FORMAL DİLLER VE OTOMATLAR

KAYNAKLAR

- An Introduction to Formal Languages and Automata, Peter Linz.
- Introduction to Automata Theory, Languages and Computation, John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman.

KONULAR

- Hesaplama Teorisine Giriş (Kümeler, Fonksiyonlar, Bağıntılar, Graflar, Ağaçlar, İspat Yöntemleri)
- Sonlu Otomatlar (Finite Automata)
  - Gerekirci Sonlu Otomatlar (Deterministic Finite Automata)
  - Gerekirci Olmayan Sonlu Otomatlar (Nondeterministic Finite Automata)
- Düzgün Diller ve Gramerler (Regular Languages & Grammars)
- Düzgün Dillerin Özellikleri
- Bağlamdan Bağımsız Diller ve Gramerler (Context-Free Languages & Grammars)

KONULAR

- Bağlamdan Bağımsız Gramerlerin Basitleştirilmesi
- Push-Down Otomatlar
- Bağlamdan Bağımsız Dillerin Özellikleri
- Turing Makinaları
- Turing Makinalarının Diğer Modelleri

HESAPLAMA TEORİSİNE GİRİŞ

Bilgisayar donanımını modellemek için «OTOMAT» terimini kullanıyoruz.

- Bir otomat dijital bir bilgisayarın gerekli tüm özelliklerine sahip olan bir yapıdır.

-Girdiyi alır

-Çıktıya dönüştürmek için gerekli kararları verebilir

-Çıktı üretebilir

-Geçici depolama alanına sahip olabilir

HESAPLAMA TEORİSİNE GİRİŞ

- Semboller kümesinden ve bu sembolleri bir araya getirerek cümleleri oluşturan bazı kurallardan meydana gelir.

- Bu kurallar ile elde edilen dizgilerin tümünü içeren bir küme olarak düşünülebilir.

- Formal dilleri tanımlamak için ifadeler, gramerler ya da tanımlanan dile ait dizgileri kabul eden otomatlar kullanılır. Bu nedenle otomatla ilişkisi önemlidir.

KÜMELER

- $X \in S$; X, S kümesinin elemanıdır.
- $X \notin S$; X, S kümesinin elemanı değildir.

- $S = \{0, 1, 2\}$ 0, 1, 2 tamsayılarından oluşan S kümesi

- S=$\{i:\ i > 0,\ \ i$ çift tamsayı$\}$

Pozitif çift tamsayıların kümesi

KÜME İŞLEMLERİ

- $S_1 \cup S_2 = \{x:\ x \in S_1\ ya\ da\ x \in S_2\}\ Birleşim$
- $S_1 \cap S_2 = \{x:\ x \in S_1\ ve\ x \in S_2\}\ Kesişim$
- $S_1 - S_2 = \{x:\ x \in S_1\ ve\ \ x \notin S_2\}\ Fark$

- $\overline{S}$: S *kümesinin tümleyenidir ve* S *kümesinin*

elemanları **dışındaki tüm elemanların kümesidir.**

$\overline{S}$=$\{x:\ \ x \in U,\ \ x \notin S\}$

U→ Evrensel küme

Ø = { } Boş küme

$S \cup$ **Ø** = $S -$ **Ø** = S

$S \cap$ **Ø** = **Ø** $\overline{\text{Ø}}$ = **U** $\overline{\overline{S}} = S$

De Morgan Kuralları

- $\overline{S_1 \cup S_2} = \overline{S_1} \cap \overline{S_2}$
- $\overline{S_1 \cap S_2} = \overline{S_1} \cup \overline{S_2}$

- $S_1 \subseteq S$; S_1 , S *kümesinin bir alt kümesidir*

S_1'in her elemanı aynı zamanda S'in de elemanıdır.

- S_1 ve S_2 *kümelerinin ortak elemanları yoksa*

$$S_1 \cap S_2 = \emptyset$$

De Morgan Kuralları

- Sonlu bir S kümesinin boyutu eleman sayısıdır ve |S| ile gösterilir. Bu kümenin bütün alt kümeleri $2^{|S|}$ ile gösterilir.

- S = {a, b, c} üç elemanlı kümenin alt kümeleri sayısı $2^3 = 8'$dir. |S|=3

$2^{|S|}$={Ø, {a}, {b}, {c}, {a,b}, {a,c}, {b,c}, {a,b,c}}

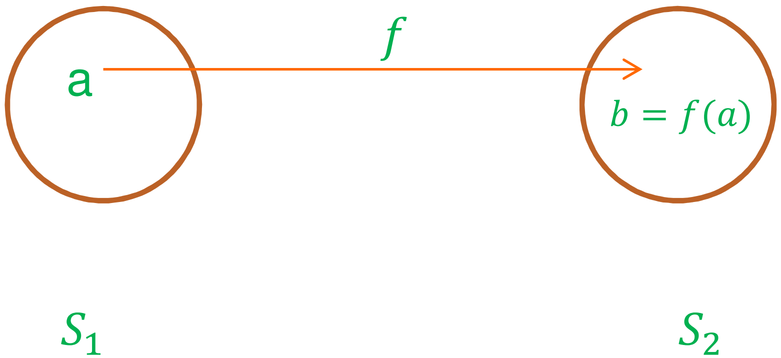
İKİ KÜMENİN KARTEZYEN ÇARPIMI

- $S = S_1 \times S_2 = \{(x, y) \colon x \in S_1, \mathrm{y} \in S_2\}$
- $S_1 = \{2{,}4\} \quad ve \quad S_2 = \{2, 3, 5, 6\}$
- $S_1 \times S_2 = \{(2{,}2), (2{,}3), (2{,}5), (2{,}6), (4{,}2), (4{,}3), (4{,}5), (4{,}6)\}$
- (4,2)$\in S_1 \times S_2$ fakat (2,4) $\notin S_1 \times S_2$

FONKSİYONLAR VE BAĞINTILAR

- Fonksiyon, bir kümenin elemanlarını diğer bir kümenin tek bir elemanına atayan bir kural olarak tanımlanır. f bir fonksiyon olmak üzere;

  $f: S_1 \to S_2$ olarak gösterilir.

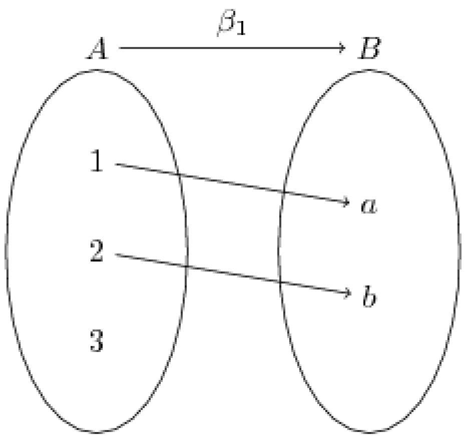


S_1 tanım kümesi

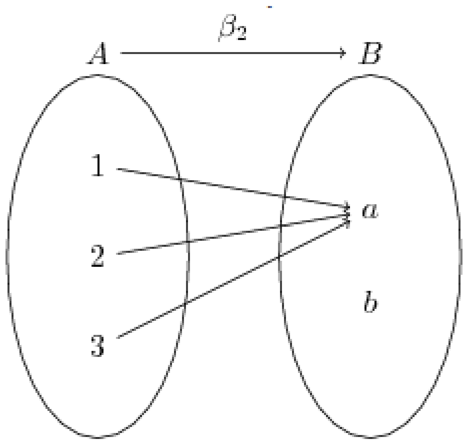
S_2 değerler kümesi

* İlk kümedeki her eleman ikinci kümedeki bir elemanla eşleşecek

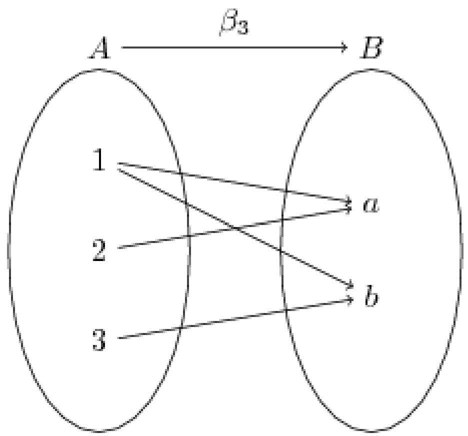
* İlk kümenin hiç bir elemanı ikinci kümede birden fazla elemanla eşleşmeyecek.



A kümesinin bir elemanı açıkta kaldığı için fonksiyon değildir.

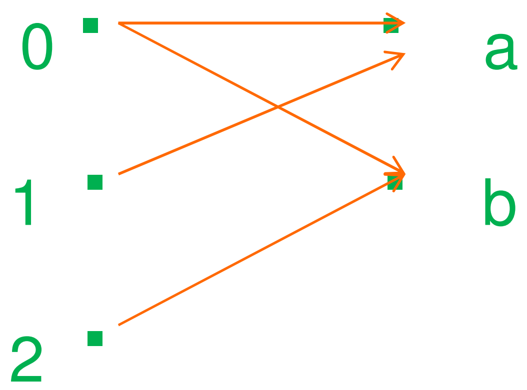


Fonksiyondur. A'nın tüm elemanları B'de bir elamanla eşleşiyor. B'nin elemanı açıkta kalabilir.



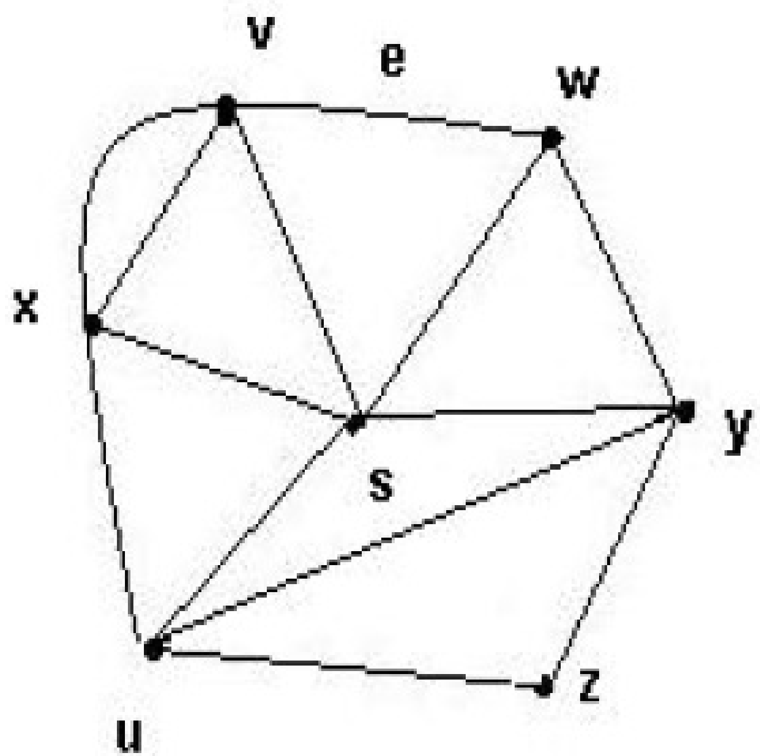
Fonksiyon değildir çünkü 1, B'nin 2 elemanıyla eşleşmiş.

Fonksiyonlarda tanım kümesindeki her x_i'nin değerler kümesinde tek bir karşılığı vardır. Aksi durumda bu küme bir bağıntı tanımlar. Bağıntılar fonksiyonlardan daha geneldir. Yani bağıntılarda tanım kümesindeki bir elemanın değer kümesinde bağlı olduğu birden fazla eleman olabilir.

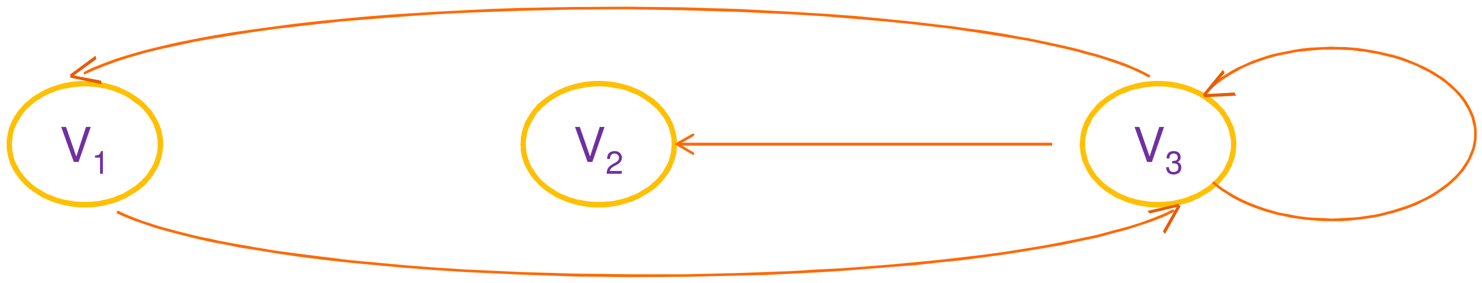


GRAFLAR VE AĞAÇLAR

- G grafi nedir ?
- G = (V, E)
  - V = V(G) = dügümler kümesi
  - E = E(G) = kenarlar kümesi
- **Örnek:**
  - V = {s, u, v, w, x, y, z}
  - E = {(x,s), $(x,v)_1$, $(x,v)_2$, (x,u), (v,w), (s,v), (s,u), (s,w), (s,y), (w,y), (u,y), (u,z),(y,z)}



- Kenar bir çift dügüm ile etiketlenmis olup
  e = (v,w) seklinde gösterilir.

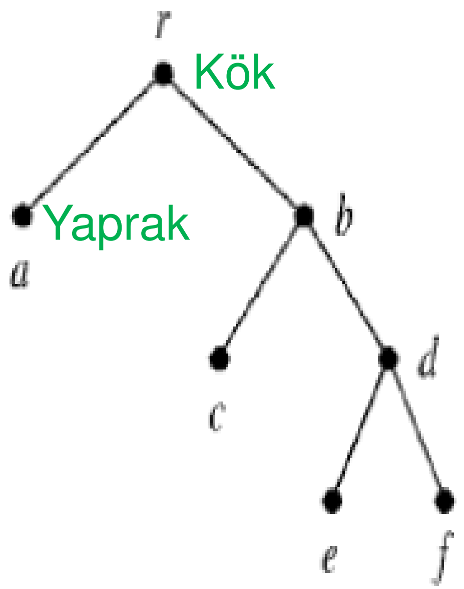


$\{V_1, V_2, V_3\}$; $\{(V_1, V_3), (V_3, V_1), (V_3, V_2), (V_3, V_3)\}$

$(V_1, V_3), (V_3, V_3), (V_3, V_1)$ → cycle (aynı kenarı kullanmadan döner)

V_1'den V_2'ye basit yol→ $(V_1, V_3), (V_3, V_2)$

- Ağaç yapıları da grafların özel bir çeşididir.
- Ağaçlarda döngü kullanılmaz ve kök(root) adı verilen tek bir uç noktası ile başlar.
- Bu şekilde kökten uçlara giden sadece tek bir yol vardır.



- Bu şekil 3 seviyeli tam ikili ağaçtır.
- 0 seviyeli düğüm kök olan r dir.
- 1 seviyeli düğümleri a ve b dir.
- 2 seviyeli düğümleri c ve d dir.
- 3 seviyeli düğümleri e ve f dir.

Ağaç yapılarında uçlar arasında ebeveyn-çocuk ilişkisi bulunur.

İSPAT YÖNTEMLERI

- Tümevarım yöntemiyle ispatlama (Proof by induction)
- Çelişme yöntemiyle ispatlama (Proof by contradiction)

Tümevarım Yöntemi ile İspatlama

P_1, P_2 ,… ispatlamak istediğimiz ifadeler olsun ve aşağıdakiler sağlansın.

1. $k \geq 1$ için P_1, P_2 ,… P_k'nın doğru olduğunu biliyoruz.
2. Herhangi bir $n \geq k$ için P_1, P_2 ,… P_n için doğru olanlar, P_{n+1} için de doğrudur.

İSPAT YÖNTEMLERI

- Her tümevarımda 3 kısım bulunur:

$P_1, P_2, \dots P_k$ **taban(basis)**

$P_1, P_2, \dots P_n$ doğru olduğunu varsayma. **Tümevarım varsayımı (inductive assumption)**

P_n ile P_{n+1} arasındaki ilişkinin kurulması **tümevarım aşaması (inductive step)**

Örnek: $n! > 2^n, \quad n \geq 4$

$4! = 4.3.2.1 = 24$ ve $2^4 = 16$ → $4! > 2^4$

Basis (taban) $4! > 2^4$

Tümevarım varsayımı $k \geq 4$ için $k! > 2^k$ olduğu varsayılır.

$(k+1)! > 2^{k+1}$?

$(k+1)k! > 2*2^k$ ------- $2*k! > 2*2^k$ → $k+1>2$ ✓

$k! > 2^k$ ✓

Örnek:

$1+3+5+....+2n-1 = n^2$, $n>0$ olduğunu göster.

İspat

Basis step: $1 = 1^2 \rightarrow 1 = 1$ ✓

Tümevarım varsayımı:

$1+3+5+...+2k-1= k^2$ kabul et

Tümevarım aşaması:

$$\underbrace{1+3+5+...+2k-1}_{k^2} + 2k+1= (k+1)^2$$

$k^2+2k+1 = (k+1)^2$ ✓

Örnek:

$2^0 + 2^1 + \ldots + 2^N = 2^{N+1} - 1$ P(N) ifadesinin tüm $N \geq 0$ için doğru olduğunu ispatlayalım.

Basis(taban): P(0)'ın doğru olduğunu gösterelim

$P(0) = 2^0 = 2^1 - 1$ DOĞRUDUR

Tümevarım Varsayımı: $K \geq 0$ ve $P(K)$

$2^0 + 2^1 + \ldots + 2^k = 2^{k+1} - 1$ doğru olduğu kabul edilir

Tümevarım aşamasında P(K+1) ifadesinin doğruluğunu ispatla

$2^0 + 2^1 + \ldots + 2^{k+1} = 2^{(k+1)+1} - 1$

Tümevarım Aşaması:

$$
\begin{aligned}
2^0 + 2^1 + \ldots + 2^{k+1} &= (2^0 + 2^1 + \ldots + 2^k) + 2^{k+1} \\
&= (2^{k+1} - 1) + 2^{k+1} \quad \text{(Tümevarım varsayımı)} \\
&= 2 * 2^{k+1} - 1 \\
&= 2^{(k+1)+1} - 1 \quad ✓
\end{aligned}
$$

ÇELIŞME YÖNTEMI ILE ISPATLAMA

- İspatlanmak istenen şeyin tersi ele alınır ve bunun bir çelişkiye yol açtığı gösterilir. Buradan da, başta varsayılanın yanlış olduğu sonucu çıkar. Bu nedenle de tersi doğru olmalıdır.
- Örneğin $\sqrt{2}$'nin rasyonel bir sayı (n/m şeklinde yazılabilen sayı) olmadığını bu yöntemle ispatlayalım.

1. $\sqrt{2}$ rasyonel bir sayıdır. (varsayım)
2. $\sqrt{2}$ = n/m, m ve n ortak çarpanı olmayan tamsayılar.
3. $2 = n^2 / m^2$
4. $2m^2 = n^2$
5. $n^2 = 2j$ (2, n^2'nin bir çarpanıdır. Yani n^2 çifttir)
6. $n=2k$ (n^2 çift → n de çift olmalıdır)
7. $2m^2 = 4k^2$
8. $m^2 = 2k^2$
9. $m^2 = 2h$
10. ÇELİŞME (6 ve 9, 2 ile çelişmektedir. İkisi de çift sayı, ortak çarpanı var)

DİLLER

- Sembollerin sonlu kümesi Σ ile gösterilir ve alfabe olarak adlandırılır. Alfabenin sembolleri biraraya getirilerek stringler(katarlar) oluşturulur.
- Örneğin $\Sigma = \{a,b\}$ alfabesi için, abab ve aaabbba bu alfabe üzerinde oluşturulan katarlardır.
- 010110 dizgisi $\Sigma = \{0,1\}$ alfabesine aittir. 111 dizgisi de aittir.

Alphabets and Strings

- We will use small alphabets: $\Sigma = \{a, b\}$
- Strings

$$a$$

$$ab \qquad u = ab$$

$$abba \qquad v = bbbaaa$$

$$baba \qquad w = abba$$

$$aaabbbaabab$$

Σ'nın elemanları için genellikle küçük harfler kullanılır ve string isimleri için genellikle u, v, w… kullanılır.

(Katarları Birbirine Bağlama (Birleştirme))

String Concatenation

$$w = a_1 a_2 \cdots a_n$$

$$v = b_1 b_2 \cdots b_m$$

$$wv = a_1 a_2 \cdots a_n b_1 b_2 \cdots b_m$$

$abba$

$bbbaaa$

Concatenation => $abbabbbaaa$

String Reverse

$$w = a_1 a_2 \cdots a_n$$

$$w^R = a_n \cdots a_2 a_1$$

$ababaaabbb$ Reverse => $bbbaaababa$

String Length

$$w = a_1 a_2 \cdots a_n$$

Length = $|w| = n$

$$|abba| = 4$$

$$|aa| = 2$$

$$|a| = 1$$

Length and Concatenation

$$|uv| = |u| + |v|$$

$$u = aab, \quad |u| = 3$$

$$v = abaab, \quad |v| = 5$$

$$|uv| = |aababaab| = 8$$

$$|uv| = |u| + |v| = 3 + 5 = 8$$

Empty String

- A string with no letters: λ

$$|\lambda| = 0$$

- Observations:

$$\lambda w = w\lambda = w$$

$$\lambda abba = abba\lambda = abba$$

Substring (Altdizgi)

Ardışık karakterlerin altdizisi:

String	Substring
$\underline{ab}bab$	ab
$\underline{abba}b$	$abba$
$ab\underline{b}ab$	b
$a\underline{bbab}$	$bbab$

Prefix and Suffix

$$w = uv$$

prefix		suffix
λ		$abbab$
	$abbab$	
a		$bbab$
ab		bab
abb		ab
$abba$		b
$abbab$		λ

More Concatenation

$$w^n = \underbrace{ww\cdots w}_{n}$$

$$w^0 = \lambda$$

$$(abba)^2 = abbaabba$$

$$(abba)^0 = \lambda$$

- L={A,B, ……,Z,a,b, ……,z} , D={0,1, ……, 9}
- LUD: harflerden ve rakamlardan oluşan set
- LD: başı harf sonu rakam olan stringlerin kümesi
- L4: 4 harften oluşan stringlerin kümesi
- L*: bos stringi de içine alan harflerden oluşan stringlerin kümesi

The * Operation

Σ bir alfabe olmak üzere, Σ^* sıfır ve daha fazla sembolün biraraya gelmesiyle elde edilen dizilerin kümesini göstermektedir. Σ^* kümesinde her zaman λ vardır.

$$\Sigma = \{a, b\}$$

$$\Sigma^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$$

The + Operation

Given an alphabet Σ

Σ^+ Is the set of all strings over the alphabet minus λ

$$\Sigma^+ = \Sigma^* - \lambda$$

$$\Sigma = \{a, b\}$$

$$\Sigma^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$$

$$\Sigma^+ = \{a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$$

Σ alfabesi sembollerin sonlu kümesinden oluştuğu halde, Σ^* ve Σ^+, bu kümelerdeki dizilerin uzunluklarında herhangi bir kısıtlama olmayacağı için her zaman sonsuzdur.

Languages (DİLLER)

Dil kavramı çok genel olarak Σ* kümesinin bir alt kümesi olarak tanımlanabilir. (Dil, verilen alfabe üzerinde tanımlı stringlerin kümesidir)

Σ bir alfabe ise ve L ⊆ Σ* ise L, Σ üzerinde tanımlanmış bir dildir.
L diline ait bir dizgi de cümle olarak adlandırılır.

$$\Sigma = \{a, b\}$$

$$\Sigma^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, \ldots\}$$

Some Languages over this alphabet include:

$$\{\lambda\}$$

$$\{a, aa, aab\}$$

$$\{\lambda, abba, baba, aa, ab, aaaaaa\}$$

Dilin her bir elemanı bir cümledir.
Sonlu sayıda cümlelerden oluştuğu için de sonlu bir dildir.

Languages, Sets, and Notations

Sets $\quad \emptyset = \{\,\} \neq \{\lambda\}$

Set size $\quad |\{\,\}| = |\emptyset| = 0$

Set size $\quad |\{\lambda\}| = 1$

String length $\quad |\lambda| = 0$

An Infinite Language

$$L = \{a^n b^n : n \geq 0\}$$

$$\left.\begin{array}{l} \lambda \\ ab \\ aabb \\ aaaaabbbbb \end{array}\right\} \in L \qquad abb \notin L$$

What is the alphabet associated with this language?

Languages and Operations

Diller, kümeler ile gösterilebildiklerinden dolayı, iki dil arasında birleşme, kesişme ve fark işlemleri tanımlanabilmektedir.

$$\{a, ab, aaaa\} \cup \{bb, ab\} = \{a, ab, bb, aaaa\}$$
$$\{a, ab, aaaa\} \cap \{bb, ab\} = \{ab\}$$
$$\{a, ab, aaaa\} - \{bb, ab\} = \{a, aaaa\}$$

The Complement of a Language

Intuitively, all strings not in the language. More precisely:

$$\overline{L} = \Sigma^* - L$$

$$\overline{\{a, ba\}} = \{\lambda, b, aa, ab, bb, aaa, \ldots\}$$

Reverse

$$L^R = \{w^R : w \in L\}$$

$$\{ab, aab, baba\}^R = \{ba, baa, abab\}$$

$$L = \{a^n b^n : n \geq 0\}$$

$$L^R = \{b^n a^n : n \geq 0\}$$

Concatenation

$$L_1 L_2 = \{xy : x \in L_1, y \in L_2\}$$

$$\{a, ab, ba\}\{b, aa\}$$

$$= \{ab, aaa, abb, abaa, bab, baaa\}$$

More Concatenation

$$L^n = \underbrace{LL\cdots L}_{n}$$

$$L^0 = \{\lambda\}$$

$$\{a, bba, aaa\}^0 = \{\lambda\}$$

$$\{a,b\}^3 = \{a,b\}\{a,b\}\{a,b\} =$$
$$\{aaa, aab, aba, abb, baa, bab, bba, bbb\}$$

Star-Closure (Kleene *)

$$L^* = L^0 \cup L^1 \cup L^2 \cdots$$

$$\{a, bb\}^* = \left\{\begin{array}{l} \lambda, \\ a, bb, \\ aa, abb, bba, bbbb, \\ aaa, aabb, abba, abbbb, \ldots \end{array}\right\}$$

GRAMERLER

- Bir G grameri G = (V, T, S, P) ile tanımlanır.
- V→ değişkenlerin sonlu kümesi
- T→ terminal sembollerin sonlu kümesi (alfabenin elemanları)
- $S \in V$ → başlangıç değişkeni
- P→ dizilerin sonlu kümesi (productions)

Bir gramerdeki en önemli kavram türetim kurallarıdır (production rules).

X→Y türetim kuralı tanımı

$X \in (V \cup T)^{+}$ ve $Y \in (V \cup T)^{*}$

GRAMERLER

- Örneğin verilen bir w = uxv dizisi için x→y kuralını uygularsak yeni bir z = uyv dizisini elde ederiz. Bu da w => z şeklinde gösterilir. z, w'dan türetilmiştir.
- u => v, v => w, w => x, x => y, y => z → $u \overset{*}{\Rightarrow} z$
- w_1 => w_2 =>w_3=>… => w_n → $w_1 \overset{*}{\Rightarrow} w_n$

0 veya daha fazla adımda w_1'den w_n türetilmiştir.

Bu türetim kurallarını farklı şekillerde uygulayarak, verilen bir gramer ile birçok dizgi elde edilebilir. Bu şekilde elde edilen terminal dizilerin tümünden oluşan küme, gramer tarafından tanımlanan «dil» dir.

Tanım: G = (V, T, S, P) bir gramer ise

$L(G) = \{w \in T^* : S \overset{*}{\Rightarrow} w\}$ G grameri kullanılarak oluşturulmuş bir dildir.

Örnek: G = ({S}, {a,b}, S, P) grameri için P şu şekilde verilmiş olsun:

$S \rightarrow aSb$

$S \rightarrow \lambda$

Bu gramer için dili bulunuz.

$S \Rightarrow aSb \Rightarrow ab$

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

$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaabbb$

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

Örnek: $L = \{a^n b^{n+1} : n \geq 0\}$ dilini üreten grameri bulun.

$S \rightarrow Ab$

$A \rightarrow aAb$

$A \rightarrow \lambda$

türetim kuralları ile $G = (\{S, A\}, \{a, b\}, S, P)$

Örnek: Aşağıda türetim kuralları verilen gramerin oluşturduğu dili tanımlayınız.

$S \rightarrow aA$

$S \rightarrow \lambda$

$A \rightarrow bS$

$L(G) = \{(ab)^n : n \geq 0\}$