BLG311 Biçimsel Diller ve Otomatlar

Yığın Tabanlı Otomatlar(Pushdown Automata(PDA)) ve Bağlamdan Bağımsız Dillerin Tanınması

A.Emre Harmancı Tolga Ovatman Berk Canberk

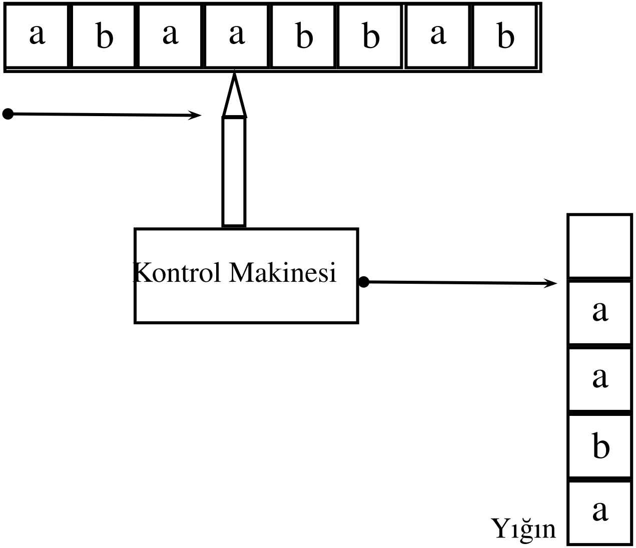
2017

İçerik

Bağlamdan bağımsız tüm dilleri tanıyan sonlu otomat mevcut değildir. Örneğin $\omega\omega^R|\omega \in \Sigma^*$ dilini tanıyabilmek için ek belleğe ihtiyaç duyulur. Bağlamdan bağımsız her dil yığın yapılı otomat tarafından tanınır.

Yığın Tabanlı Otomatlar

Bir yığın tabanlı otomatta, sonlu otomatlardan farklı olarak yığın kurallarıyla çalışan bir bellek mevcuttur. PDA'lar genellikle non-determinist tanımlanmaktadır ve bu tür geçişler sonlu otomatlardan farklı olarak her zaman elimine edilemez.



PDA'lar non-deterministtir. Giriş sekansı sadece okunabilirken, yığın üzerinde okuma ve yazma işlemleri uygulanabilir.

Bir PDA'nın biçimsel tanımı

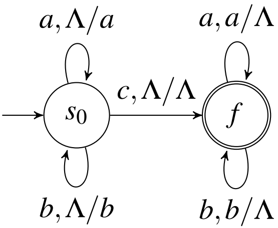
Bir PDA 6'lı kullanılarak tanımlanabilir $M = (S, \Sigma, \Gamma, \delta, s_0, F)$:

- S: Sonlu ve boş olmayan bir durum kümesi $s \in S$.
- Σ: Giriş alfabesi (sonlu ve boş olmayan bir sembol kümesi)
- Γ: Yığın alfabesi
- $s_0 \in S$: S içerisinde bir başlangıç durumu.
- δ: Durum geçiş bağıntısı $\delta \subseteq (S \times \Sigma \cup \{\Lambda\} \times \Gamma \cup \{\Lambda\}) \times (S \times \Gamma^*)$
- F: Son durumlar kümesi $F \subseteq S$.

Örnek

$$(\omega c \omega^R | \omega \in \{a,b\}^*)$$
$$M = (S, \Sigma, \Gamma, \delta, s_0, F)$$
$$S = \{s_0, f\},\ \Sigma = \{a,b,c\},\ \Gamma = \{a,b\},\ F = \{f\}$$
$$\delta = \{[(s_0,a,\Lambda),(s_0,a)],[(s_0,b,\Lambda),(s_0,b)],[(s_0,c,\Lambda),(f,\Lambda)],$$
$$[(f,a,a),(f,\Lambda)],[(f,b,b),(f,\Lambda)]\}$$

durum	girdi	yığın	geçiş
s_0	abb **c** bba	Λ	[(s_0,a,Λ),(s_0,a)]
s_0	bb **c** bba	a	[(s_0,b,Λ),(s_0,b)]
s_0	b **c** bba	ba	[(s_0,b,Λ),(s_0,b)]
s_0	**c** bba	bba	[(s_0,c,Λ),(f,Λ)]
f	bba	bba	[(f,b,b),(f,Λ)]
f	ba	ba	[(f,b,b),(f,Λ)]
f	a	a	[(f,a,a),(f,Λ)]
f	Λ	Λ	



Örnek

durum	girdi	yığın	geçiş
s	abb **c** bba	Λ	[(s,a,Λ),(s,a)]
s	bb **c** bba	a	[(s,b,Λ),(s,b)]
s	b **c** bba	ba	[(s,b,Λ),(s,b)]
s	**c** bba	bba	[(s,c,Λ),(f,Λ)]
f	bba	bba	[(f,b,b),(f,Λ)]
f	ba	ba	[(f,b,b),(f,Λ)]
f	a	a	[(f,a,a),(f,Λ)]
f	Λ	Λ	

$$G = (N, \Sigma, n_0, \mapsto)$$
$$N = \{S\}$$
$$\Sigma = \{a, b, c\}$$
$$n_0 = S$$
$$< S >::= a < S > a \mid b < S > b | c$$

Tanımlar

Ekleme: Yığına bir sembol ekleme(push) $[(p,u,\Lambda),(q,a)]$

Çıkarma: Yığından bir sembol çıkarma(pop) $[(p,u,a),(q,\Lambda)]$

Konfigürasyon: $S \times \Sigma^* \times \Gamma^*$ üzerine tanımlanır. Örneğin (q,xyz,abc) konfigürasyonunda a yığın tepesi, c yığın dibidir. Anlık tanım (bir

adımda üretme):
Örneğin $[(p,u,\beta),(q,\gamma)] \in \delta$ ve $\forall x \in \Sigma^* \wedge \forall \alpha \in \Gamma^*$

$(p,ux,\beta\alpha) \vdash_M (q,x,\gamma\alpha)$
Bu durumda u giriş olarak okunur ve β yığından okunurken γ yığına yazılmaktadır.

Tanımlar

$(p, ux, \beta\alpha) \vdash_M (q, x, \gamma\alpha)$

Örneğin ${\vdash_M}^*$, $\vdash_M$'in yansımalı geçişli kapanışı olsun ve $\omega \in \Sigma^*$ ile s_0 ilk durumlar olsun. M otomatının ω katarını kabul etmesi için:

$(s, \omega, \Lambda){\vdash_M}^*(p, \Lambda, \Lambda)$ and $p \in F$
$C_0 = (s, \omega, \Lambda)$ ve $C_n = (p, k, \Lambda)$ ise $C_0 \vdash_M C_1 \vdash_M \ldots \vdash_M C_{n-1} \vdash_M C_n$

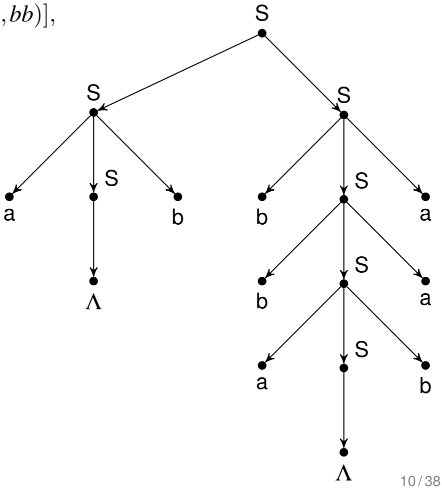
Bu işleme M otomatının yaptığı *hesaplama* denir; bu hesapta n adım bulunur.

$L(M)$ dili M tarafından tanınan katarlar kümesi olsun.
$L(M) = \{\omega | (s, \omega, \Lambda){\vdash_M}^*(p, \Lambda, \Lambda) \wedge p \in F\}$

Örnek 1

$\omega \in \{\{a,b\}^*|\#(a) = \#(b)\}$
$M = (S,\Sigma,\Gamma,\delta,s_0,F)$
$\delta = \{[(s,\Lambda,\Lambda),(q,c)],[(q,a,c),(q,ac)],[(q,a,a),(q,aa)],$
$[(q,a,b),(q,\Lambda)],[(q,b,c),(q,bc)],[(q,b,b),(q,bb)],$
$[(q,b,a),(q,\Lambda)],[(q,\Lambda,c),(f,\Lambda)]\}$

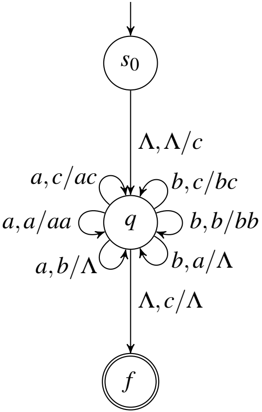
durum	girdi	yığın	geçiş
s	abbbabaa	Λ	[(s,Λ,Λ),(q,c)]
q	abbbabaa	c	[(q,a,c),(q,ac)]
q	bbbabaa	ac	[(q,b,a),(q,Λ)]
q	bbabaa	c	[(q,b,c),(q,bc)]
q	babaa	bc	[(q,b,b),(q,bb)]
q	abaa	bbc	[(q,a,b),(q,Λ)]
q	baa	bc	[(q,b,b),(q,bb)]
q	aa	bbc	[(q,a,b),(q,Λ)]
q	a	bc	[(q,a,b),(q,Λ)]
q	Λ	c	[(q,Λ,c),(f,Λ)]
f	Λ	Λ	



Örnek 1

$\omega \in \{\{a,b\}^* | \#(a) = \#(b)\}$
$M = (S, \Sigma, \Gamma, \delta, s_0, F)$
$\delta = \{[(s,\Lambda,\Lambda),(q,c)],[(q,a,c),(q,ac)],[(q,a,a),(q,aa)],$
$[(q,a,b),(q,\Lambda)],[(q,b,c),(q,bc)],[(q,b,b),(q,bb)],$
$[(q,b,a),(q,\Lambda)],[(q,\Lambda,c),(f,\Lambda)]\}$

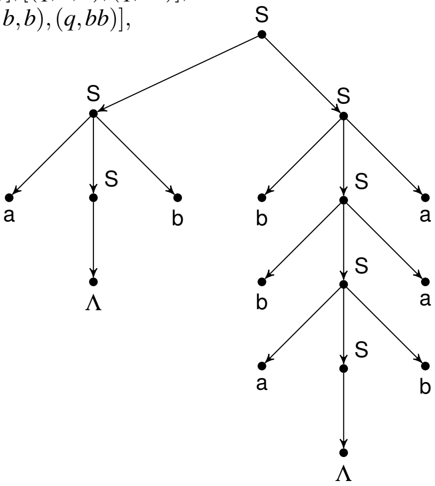
durum	girdi	yığın	geçiş
s	abbbabaa	Λ	[(s,Λ,Λ),(q,c)]
q	abbbabaa	c	[(q,a,c),(q,ac)]
q	bbbabaa	ac	[(q,b,a),(q,Λ)]
q	bbabaa	c	[(q,b,c),(q,bc)]
q	babaa	bc	[(q,b,b),(q,bb)]
q	abaa	bbc	[(q,a,b),(q,Λ)]
q	baa	bc	[(q,b,b),(q,bb)]
q	aa	bbc	[(q,a,b),(q,Λ)]
q	a	bc	[(q,a,b),(q,Λ)]
q	Λ	c	[(q,Λ,c),(f,Λ)]
f	Λ	Λ	



Örnek 1

$\omega \in \{\{a,b\}^*|\#(a)=\#(b)\}$
$M=(S,\Sigma,\Gamma,\delta,s_0,F)$
$\delta=\{[(s,\Lambda,\Lambda),(q,c)],[(q,a,c),(q,ac)],[(q,a,a),(q,aa)],$
$[(q,a,b),(q,\Lambda)],[(q,b,c),(q,bc)],[(q,b,b),(q,bb)],$
$[(q,b,a),(q,\Lambda)],[(q,\Lambda,c),(f,\Lambda)]\}$

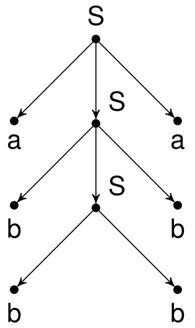
$$G=(N,\Sigma,n_0,\mapsto)$$
$$N=\{S\}$$
$$\Sigma=\{a,b\}$$
$$n_0=S$$
$$<S>::=a<S>b$$
$$::=b<S>a$$
$$::=<S><S>$$
$$::=\Lambda$$



Örnek 2

$\omega \in \{xx^R | x \in \{a,b\}^*\}$
$M = (S, \Sigma, \Gamma, \delta, s_0, F)$
$\delta =$
$\{[(s,a,\Lambda),(s,a)],[(s,b,\Lambda),(s,b)],[(s,\Lambda,\Lambda),(f,\Lambda)],[(f,a,a),(f,\Lambda)],[(f,b,b),(f,\Lambda)]\}$

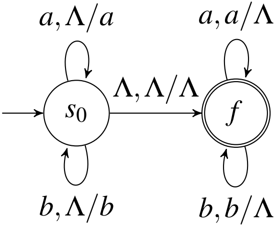
durum	girdi	yığın	geçiş
s	abbbba	Λ	[(s,a,Λ),(s,a)]
s	bbbba	a	[(s,b,Λ),(s,b)]
s	bbba	ba	[(s,b,Λ),(s,b)]
s	bba	bba	[(s,Λ,Λ),(f,Λ)]
f	bba	bba	[(f,b,b),(f,Λ)]
f	ba	ba	[(f,b,b),(f,Λ)]
f	a	a	[(f,a,a),(f,Λ)]
f	Λ	Λ	



Örnek 2

$\omega \in \{xx^R | x \in \{a,b\}^*\}$
$M = (S, \Sigma, \Gamma, \delta, s_0, F)$
$\delta =$
$\{[(s,a,\Lambda),(s,a)],[(s,b,\Lambda),(s,b)],[(s,\Lambda,\Lambda),(f,\Lambda)],[(f,a,a),(f,\Lambda)],[(f,b,b),(f,\Lambda)]\}$

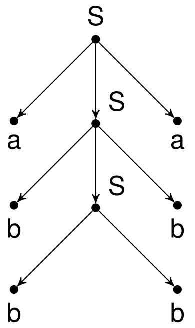
durum	girdi	yığın	geçiş
s	abbbba	Λ	[(s,a,Λ),(s,a)]
s	bbbba	a	[(s,b,Λ),(s,b)]
s	bbba	ba	[(s,b,Λ),(s,b)]
s	bba	bba	[(s,Λ,Λ),(f,Λ)]
f	bba	bba	[(f,b,b),(f,Λ)]
f	ba	ba	[(f,b,b),(f,Λ)]
f	a	a	[(f,a,a),(f,Λ)]
f	Λ	Λ	



Örnek 2

$\omega \in \{xx^R | x \in \{a,b\}^*\}$
$M = (S, \Sigma, \Gamma, \delta, s_0, F)$
$\delta =$
$\{[(s,a,\Lambda),(s,a)],[(s,b,\Lambda),(s,b)],[(s,\Lambda,\Lambda),(f,\Lambda)],[(f,a,a),(f,\Lambda)],[(f,b,b),(f,\Lambda)]\}$

$G = (N, \Sigma, n_0, \mapsto)$
$N = \{S\}$
$\Sigma = \{a,b\}$
$< S > ::= a < S > a \mid b < S > b \mid aa \mid bb$



Determinist PDA

Determinist PDA

1) $\forall s \in S \wedge \forall \gamma \in \Gamma$ if $\delta(s, \Lambda, \gamma) \neq \varnothing \Rightarrow \delta(s, \sigma, \gamma) = \varnothing; \forall \sigma \in \Sigma$
2) Eğer $a \in \Sigma \cup \{\Lambda\}$ ise $\forall s, \forall \gamma$ ve $\forall a$ Card$(\delta(s, a, \gamma)) \leq 1$

- Her hangi bir konfigürasyondan tek adımda üretilebilen konfigürasyon sayısı tek ise, giriş "Λ" olduğunda (k, Λ, γ)'dan tek adımda tek bir üretim yapılabiliyorsa, aynı koşullarda giriş başka bir değer aldığında üretim olmamaktadır, ve $\delta(k, a, \gamma) = \varnothing$ olur.
- Sonlu otomatlarda, kabul edilen diller açısından determinist ve determinist olmayan otomatlar arasında eşdeğerlilik bulunabilmekteydi, ancak PDA'lar için bu geçerli değildir.
- Örneğin $\omega\omega^R$ determinist olmayan bir PDA tarafından kabul edildiği halde, bu dili kabul eden bir determinist PDA yoktur.

Chomsky Hiyerarşisi

Tipi	Dil (Gramer)	Gramerdeki türetim kuralları	Tanıyan Otomat
0	Recursively enumerable (kısıtsız)	$\alpha \to \beta$ $(\alpha, \beta \in (N \cup \Sigma)^*$, α değişken içerir$)$	Turing makinesi
1	Bağlama duyarlı	$\alpha \to \beta$ $(\alpha, \beta \in (N \cup \Sigma)^*, \lvert\beta\rvert \geq \lvert\alpha\rvert$, α değişken içerir$)$	doğrusal sınırlı otomat
2	Bağlamdan bağımsız	$A \to \alpha$ $(A \in N, \alpha \in (N \cup \Sigma)^*)$	yığın yapılı otomat
3	Düzenli	$A \to aB$ $(A, B \in N, a \in \Sigma)$	Sonlu otomat

- Bağlamdan bağımsız gramerler tarafından üretilebilecek katarlar hakkında fikir yürütmek için üretim kurallarını belirli kalıplarda ele almak işimizi kolaylaştırabilir.
- Belirli kalıplara sahip gramerlerde bir takım üretimlerin bulunmayacağını bildiğimizden, üretim işlemini daha basit bir biçimde ele alabiliriz.
- Örneğin bir x katarının bir G grameri tarafından üretilebildiğini anlamak için n adımlı tüm türetimleri incelemek isteriz. Sadece belirli basit üretim kuralları içeren bir gramere sahipsek bu inceleme kolaylaşır.

Tanım

Bağlamdan bağımsız bir gramer yalnızca aşağıdaki tip üretim kuralları içeriyorsa *Chomsky normal forma*(CNF) uygundur denir:

- $A \rightarrow BC$ (B ve C non-terminal)
- $A \rightarrow \sigma$ (σ terminal)

Teorem

Her bağlamdan bağımsız G grameri için, CNFe uygun başka bir G_1 grameri $L(G_1) = L(G) - \Lambda$ olacak şekilde bulunmaktadır.

CNF Dönüşümü

Aşağıdaki algoritma ile bir Tip-2 G gramerinden CNFe uygun bir G_1 gramerine dönüşüm yapılabilir:

1. Önce bütün birim üretimleri, ardından bütün Λ üretimlerini eleyin.
2. Sağında iki ve fazla sayıda terminal içeren katarları parçalayın.
3. Sağında üç ve fazla sayıda non-terminal içeren kuralları eşdeğer iki non-terminal kurallarla değiştirin.

Örnek CNF Dönüşümü

Tip-2 Gramer

$S = a, b, c$
$N = S, T, U, V, W$
$\mapsto = \{$
S $\rightarrow$ TU | V
T $\rightarrow$ aTb | Λ
U $\rightarrow$ cU | Λ
V $\rightarrow$ aVc | W
W $\rightarrow$ bW | Λ
$\}$
Bu gramer $a^i b^j c^k | i = j$ or $i = k$ dilini üretebilir.

CNF Dönüşümü

1. Önce bütün birim üretimleri, ardından bütün Λ üretimlerini eleyin.
   Birim üretime ör.: $V \rightarrow W$
   Λ üretime ör.: $T \rightarrow \Lambda$

Tip-2 Gramer

S $\rightarrow$ TU | V
T $\rightarrow$ aTb | Λ
U $\rightarrow$ cU | Λ
V $\rightarrow$ aVc | W
W $\rightarrow$ bW | Λ

Birim üretimler elendi

S $\rightarrow$ TU | aVc | bW | Λ
T $\rightarrow$ aTb | Λ
U $\rightarrow$ cU | Λ
V $\rightarrow$ aVc | bW | Λ
W $\rightarrow$ bW | Λ

CNF Dönüşümü

1. Önce bütün birim üretimleri, ardından bütün Λ üretimlerini eleyin.
   Birim üretime ör.: $V \rightarrow W$
   Λ üretime ör.: $T \rightarrow \Lambda$

Tip-2 Gramer

S $\rightarrow$ TU | aVc | bW
T $\rightarrow$ aTb | Λ
U $\rightarrow$ cU | Λ
V $\rightarrow$ aVc | bW | Λ
W $\rightarrow$ bW | Λ

Λ üretimler elendi

S $\rightarrow$ TU | aVc | bW | aTb | ab | cU | c | b | ac
T $\rightarrow$ aTb | ab
U $\rightarrow$ cU | c
V $\rightarrow$ aVc | bW | b | ac
W $\rightarrow$ bW | b

CNF Dönüşümü

2. Sağında iki ve fazla sayıda terminal içeren katarları parçalayın. Tüm σ terminalleri için X_σ biçiminde bir non-terminal ve $X_\sigma \to \sigma$ biçiminde bir üretim kuralı ekleyin.

Tip-2 Gramer

S → TU | aTb | aVc | cU
S → ab | ac | c | b | bW
T → aTb | ab
U → cU | c
V → aVc | ac | bW | b
W → bW | b

Çoklu terminaller el.

S → TU | X_aTX_b | X_aVX_c | X_cU
S → X_a X_b | X_a X_c | c | b | X_bW
T → X_aTX_b | X_a X_b
U → X_cU | c
V → X_aVX_c | X_a X_c | X_bW | b
W → X_bW | b
X_a → a
X_b → b
X_c → c

CNF Dönüşümü

3. Sağında üç ve fazla sayıda non-terminal içeren kuralları eşdeğer iki non-terminal kurallarla değiştirin.

Tip-2 Gramer

S $\rightarrow$ TU | X_aTX_b | X_cU | X_aVX_c
S $\rightarrow$ X_a X_b | X_a X_c | c | b | X_bW
T $\rightarrow$ X_aTX_b | X_a X_b
U $\rightarrow$ X_cU | c
V $\rightarrow$ X_aVX_c | X_a X_c | X_bW | b
W $\rightarrow$ X_bW | b
X_a $\rightarrow$ a
X_b $\rightarrow$ b
X_c $\rightarrow$ c

Çoklu non-termi. el.

S $\rightarrow$ TU | X_a Y_1 | X_cU | X_a Y_2
Y_1 $\rightarrow$ TX_b
Y_2 $\rightarrow$ VX_c
S $\rightarrow$ X_a X_b | X_a X_c | X_bW | b | c
T $\rightarrow$ X_a Y_1 | X_a X_b
U $\rightarrow$ X_cU | c
V $\rightarrow$ X_a Y_2 | X_a X_c | X_bW | b
W $\rightarrow$ X_bW | b
X_a $\rightarrow$ a
X_b $\rightarrow$ b
X_c $\rightarrow$ c

Başka bir örnek

1. Birim üretimleri eleyin.

Tip-2 Gramer

S $\rightarrow$ AaA | bA | BaB
A $\rightarrow$ aaBa | CDA | aa | DC
B $\rightarrow$ bB | bAB | bb | aS
C $\rightarrow$ Ca | bC | D
D $\rightarrow$ bD | Λ

Birim üretimler kısmen elendi

S $\rightarrow$ AaA | bA | BaB
A $\rightarrow$ aaBa | CDA | aa | DC
B $\rightarrow$ bB | bAB | bb | aS
C $\rightarrow$ Ca | bC | bD | Λ
D $\rightarrow$ bD | Λ

1. Λ üretimleri eleyin.

Tip-2 Gramer

S $\rightarrow$ AaA | bA | BaB
A $\rightarrow$ aaBa | CDA | aa | DC
B $\rightarrow$ bB | bAB | bb | aS
C $\rightarrow$ Ca | bC | bD | Λ
D $\rightarrow$ bD | Λ

Λ üretimler kısmen elendi

S $\rightarrow$ AaA | bA | BaB
A $\rightarrow$ aaBa | CDA | aa | DC | CA
A $\rightarrow$ DA | C | D | Λ
B $\rightarrow$ bB | bAB | bb | aS
C $\rightarrow$ Ca | bC | bD | b | a
D $\rightarrow$ bD | b

1. Λ üretimleri eleyin.

Tip-2 Gramer

$S \rightarrow AaA \mid bA \mid BaB$
$A \rightarrow aaBa \mid CDA \mid aa \mid DC \mid CA$
$A \rightarrow DA \mid C \mid D \mid \Lambda$
$B \rightarrow bB \mid bAB \mid bb \mid aS$
$C \rightarrow Ca \mid bC \mid bD \mid b \mid a$
$D \rightarrow bD \mid b$

Λ üretimler elendi

$S \rightarrow AaA \mid bA \mid BaB \mid a \mid b$
$A \rightarrow aaBa \mid CDA \mid aa \mid DC \mid CA$
$A \rightarrow DA \mid C \mid D \mid CD$
$B \rightarrow bB \mid bAB \mid bb \mid aS$
$C \rightarrow Ca \mid bC \mid bD \mid b \mid a$
$D \rightarrow bD \mid b$

1. Birim üretimleri eleyin.

Tip-2 Gramer

$S \rightarrow AaA \mid bA \mid BaB \mid a \mid b$
$A \rightarrow aaBa \mid CDA \mid aa \mid DC \mid CA$
$A \rightarrow DA \mid CD \mid C \mid D$
$B \rightarrow bB \mid bAB \mid bb \mid aS$
$C \rightarrow Ca \mid bC \mid bD \mid b \mid a$
$D \rightarrow bD \mid b$

Birim üretimler elendi

$S \rightarrow AaA \mid bA \mid BaB \mid a \mid b$
$A \rightarrow aaBa \mid CDA \mid DC \mid CA \mid aa$
$A \rightarrow DA \mid CD \mid Ca \mid bC \mid bD \mid a \mid b$
$B \rightarrow bB \mid bAB \mid aS \mid bb$
$C \rightarrow Ca \mid bC \mid bD \mid a \mid b$
$D \rightarrow bD \mid b$

Dönüşüm 2 ve 3 numaralı adımlarla devam eder.

Aşağıdaki dilleri ele alalım. Bu dilleri tanımlamak için bir PDA tasarlanabilir mi?

Örnek 1

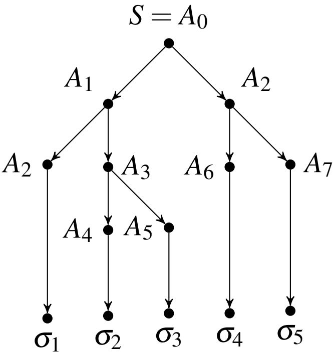
$L(G_1) = \{a^n b^n c^n | n \geq 0\}$

Örnek 2

$L(G_2) = \{xx \mid x \in \{a,b\}^*\}$

Bazı dillerin tanınabilmesi için yığınlardan daha yetenekli bellek yapılarına ihtiyaç duyulur.

Bağlamdan Bağımsız Diller için Pumping Lemma



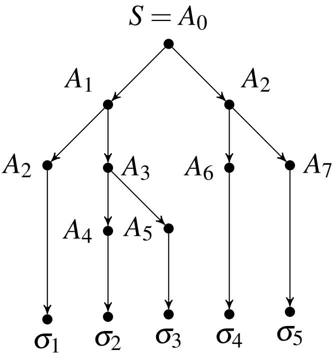
Chomsky Normal Formu

CNF ile tanımlanmış bir dilin ayrışma ağacı ikili bir ağaçtır. Ayrışma ağacının yaprakları soldan sağa geçildiğinde dilin bir sözcüğü oluşturulmuş olur.

Teorem

Her bağlamdan bağımsız G grameri için, CNFe uygun başka bir G_1 grameri $L(G_1) = L(G) - \Lambda$ olacak şekilde bulunmaktadır. G_1 grameri kurallarında sağ tarafta ya çift non-terminal ya da tek terminal (uç simge) bulunur.

Bağlamdan Bağımsız Diller için Pumping Lemma



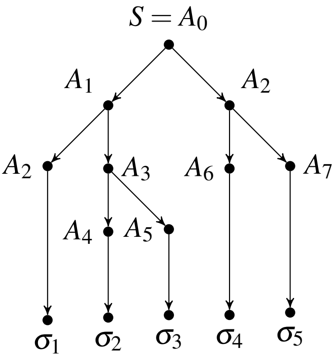
Teorem

CNF'e uygun bir gramer, G_{CNF}, tarafından üretilen bir ayrıştırma ağacı sonucu $u = \sigma_1\sigma_2\ldots\sigma_i\ldots\sigma_n \quad (\sigma_i \in \Sigma)$ katarı üretiliyor olsun.

u'yu her bir terminali üreten non-teminaller halinde yazarsak: $u = A_1 B_2 \ldots A_n$

Ör. $S = \sigma_1\sigma_2\sigma_3\sigma_4\sigma_5$ ya da
Ör. $S = A_2A_4A_5A_6A_7$

Bağlamdan Bağımsız Diller için Pumping Lemma



Ayrıştırma ağacı en uç ihtimalde bir ikili tam ağaç olabilir; bu durumda bu ağaçta yapraklara giden bütün yollar aynı uzunlukta ve bu ağacın derinliği[a] kadardır. Tam bir ikili ağaçta derinlik n ise yaprak sayısı 2^n'dir.

CNF bir gramerin ayrıştırma ağacında her yaprak, atası non-terminal olan tek çocuk olduğundan ağaç derinliği bir birim artar ($h = n + 1$) ama yapraklardan oluşan sözcük uzunluğu yine n derinlikli ağacın yaprak sayısı kadar olacaktır ($|w| = 2^{h-1} = 2^n$).

---

[a] Derinlik:Köke en uzak yaprağa olan yolun uzunluğu

Bağlamdan Bağımsız Diller için Pumping Lemma

Bu durumda $n+1$ derinlikli bir ayrıştırma ağacından en fazla 2^n uzunluklu bir sözcük türetilebilir. Bu ağaçta

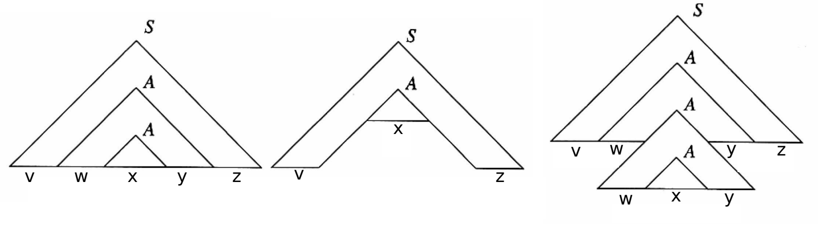
1. Her iç düğüm farklı bir non-terminale denk düşmüyorsa, başka bir deyişle kural tekrarı varsa, üretilen sözcükte tekrar edilebilecek alt katarlar bulunuyor demektir.
2. Her iç düğüm farklı bir non-terminale denk düşüyorsa bu durumda da 2^n'den daha uzun bir sözcük türetilebiliyorsa non-terminal tekrarı ortaya çıkacaktır.

Teorem

L bir bağlamdan bağımsız dil olsun. Bu durumda bütün $u \in L$ ve $|u| \geq n$ sağlayan n tamsayıları için, u katarı $u = vwxyz$ olarak yazılabilir. v, w, x, y, ve z aşağıdaki kuralları sağlar:

1. $|wy| > 0$
2. $|wxy| \leq n$
3. Bütün $m \geq 0$, $vw^mxy^mz \in L$.

Bağlamdan Bağımsız Diller için Pumping Lemma



Örnek 1

$$L = \{a^m b^m c^m | m \geq 0\}$$

Varsayımlar:

- L'nin CFG'si vardır
- $u = vwxyz$ ve $|u| \geq n$ olsun.
- n non-terminal sayısı olsun
- $|wxy| \leq n$

a^{n-1}	a	b^{n-2}	b	c^n
v	w	x	y	z

$$vw^2xy^2z = a^{n+1}b^nc^n$$

$$vxz = a^{n-1}b^{n-2}c^n$$

Örnek 2

$$L = \{xx \mid x \in \{a,b\}^*\}$$

Varsayımlar:

- L'nin CFG'si vardır
- $u = a^n b^n a^n b^n$ ve $|u| \geq n$ olsun
- n non-terminal sayısı olsun
- $|wxy| \leq n$

1. $\begin{array}{c|c|c|c|c} a^n b^3 & b & b^{n-4} & a & a^{n-1} b^n \\ v & w & x & y & z \end{array} \quad \begin{array}{l} a^n b^3 b^{n-4} a^{n-1} b^n \\ v w^0 x y^0 z \end{array}$
2. ya da $\begin{array}{c|c|c|c|c} a^n b & b & b^{n-3} & b & a^n b^n \\ v & w & x & y & z \end{array} \quad \begin{array}{l} a^n b^{n-2} a^n b^n \\ v w^0 x y^0 z \end{array}$
3. ya da $\begin{array}{c|c|c|c|c} a^{n-1} & a & b^{n-3} & b & b^2 a^n b^n \\ v & w & x & y & z \end{array} \quad \begin{array}{l} a^{n-1} b^{n-1} a^n b^n \\ v w^0 x y^0 z \end{array}$

Örnek 3

$$L = \{x \in \{a,b,c\}^* \mid \#(a) < \#(b) \text{ and } \#(a) < \#(c)\}$$

Varsayımlar:

- L'nin CFG'si vardır
- $u = a^n b^{n+1} c^{n+1}$ ve $|u| \geq n$ olsun
- n non-terminal sayısı olsun
- $|wxy| \leq n$

1. $\begin{array}{c|c|c|c|c} a^{n-1} & a & b^{n-3} & b & b^3c^{n+1} \\ v & w & x & y & z \end{array}$ $\quad \begin{array}{l} a^{n+1}b^{n+2}c^{n+1} \\ vw^2xy^2z \end{array}$

2. ya da $\begin{array}{c|c|c|c|c} a^nb^5 & b & b^{n-5} & c & c^n \\ v & w & x & y & z \end{array}$ $\quad \begin{array}{l} a^nb^nc^n \\ vw^0xy^0z \end{array}$