$

One of the languages is regular, one context-free and not regular and one not context-free. Which are the regular and the non-regular context-free languages?

SORU

• $L1 = \{a^n b^m c d^m f^n \mid n, m \geq 0\}$

$G = (V, \Sigma, R, S)$ $V = \{S, A, B, C, a, b, c, d, f\}$ $\Sigma = \{a, b, c, d, f\}$
 $S = \{S\}$

$R = \{ \underline{S \rightarrow aSf} \mid \overset{A}{\cancel{aAf}}, \underline{A \rightarrow bAd} \mid \overset{C}{\cancel{bCd}}, \underline{C \rightarrow c} \mid \cancel{e} \}$

Öyle ise $L1$ context-free

SORU

- $L2 = \{(ab)^n cd^m \mid n, m \geq 0\}$
- $L2a = (ab)^n$ regular
- $L2b = c$ regular
- $L2c = d^m$ regular
- Öyle ise $L2 = L2a.L2b.L2c$ regular (closure under concatenation)

SORU

- $L3 = \{a^n b(cd)^n e^n \mid n \geq 0\}$
- $L3$ ne RL ne de CFL

Ödev

- Problemleri çözünüz 3.1.1b, 3.1.1c, 3.1.1c (sayfa 120)
- Problemleri çözünüz 3.1.2, 3.1.3 (sayfa 120)
- Problemleri çözünüz. 3.1.8 (sayfa 121)
- Problemleri çözünüz 3.2.4 (sayfa 129)