OTOMATA

FORMAL DİLLER VE OTOMATA

Otomata nedir?

- Teorik bilgisayar biliminde soyut makineleri ve bu makineleri kullanarak hesaplama problemlerinin çözülmesini araştıran bilim dalına otomata teorisi denir.

* Soyut makineler "otomat" olarak adlandırılır.

* Otomata teorisi ile birlikte anılan formal / biçimsel diller kuramı ise;
  * Dilde bulunan bütün öğeler ve dilin ulaşabileceği sınırlar belirli kurallar dahilinde tanımlanabiliyorsa bu dillere biçimsel dil adı verilir.

* Dilin en küçük yapıtaşı olan alfabeyi V ile gösteririz.
* Sözcük bilgisindeki bazı elemanlar diğer simgeler ile yer değiştirilemezler. Bunlara uç (T) denir.
* Uç olmayanlar ise N ile gösterilir.

* $G=(V, T, S, P)$ ab yapısıdır.

  G = Gramer
  V = alfabe / sözcük
  T = uç
  S = başlangıç değişkeni
  P = sonlu türetme kuralları kümesinden oluşur.

  $V - T = N$

  P'deki her türetme kuralı sol tarafında en az bir uç olmayan içermelidir.

  $\lambda \Rightarrow$ harf içermeme durumu

ÖRNEK: $G = (V, T, S, P)$ 'deki $V = \{a, b, A, B, S\}$

$T = \{a, b\}$, S başlangıç simgesi ve

$P = \{S \to ABa, A \to BB, B \to ab, AB \to b\}$ olsun

G öbek yapısı gramerine örnektir.

Kleene Star * ile λ dahil alfabedeki tüm karakterlerin mümkün olan tüm birleşimlerini elde eder.
Bir alfabenin önüne * konulduğu zaman istediği kadar kombinasyon yapabileceği anlamına gelir.

↳ $a^* \to aaa, aa, a, aaaaa$

ÖRNEK: $V = \{S, A, a, b\}$, $T = \{a, b\}$, S başlangıç simgesi ve

$P = \{S \to aA, S \to b, A \to aa\}$ türetme kuralları kümesi olsun

Gramerin dili: $L(G)$ nedir?

$S \to aA^{aa} = aaa$
$S \to b$

$L(G) = \{aaa, b\}$

ÖRNEK: $\{0^n 1^n \mid n = 0, 1, 2 \ldots\}$ kümesini oluşturan öbek yapısı gramerini veriniz.

↳ $S \to 0S1$
$S \to \lambda$

000111, 01, 0011 ...
0 ve 1 eşit sayıda

$P = \{S \to 0S1, S \to \lambda\}$

ÖRNEK: $\{0^m 1^n \mid m \text{ ve } n \text{ negatif olmayan tam sayılar}\}$

kümesini oluşturan öbek yapısı gramerini verin.

G. $V = \{S, 0, 1\}$ $T = \{0, 1\}$ olsun o zaman; $P = ?$

$S \rightarrow 0S$, $S \rightarrow S1$, $S \rightarrow \lambda$

ÖRNEK: $\{0^n 1^n 2^n \mid n = 0, 1, 2 \ldots\}$, $V = \{0, 1, 2, S, A, B, C\}$

$T = \{0, 1, 2\}$ S başlangıç durumu $P = ?$

$S \rightarrow C$, $C \rightarrow 0CAB$, $S \rightarrow \lambda$

$BA \rightarrow AB$, $0A \rightarrow 01$, $1A \rightarrow 11$

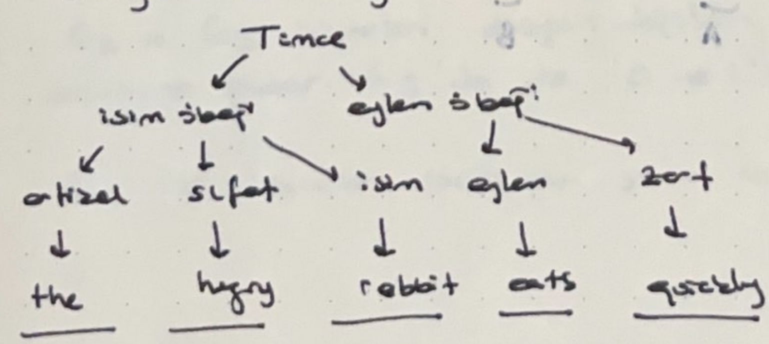
$1B \rightarrow 12$, $2B \rightarrow 22$

Bir adet örnek

$0CAB \rightarrow 0SAB \rightarrow 0AB \rightarrow 01B \rightarrow 012$

Türetme Ağacı

- Ağaç şeklinde gösterilir.
- Kök başlangıç değerini temsil eder.
- Ağacın ara düğümleri ucu olmayan N'yi,
- Ağacın son düğümleri ise ucu T'yi temsil eder.



Gramer Tipleri

Tip	$w_1 \to w_2$ türetme kurallarındaki sınırlama	Açıklama
0	Sınırlama yok	—
1	$w_1 = LAr$ ve $w_2 = Lwr$ $A \in N$, $L, w, r \in (N \cup T)^*$ ve $w \neq \lambda \to w_1 = S$ $w_2 = \lambda$	Bağlama bağımlı
2	$w_1 = A$ (değişken olmak şartıyla)	Bağlamdan bağımsız
3	$w_1 = A$ $w_2 = aB$ veya $w_2 = a$ $A \in N$, $B \in N$ ve $a \in T$ veya $w_1 = S$ ve $w_2 = \lambda$	Düzgün gramer

ÖRNEK: $V = \{a, b, c, A, B, C, S\}$ $T = \{a, b, c\}$ S başlangıç değ.

Türetme kuralları ise; $P = \{S \to AB,\ A \to Ca,\ B \to Ba,$
$B \to Cb,\ B \to b,\ C \to cb,\ C \to b\}$

old. bilindiğine göre, cbab sözcüğünün gramer $G = (V, T, S, P)$ tarafından oluşturulan dile ait olup olmadığını belirle.

Çözüm: Yukarıdan-aşağı ayrıştırma yöntemi kullanılacaktır.

$$S \Rightarrow AB \Rightarrow CaB \Rightarrow cbaB \Rightarrow cbab$$

Tam tersi ise Aşağıdan-yukarı ayrıştırmadır.

$$\underbrace{c}_{C}bab \Rightarrow \underbrace{Ca}_{A}b \Rightarrow A\underbrace{b}_{B} \Rightarrow AB \Rightarrow S$$

Backus-Naur Biçimi (BNF Gramer)

* Komut / komut kümelerinin, belge biçimlendirmesinin ve iletişim protokollerinin söz dizimini belirlemek için kullanılan gösterimdir.

ÖRNEK: $\longrightarrow$ simgesi yerine ::= kullanılır.
uç olmayanlar ⟨ ⟩ ile gösterilir.

$A \longrightarrow Aa$, $A \to a$ ve $A \to AB$ türetme kuralı,

$\langle A \rangle ::= \langle A \rangle a \mid a \mid \langle A \rangle \langle B \rangle$

---

* Sonlu özdevinirler tarafından tanımlanan kümelere **dizgin kümeler** (regular expression - Rex) denir.

* Bir küme verildiğinde bu kümenin dizgin bir küme olup olmadığını bu kümeyi tanıyan sonlu bir özdevinirin bulunup bulunmaması tarafından belirlenir.

ÖRNEK: G_1 ilk ve son simgesi 1 olan en az 5 uzunluğundaki dizgiler kümesidir.

$G_2 = \{0^n 1^n \mid n \geq 1\}$

G_3 içinde eşit sayıda 0 ve 1 bulunan dizgiler kümesidir.

G_2 ve G_3 kümeleri dizgin değildir. Çünkü G_2'de n sonsuza gider. G_3'de de 0 ve 1'in sınırı belirlenmemiştir.

G_1 ise sınırları belli olan sonlu ve dizgin bir kümedir.

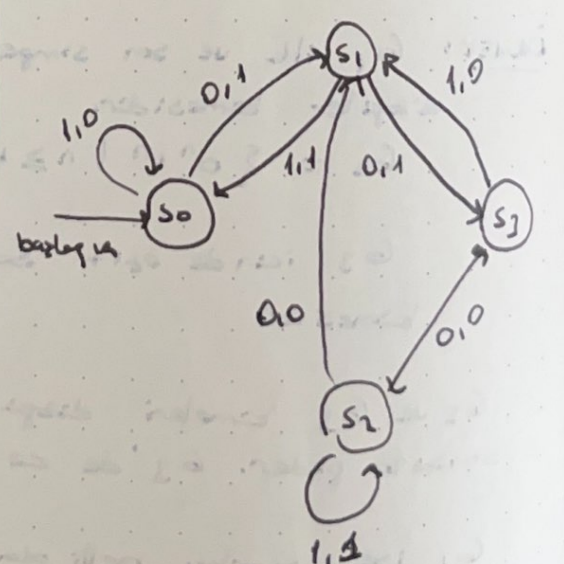
SONLU DURUMLU MAKİNELER (FSM)

* Bilgisayar bileşenlerini kapsayan bir çok türde makine, sonlu-durum makine adı verilen bir yapı kullanılarak modellenebilir.

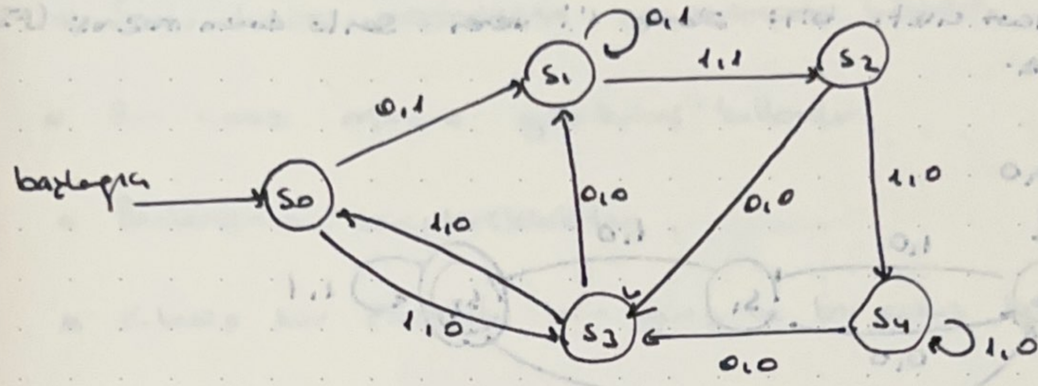
* Sonlu durum makine $M = (S, I, O, f, g, s_0)$ ile gösterilir.
  - $S \rightarrow$ sonlu durum kümesi
  - $I \rightarrow$ sonlu girdi alfabesi
  - $O \rightarrow$ sonlu çıktı alfabesi
  - $f \rightarrow$ her bir durum ve girdi çiftine yeni durum atar.
  - $G \rightarrow$ her bir durum ve girdi " çıktı ~~f~~ atar
  - $s_0 \rightarrow$ başlangıç durumu

ÖRNEK: $S = \{s_0, s_1, s_2, s_3\}$, $I = \{0,1\}$ ve $O = \{0,1\}$ bilgilerine sahip olan ~~…~~ durum tablosu, sonlu-durum makineyi tanımlar. Tabloya göre sonlu-durum makinesi için durum çizelgesini çiz.

Durum	f		g	
	Girdi		Çıktı	
	0	1	0	1
s_0	s_1	s_0	1	0
s_1	s_3	s_0	1	1
s_2	s_1	s_2	0	1
s_3	s_2	s_1	0	0

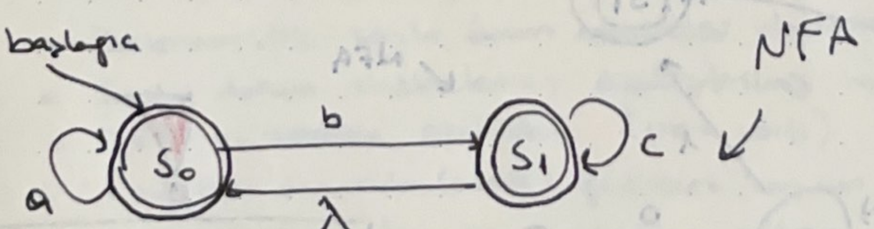


ÖRNEK: Şekildeki durum çizeneğine sahip sonlu-durum makine için durum tablosunu oluşturunuz.

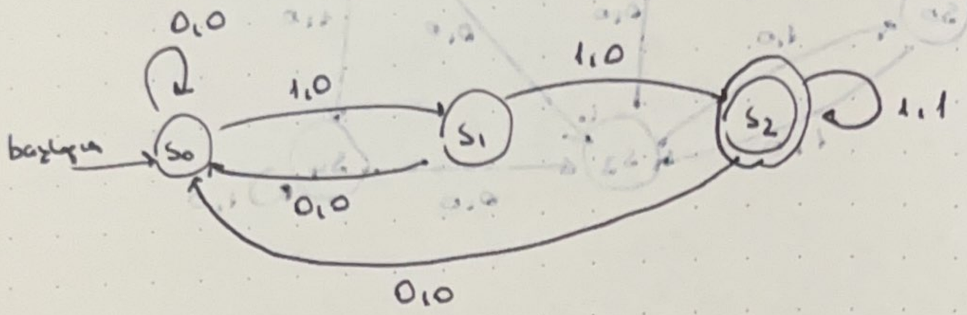


Durum	f Girdi 0	f Girdi 1	g Çıktı 0	g Çıktı 1
s_0	s_1	s_3	1	0
s_1	s_1	s_2	1	1
s_2	s_3	s_4	0	0
s_3	s_1	s_0	0	0
s_4	s_3	s_4	0	0

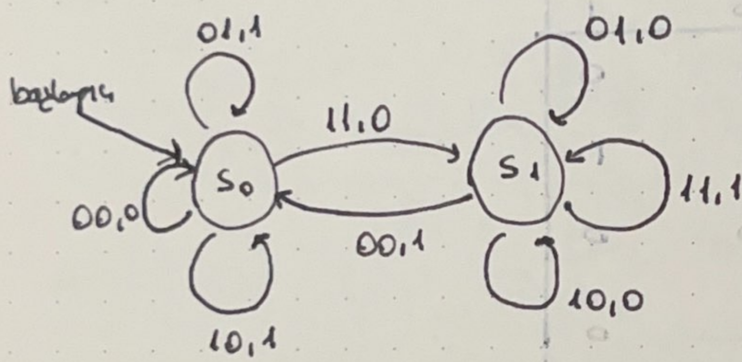
ÖRNEK: $P = (a + bc^*)^*$ dizgin değimini tanıyan FSM çiz.



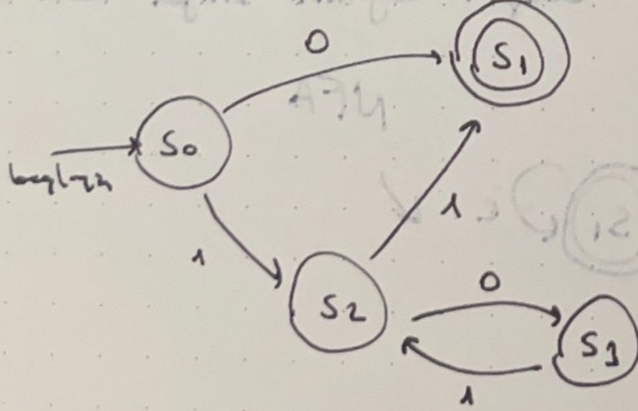
ÖRNEK: Alınan son üç bitin tümü ortak ve ortak 1 ise mevcut çıktı biti olarak 1'i veren sonlu durum makinesi (FSM) çiz.



ÖRNEK: İki tabanında iki pozitif tamsayıyı toplayan FSM çiz.



ÖRNEK: $P = (0 + 1(01)^* 1)$ FSM çiz ?



↙ NFA

+ → ∨ → veya anlamında

• → ∧ → ve anlamında

MEALY & MOORE MODELLERİ

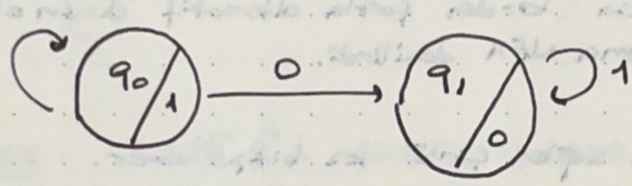
* Sonlu durum makinelerin gösteriminde kullanılır.

* En çok Moore gösterimi kullanılır.

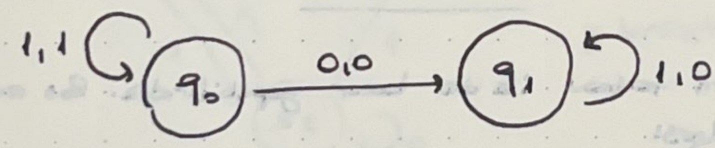
* Birbirlerine dönüştürülebilirler.

* Klasik bir FSM'de bir giriş ve bir çıkış bulunur.

* Moore makinelerinde çıkış değerleri düğümlere yazılırken giriş değerleri kenarlar üzerinde gösterilir.



* Mealy makinelerinde ise giriş ve çıkış değerleri kenarlar üzerinde aralarında slash (/) konularak gösterilir.



* Aynı işlemi yapan iki makinenin farklı gösterimleri

DETERMİNİSTİK SONLU OTOMAT (DFA)

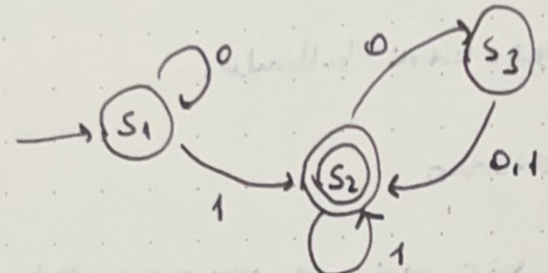
* Deterministik sonlu durum makinesi olarak da bilinir.
* Sonlu durum makinelerinin özelleştirilmiş halidir.
* DFA'nın ~~[illegible]~~ özellikleri (olma şartı)

1 ↳ Her durumdan (state) gidilecek koşulun tek bir durum göstermesi. Yani bir kelime ile sadece bir duruma geçilebilir.

2 ↳ Herhangi bir girdi için tek bitiş durumu kabul edilir.

3 ↳ Lambda (veya epsilon) durumlar arasında yer alamaz.

ÖRNEK: Aşağıdaki sonlu durum makinesinin (FSM) deterministik olup olmadığını bulunuz.



Çözüm: 1. şartı sağlar çünkü herhangi bir durum sadece bir state gitmiş belirsizlik yoktur. Şayet herhangi birisi için birden fazla alternatif durum olsaydı bu duruma NFA denilirdi.

2. şartı sağlar çünkü tek bitiş vardır.

3. durumu sağlar çünkü lambda yoktur.

Bu makine Deterministiktir.

Bu çözümü durum tablosu ile de ~~tasar~~ yapabilirdik. Bu makinenin durum ~~geçiş~~ çizelgesi:

	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

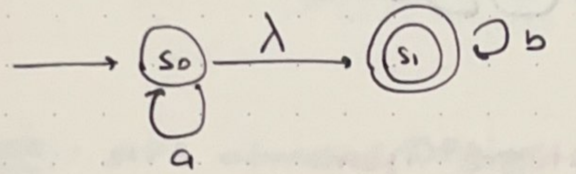
Bu hücrelerde birden fazla durum bulunması halinde NFA dır. Her hücrede tek durum olması ise DFA olarak yorumlanabilir.

NON-DETERMINISTIK SONLU OTOMATA (NFA)

* DFA'nın tersine her durumda gidişin kesiz old. ve her durum için bir sonraki kelimede nereye gidileceğinin belli olmadığı otomatlardır.

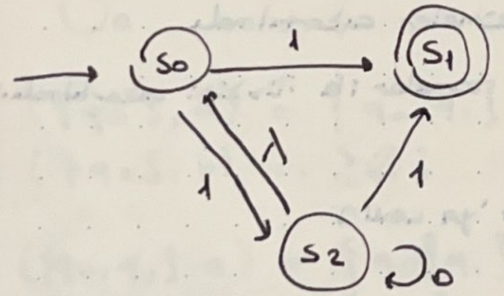
* DFA kurallarına uymayan tüm otomatlar NFA'dır.

ÖRNEK: $a^* b^*$ dizgi ifadesini çiz.



NFA'dır çünkü λ var.

ÖRNEK: $1+1^*0^*1$ dizgi ifadesini çiz.



NFA'dır çünkü λ var ve bir kelime iki duruma gitmiştir.

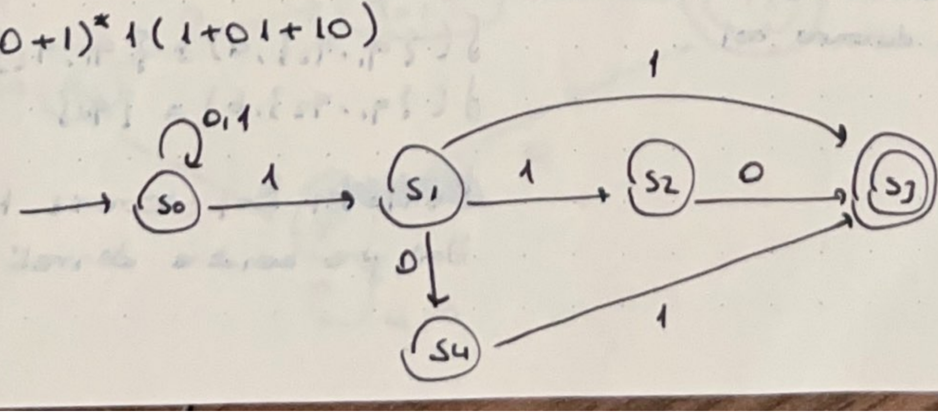
ÖRNEK: {0,1} alfabesinde 11, 101 yada 110 ile biten dizileri tanıyan makineyi çiz.

a) Yukarıdaki şartlara uyan dizgin deyimi yaz.

b) NFA'yı çiz.

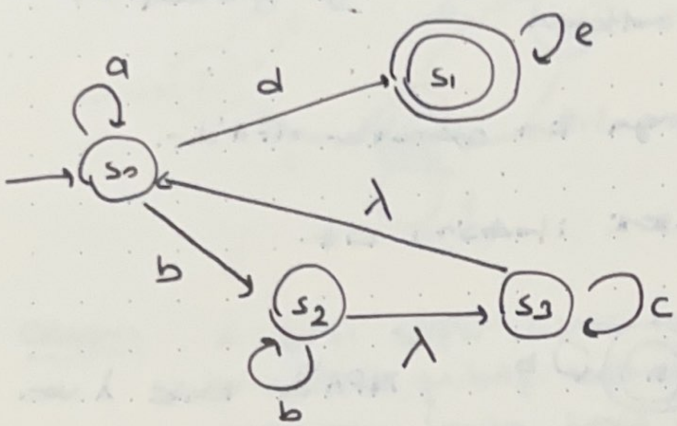
a) $(0+1)^* 1 (1+01+10)$

b)



ÖRNEK: $(a+bb^*c^*)^*de^*$ NFA'yı çiz.

Çekebilecekleri → d, ad, bd, aaaadeee, bcccd, bcde ...

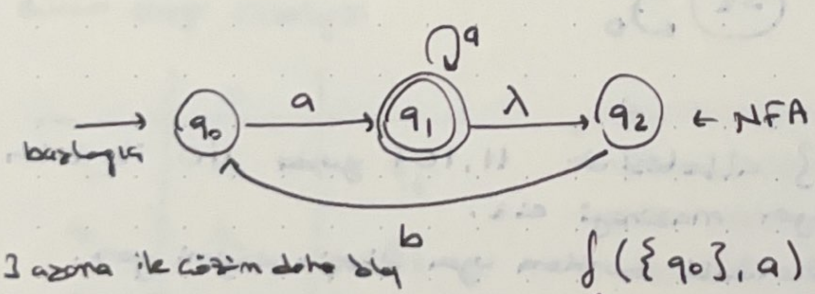


NFA'dan DFA'ya Dönüştürme İşlemi

• Bu dönüşüm sırasında 2 temel adım uygulanır.

1. Birbiri ile özdeş kümeler çıkartılmalı.
2. Bu kümelerin diğer kümeler ile ilişkisi çıkartılmalı.

ÖRNEK: NFA otomatını DFA'ya çevir.



Bu 3 aşama ile çözüm daha iyi

1- başlangıç durumunu belirle

2- transactionları kontrol et

3- bitiş durumunu bul

$\delta(\{q_0\}, a) = \{q_1, q_2\}$

$\delta(\{q_0\}, b) = \{\phi\}$

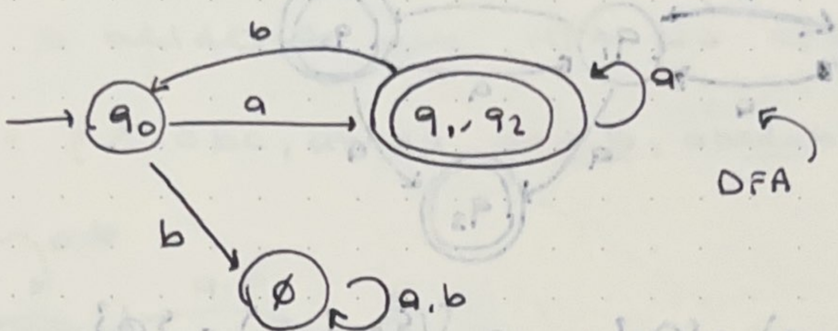
$\delta(\{q_1, q_2\}, a) = \{q_1, q_2\}$

$\delta(\{q_1, q_2\}, b) = \{q_0\}$

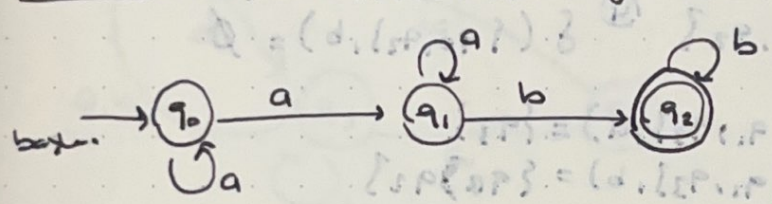
Boş kümemiz trap'tır.

Yani yine kendine dönmeli.

Son aşama final state bulma NFA'da q1 final.
O zaman belirlediğimiz kümelerden hepsinde q1 varsa
DFA'da o state final olur.



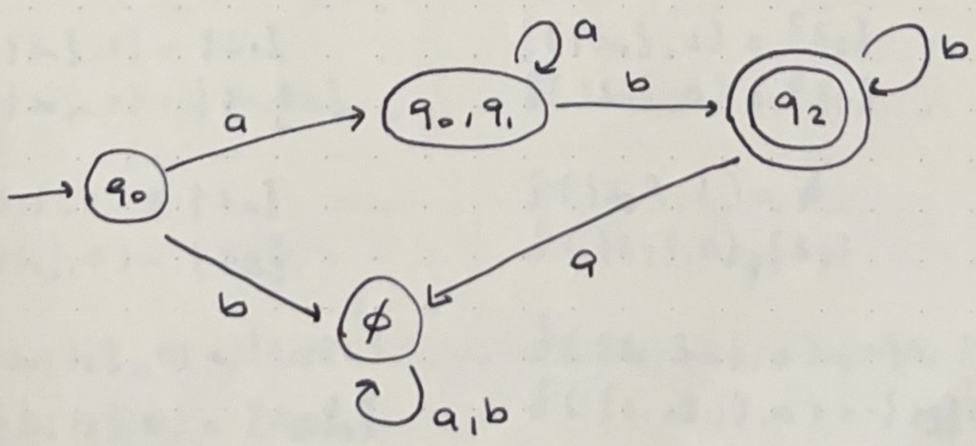
ÖRNEK: NFA otomatını DFA'ya çevir.



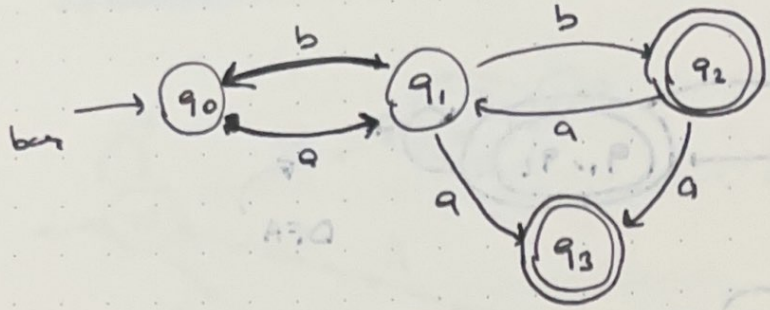
$\delta(\{q_0\}, a) = \{q_0, q_1\}$
$\delta(\{q_0\}, b) = \{\emptyset\}$

$\delta(\{q_0, q_1\}, a) = \{q_0, q_1\}$
$\delta(\{q_0, q_1\}, b) = \{q_2\}$

$\delta(\{q_2\}, a) = \{\emptyset\}$
$\delta(\{q_2\}, b) = \{q_2\}$



ÖRNEK: NFA otomatını DFA'ya çevir.



① $\delta(\{q_0\}, a) = \{q_1\}$

$\delta(\{q_0\}, b) = \{\emptyset\}$

② $\delta(\{q_1\}, a) = \{q_3\}$

$\delta(\{q_1\}, b) = \{q_0, q_2\}$

③ $\delta(\{q_3\}, a) = \{\emptyset\}$

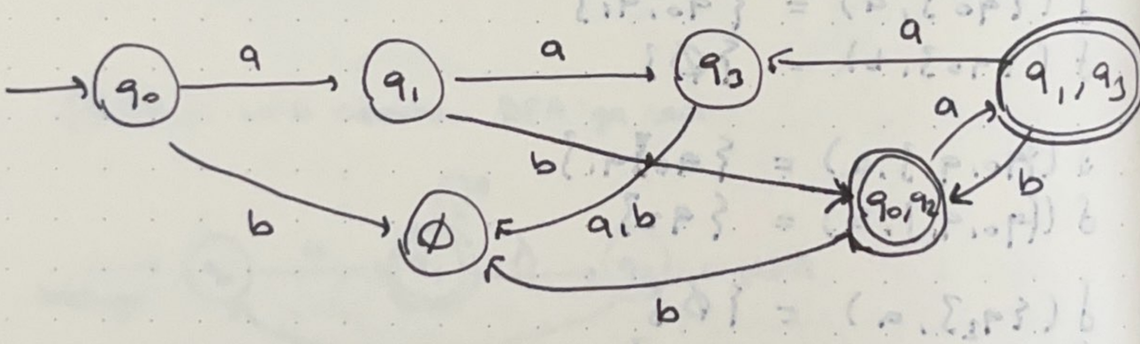
$\delta(\{q_3\}, b) = \{\emptyset\}$

④ $\delta(\{q_0, q_2\}, a) = \{q_1, q_3\}$

$\delta(\{q_0, q_2\}, b) = \emptyset$

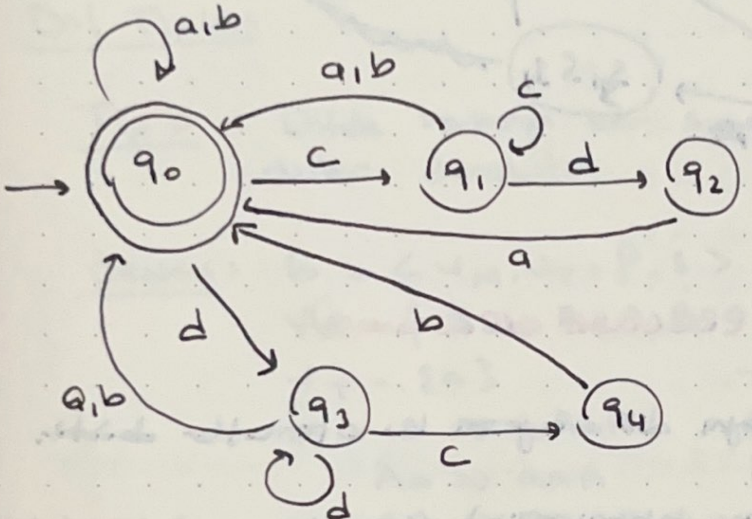
⑤ $\delta(\{q_1, q_3\}, a) = \{q_3\}$

$\delta(\{q_1, q_3\}, b) = \{q_0, q_2\}$

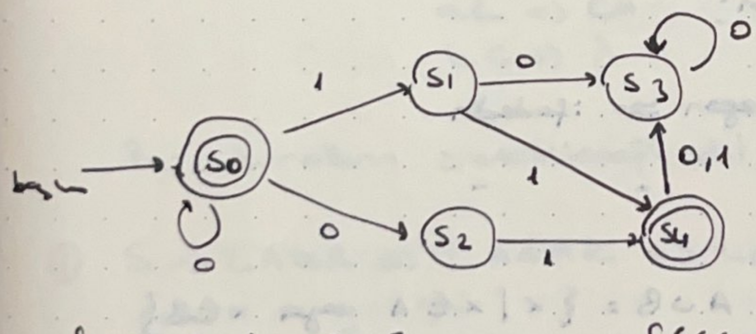


ÖRNEK: $L = \{a, b, c, d\}$ her cd alt dizisinden sonra en az bir a, her dc alt dizisinden sonra ise en az bir b bulunduğunda göre NFA çiz olası kelimeleri yaz.

$L = \{\lambda, abc, ab\underline{cda}, da\underline{dcb}, abc\underline{da}b\underline{dcb}, \dots\}$



ÖRNEK: NFA otomatını DFA'ya çevir.



$\delta(\{s_3, s_4\}, 1) = \{s_3\}$
$\delta(\{s_3, s_4\}, 0) = \{s_3\}$

$\delta(\{s_0\}, 1) = \{s_1\}$
$\delta(\{s_0\}, 0) = \{s_0, s_2\}$

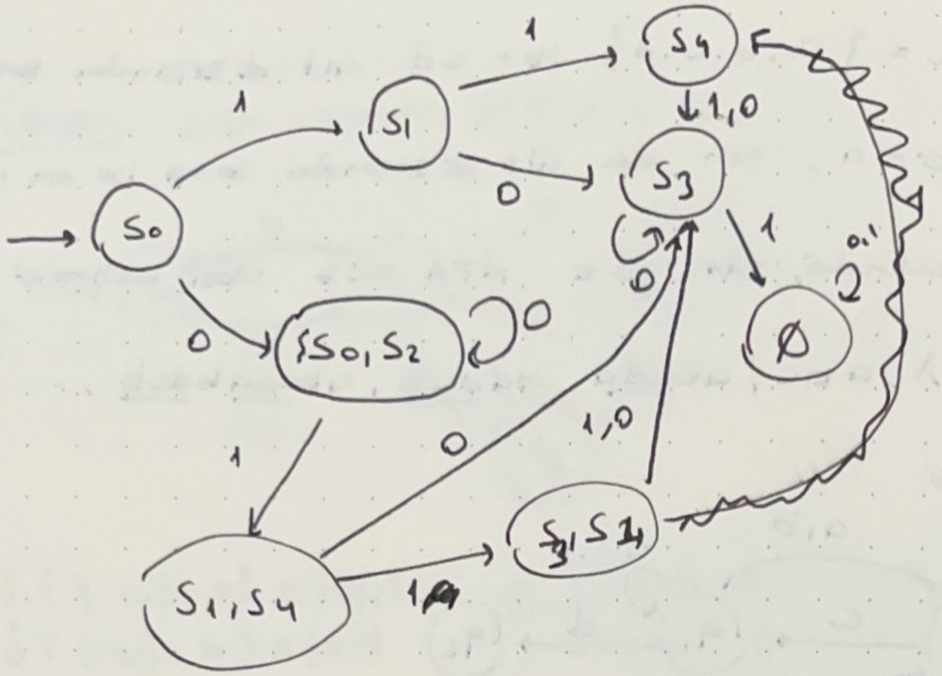
$\delta(\{s_1\}, 1) = \{s_4\}$
$\delta(\{s_1\}, 0) = \{s_3\}$

$\delta(\{s_0, s_2\}, 1) = \{s_1, s_4\}$
$\delta(\{s_0, s_2\}, 0) = \{s_0, s_2\}$

$\delta(\{s_4\}, 1) = \{s_3\}$
$\delta(\{s_4\}, 0) = \{s_3\}$

$\delta(\{s_3\}, 1) = \phi$
$\delta(\{s_3\}, 0) = \{s_3\}$

$\delta(\{s_1, s_4\}, 1) = \{s_3, s_4\}$
$\delta(\{s_1, s_4\}, 0) = \{s_3\}$



DÜZENLİ İFADELER (REGULAR EXPRESSION)

- Eğer otomatlar aynı dili anlıyorsa bu otomatlar denktir.

- Kararlı olmayan (non deterministik) otomatların, denk old. kararlı (deterministik) otomata her zaman vardır.

- A ve B diller olsun:

  1. ϕ simgesi düzgün bir ifadedir.
  2. λ " " " "
  3. Birleştirme ; $A \cup B = \{x \mid x \in A$ veya $x \in B\}$
  4. Birbirine bağlama ; $A \circ B = \{xy \mid x \in A$ ve $y \in B\}$
  5. Kleene yıldızı ; $A^* = \{x_0, x_1, x_2 \ldots x_k,\ k \geq 0$ ve her $x_i \in A\}$

- Dil tipleri burada bahsedilmeliydi, Tip 0, Tip 1, Tip 2, Tip 3

ÖRNEK: $0 \cup 01$ ve $0(0 \cup 1)^*$ ne ifade eder?

$0 \cup 01 \rightarrow$ 0 içeren dizgi ile 01 içeren dizginin birleşimi

$0(0 \cup 1)^* \rightarrow$ 0 ile başlayan herhangi bir dizgi

Dil Tipleri

Tip 0 : Dilde herhangi bir sınırlama yoktur. Özyinelemeli diller denebilir.

ÖRNEK: $G = \langle V_N, V_T, P, S \rangle$

$V_N = \{S, L, R, A, B, C\}$ → sonlu terime

$V_T = \{a\}$ → uç simgeler

$P = S \Rightarrow LAaR$ S → başlangıç değişkeni

$Aa \Rightarrow aaA$ P → türetme

$AR \Rightarrow BR \mid C$

$aB \Rightarrow Ba$

$LB \Rightarrow LA$

$aC \Rightarrow Ca$

$LC \Rightarrow \lambda$

Bu kuralların üretebileceği dil:

① $S \Rightarrow LAaR \Rightarrow LaaAR \Rightarrow LaaC \Rightarrow LaCa \Rightarrow LCaa$

$\Rightarrow aa$ Bu bir örnek bunun gibi başka kelimelerde türeyebilir.

② $S \Rightarrow LAaR \Rightarrow LaaAR \Rightarrow LaaBR \Rightarrow LaBaR \Rightarrow LBaaR$

$\Rightarrow LAaaR \Rightarrow LaaAaR \Rightarrow LaaaaAR \Rightarrow LaaaaC \Rightarrow LaaaCa$

$\Rightarrow LaaCaa \Rightarrow LaCaaa \Rightarrow LCaaaa \Rightarrow$ ~~aaaa~~ $aaaa$

Üretilen kelimelere bakıldığında $L(G) = \{a^i \mid i = 2^n, n \geq 1\}$

Tip 1: Bağlam-bağımlı bir dildir. Buna özyinelemeli bir dildir. Tip 0'da fazla olarak kısıtlaması vardır.

$\alpha \Rightarrow \beta : \alpha \in V^{T} \quad \beta \in V, \; |\alpha| \leq |\beta|$

fark

ÖRNEK: $G = \langle V_N, V_T, P, S \rangle$

$V_N = \{S, A, B\}$

$V_T = \{a, b, c\}$

$P:$ $S \Rightarrow aSAB$

$S \Rightarrow aAB$

$BA \Rightarrow AB$

$aA \Rightarrow ab$

$bA \Rightarrow bb$

$bB \Rightarrow bc$

$cB \Rightarrow cc$

① $S \Rightarrow aSAB \Rightarrow aaABAB \Rightarrow aabBAB \Rightarrow \cancel{aabcAB}$

$\Rightarrow aabABB \Rightarrow aabbBB \Rightarrow aabbcB \Rightarrow aabbcc$

② $S \Rightarrow aAB \Rightarrow abB \Rightarrow abc$

$L(G) = \{ a^n b^n c^n \mid n \geq 1 \}$

Tip 2: Bağlamdan-bağımsız dildir. Yeniden yazma kurallarının sol tarafında tek bir değişken (A) vardır.

$A = B: \; A \in V_N \quad B \in V^*$

Programlama dilleri ve yazılım sistemlerinin bir çoğunda bu modell kullanılır.

ÖRNEK: $G = \langle V_N, V_T, P, S \rangle$

$V_N = \{S, A, B\}$

$V_T = \{a, b\}$

$P: S \Rightarrow aB \mid bA$

$A \Rightarrow a \mid aS \mid bAA$

$B \Rightarrow b \mid bS \mid aBB$

① $S \Rightarrow aB \Rightarrow aaBB \Rightarrow aabSB \Rightarrow aabbAB \Rightarrow aabbaB$

$\Rightarrow aabbab$

② $S \Rightarrow bA \Rightarrow baS \Rightarrow babA \Rightarrow baba$

③ $S \Rightarrow aB \Rightarrow abS \Rightarrow abbA \Rightarrow abbaS \Rightarrow abbaaB$

$\Rightarrow abbaab$

L(G), $\{a, b\}$ alfabesinde eşit sayıda a ve b içeren dizgiler kümesi olur.

Tip 3: Düzgün ifadedir. (Regular)

ÖRNEK: $G = \langle V_N, V_T, P, S \rangle$

$V_N = \{S, A, B\}$

$V_T = \{0, 1\}$

$P: S \Rightarrow 0S \mid 0A \mid 0 \mid \lambda$

$A \Rightarrow 0B$

$B \Rightarrow 1S$

① $S \Rightarrow 0S \Rightarrow 00A \Rightarrow 000B \Rightarrow 0001S \Rightarrow 00010$

② $S \Rightarrow 0A \Rightarrow 00B \Rightarrow 001S \Rightarrow 001$

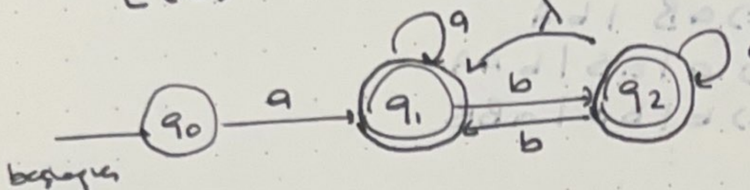
③ $S \Rightarrow 0$

④ $S \Rightarrow 0S \Rightarrow 00$

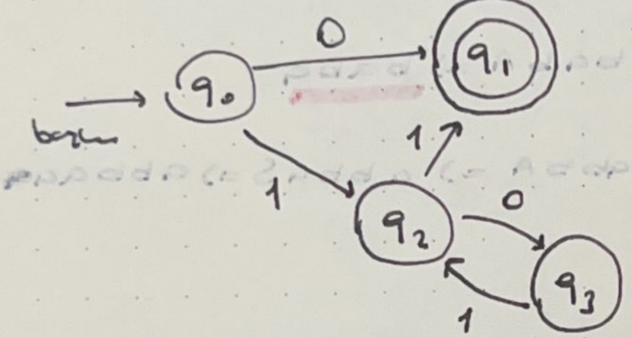
L(G) kümesi $\{0, 1\}$ alfabesinde her 1 ifadesinden önce en az iki sıfır (00) bulunan kümedir.

ÖRNEK: $L = aa^* (bc^*)^*$

$L(6)$ = a, aaaa, abcccc, abcbcbc ---



ÖRNEK: $0 \cup 1(01)^* 1$ ifadesini tanıyan otomatayı çiz



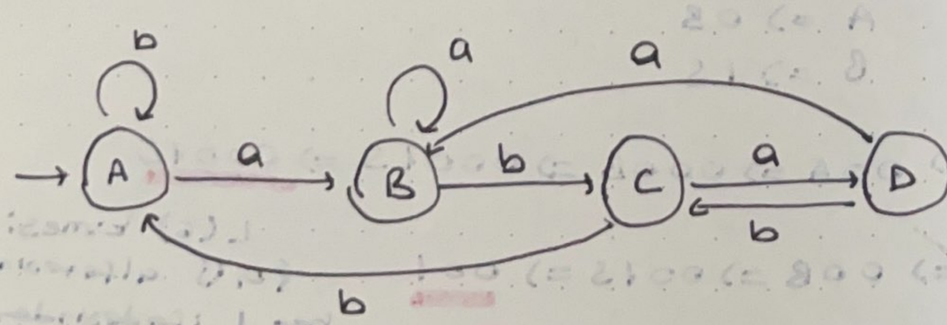
ÖRNEK: $V_N = \{A, B, C, D\}$ $V_T = \{a, b\}$

S => A başlangıç durumu

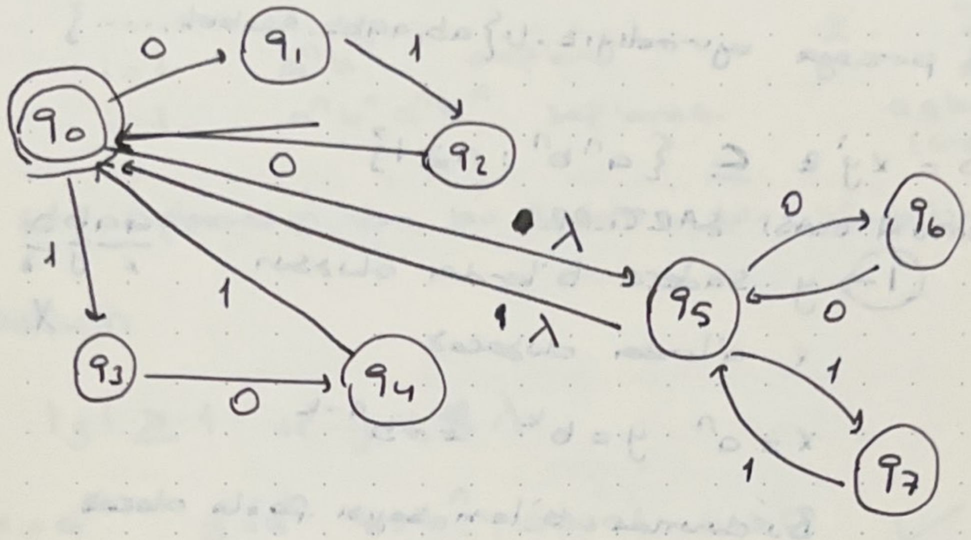
D bitiş " "

Tabloya göre DFA'yı çiz

	a	b
A	B	A
B	B	C
C	D	A
D	B	C



ÖRNEK: $[(010) \cup (01) \cup (00 \cup 11)^*]^*$ NFA'yı çiz



PUMPING LEMMA

* Bir dilin düzenli ifade olup olmadığını anlamak için kullanırız.
* Dilin düzenli ifade olup olmadığını anlamak için;

$L = w$ dili için
$w = xyz$ şeklinde açılımı olsun.

Şu koşulların sağlanması beklenir.

① $|y| \geq 1$

② $|xy| \leq p$

③ $\forall\ i \geq 0$ için $xy^iz \in L$

* Bu 3 şart sağlanıyorsa ifade düzenlidir deriz.

* Yani dilden üretilen bir kelime üç parçaya bölündüğünde ortadaki parça boş olmamalı istenildiği kadar tekrarlanabilmeli

ÖRNEK: $\{a^n b^n : n \geq 1\}$ dili düzenli old. varsayalım

bu durumda tüm cizgilerimizi x, y, z olacak şekilde

3 parçaya ayırmalıyız. $L = \{ab, aabb, aaabbb, \ldots\}$

$S = xy^iz \subseteq \{a^n b^n : n \geq 1\}$

3. DURUMUN OLASI ŞARTLARI

1- y sadece b'lerden oluşsun

x a'lardan oluşacak

$\underbrace{aa}_{x}\underbrace{b}_{y}\underbrace{b}_{z}$ X

$x = a^n \quad y = b^k \quad z = b^{n-k}$

Bu durumda b'lerin sayısı fazla olacak

$S = a^n b^k b^{n-k}$

$= a^n b^n$

çünkü xy^iz $i=1$ $a^n b^n$ sağlar

$i=2$ $a^n b^n b^n$ sağlamaz

gerisine bakmaya gerek yok.

$\underbrace{aa}_{x}\underbrace{bb}_{y}\underbrace{b}_{z}$ $\underset{a}{2} \neq \underset{b}{3}$

2. y sadece a'lardan oluşsun

z b'lerden oluşsun

$\underbrace{a}_{x}\underbrace{a}_{y}\underbrace{bb}_{z}$

$x = a^{n-k} \quad y = a^k \quad z = b^n \quad 1 \leq k \leq n$

$S = a^{n-k} a^k b^n = a^n b^n$ X

$i=1$ xyz sağlar

$i=2$ xy^2z sağlamaz

$\underbrace{a}_{x}\underbrace{aa}_{y^2}\underbrace{bb}_{z}$ $\underset{a}{3} \neq \underset{b}{2}$

3 = 3 ama aabbaabb (sıra ahed)

③ y, a ve b'lerden oluşabilir.

$x = \lambda \quad y = a^n b^n \quad z = \lambda$

aabb or
y

aabb
x y z

aababb
(sıra olmadığı)
3 = 3 ama

$i = 1 \quad a^n b^n$ sağlar

$i = 2 \quad a^n b^n a^n b^n$ sağlamaz.

3 şart sağlanmadığı için, bu düzensiz bir ifadedir.

1. Durum

$|y| \geq 1$ ve $y \neq \lambda$

$x = a^{n-k} \quad y = a^k \quad z = b^n$ olsun $k > 0$ ✓

o zaman şart sağlanır.

2. Durum

DÜZENLİ DEĞİLDİR.

$|xy| \leq p$

$x = a^{n-k} \quad y = a^k \quad z = b^n$ olsun ✓

$|xy| = n \quad p = 2n \quad |xy| \leq p$ sağlanır

Bağlamdan Bağımsız Diller için Pumping Lemma

* Bir dilin içerik bağımsız grammer (CFG) ile gösterilemeyeceğini ispatlar

- Yani CFG olmadığı ispatlanabilir ama olduğu ispatlanamaz.

~~CFG Nedir?~~

* Bazı bağlamdan bağımsız (CFG) diller, düzenli değildir. Fakat tüm düzenli diller bağlamdan bağımsız (CFG)'dir.

* CFG Nedir?
  * Bilgisayar biliminde dil tasarımı sırasında kullanılan grammer tipidir. Bağlamdan bağımsız = C(
  * Bir dilin kurallarını tanımlamak için kullanılır.

ÖRNEK: $S \Rightarrow a$

* Bir dili içerikten bağımsız yapan, o dilin bir non deterministik push down automata tarafından üretilebilir olmasıdır.

* Bilinen çoğu prog. dili CFG'dir.
* Temel olarak CFG 4 özellik (temel) içerir.

sırası değişir

$G = (V_T, V_N, P, S)$

V_T: uç simge; V_N: sonlu küme; P: türetim fonk.; S: başlangıç

$S \Rightarrow aSb \mid ab$

$G = (V_N, V_T, P, S) = (\{S\}, \{a,b\}, \{S \to aSb|a$

* Şayet pumping lemma önsavını geçemiyorsa CFG değildir. Ancak geçmesi CFG old. göstermez.

$s = uvxyz$

① $\forall i \geq 0$

② $|vy| > 0 \quad v \neq \lambda \mid y \neq \lambda$

③ $|vxy| \leq p$

Bağlamdan bağımsız ~~[illegible]~~ olması için bu 3 şart sağlanmalıdır. ~~[illegible]~~

ÖRNEK: $L = \{a^i b^i c^i \mid i > 0\}$ pumping lemma ile

yukarıdaki dilin CFG olmadığını ispatla

1. kombinasyon

$v = a \quad x = b \quad y = c \quad u, z = \lambda$ olsun

$s = a^i b^i c^i \quad i > 0$ olsun

$i = 1 \quad abc$

$i = 2 \quad a^2 b^2 c^2$ olmalı fakat $i = 2$ olduğunda

$s = u v^i x y^i z$ 'den $aabcc$ olur.

bu durum çelişki yaratır.

2. kombinasyon

$u = a \quad v = b \quad y = c \quad x, z = \lambda$ olsun

$i = 1 \quad abc$

$i = 2 \quad a^2 b^2 c^2$ olmalı fakat $abbcc$ olur

bu durumda çelişki yaratır.

3. kombinasyon

$v = a \quad y = b \quad z = c \quad u, x = \lambda$

$i = 1 \quad abc$

$i = 2 \quad a^2 b^2 c^2$ olmalı fakat $aabbc$ olur

bu durumda çelişki yaratır.

O zaman CFG değildir. Yani bağlamdan bağımsız olmadığı ispatlanmıştır.

CHOMSKY NORMAL FORMU (CNF)

* Şayet bir CFG aşağıdaki kurallara uyuyorsa bu dil bilgisine CNF'ye uygun denilebilir.

A → BC veya
A → a veya
S → λ şeklinde kurallara uyuluyorsa

* CNF kurallarına uygun olan her dil CFG'dir.

* CNF'ye uygun bir dil ile parçalanan kelime ve oluşan ayrıştırma ağacında belirsizlik bulunmaz. Her zaman tek bir ağaç çizer.

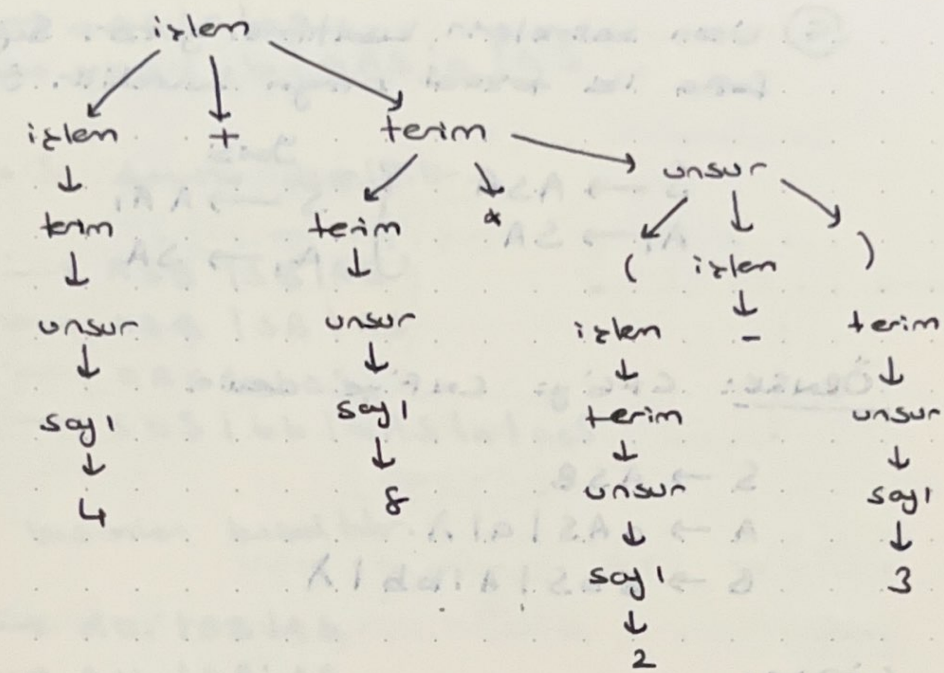
* Parçalama / Ayrıştırma Ağacı Nedir?

  * Verilen bir dil bilgisine (grammer) göre verilen cümlenin / kelimenin nasıl parçalandığını şekilsel olarak gösteren ağaçtır.

ÖRNEK: BNF yapısında verilmiş dilin ayrıştırma ağacını çizin.

<dil> ::= <işlem>
<işlem> ::= <işlem> + <term> | <işlem> - <term> | <term
<term> ::= <term> * <unsur> | <term> / <unsur> | <unsu
<unsur> ::= sayı | (<işlem>)
<sayı> ::= 1|2|3 ...|9|0

4 + 8 * (2-3) yapısının ağacı →



* Chomsky'ye geri dönelim :
* Chomsky normal form kuralları;

① Başlangıç (S) sağ tarafta bulunmasına karşılık yeni bir başlangıç ~~olmaya~~ terminal olmayan eklenir.

$S_0 \longrightarrow S$ eklenir başa

② Landa (λ) veya Epsilon (ε) varsa temizlenir.

③ Tek terimliler kaldırılır. Örn;

$S_0 \longrightarrow S$
$S \longrightarrow ASA \mid aB \mid AS \mid SA \mid S \mid a$
$A \longrightarrow B \mid S$
$B \longrightarrow b$

tek terimler silinir.

④ Terminal olmayan teriminde terminal olan varsa bu ifadeyi basitleştirmek için ilave bir terminal eklenir. Örn;

$S \longrightarrow aB$
$U \longrightarrow a$

Yenisi
$S \longrightarrow UB$
$U \longrightarrow a$

⑤ Uzun kelimelerin kısaltılması gerekir. Sağ tarafta en fazla iki terminal olmayan bulunabilir. Örn;

$S \rightarrow ASA$ ⎫ Yeni:
$A_1 \rightarrow SA$ ⎭ $S \rightarrow AA_1$
$A_1 \rightarrow SA$

ÖRNEK: CFG'yi CNF'ye çevir.

$S \rightarrow ASB$
$A \rightarrow aAS \mid a \mid \lambda$
$B \rightarrow SbS \mid A \mid bb \mid \lambda$

Çözüm:

① Başlangıç terimi S_0 eklenir.

$S_0 \rightarrow S$

② λ'lar temizlenir. $A \rightarrow \lambda$. A değişkeninin kullanıldığı yerlerde eklenme olur.

$S_0 \rightarrow S$ $\qquad S \rightarrow ASB \mid SB$
$A \rightarrow aAS \mid a \mid aS$
$B \rightarrow SbS \mid A \mid bb \mid \lambda$

$B \rightarrow \lambda$ temizlenir.

$S_0 \rightarrow S$
$S \rightarrow ASB \mid SB \mid AS \mid S$
$A \rightarrow aAS \mid a \mid aS$
$B \rightarrow SbS \mid A \mid bb$

③ Tez ~~devamlı~~ terimli durumlar silinir.

$S_0 \rightarrow S$
$S \rightarrow ASB \mid SB \mid AS$

$A \rightarrow aAS \mid a \mid aS$
$B \rightarrow SbS \mid bb \mid aAS \mid a \mid aS$

(4) $S_0 \rightarrow S$ durumu düzeltilir.

$S_0 \rightarrow ASB \mid SB \mid AS$
$S \rightarrow ASB \mid SB \mid AS$
$A \rightarrow aAS \mid a \mid aS$
$B \rightarrow SbS \mid bb \mid aAS \mid a \mid aS$

(5) Uzun kısımlar kısaltılır.

$S_0 \rightarrow AU_1 \mid SB \mid AS$
$S \rightarrow AU_2 \mid SB \mid AS$
$A \rightarrow Y_1U_3 \mid a \mid Y_1S$
$B \rightarrow SU_4 \mid Y_2Y_2 \mid Y_1U_5 \mid a \mid Y_1S$
$U_1 \rightarrow SB$ → S_0 için
$U_2 \rightarrow SB$ → S için
$U_3 \rightarrow AS$ → A için
$U_4 \rightarrow Y_2S$ → B için
$U_5 \rightarrow AS$ → B için
$\left.\begin{array}{l} Y_1 \rightarrow a \\ Y_2 \rightarrow b \end{array}\right\}$ uç terimler

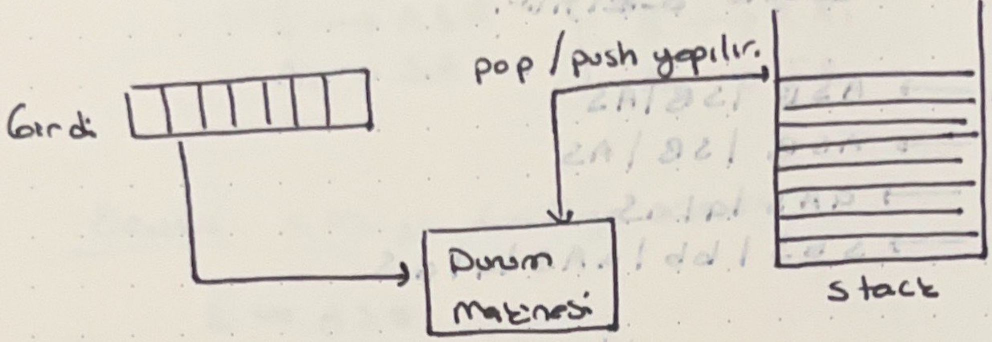
PUSHDOWN AUTOMATA (PDA)

- Yığıtlı otomatadır. Hafızaya sahiptir.
- Deterministik veya non-deterministik olabilir.
- Sonlu otomattan farkı yığın (stack) kullanmasıdır.
- PDA'yı tanımlamak için 6 bilgi gerekir.

$M = (Q, \Sigma, \Gamma, \delta, q_0, F)$

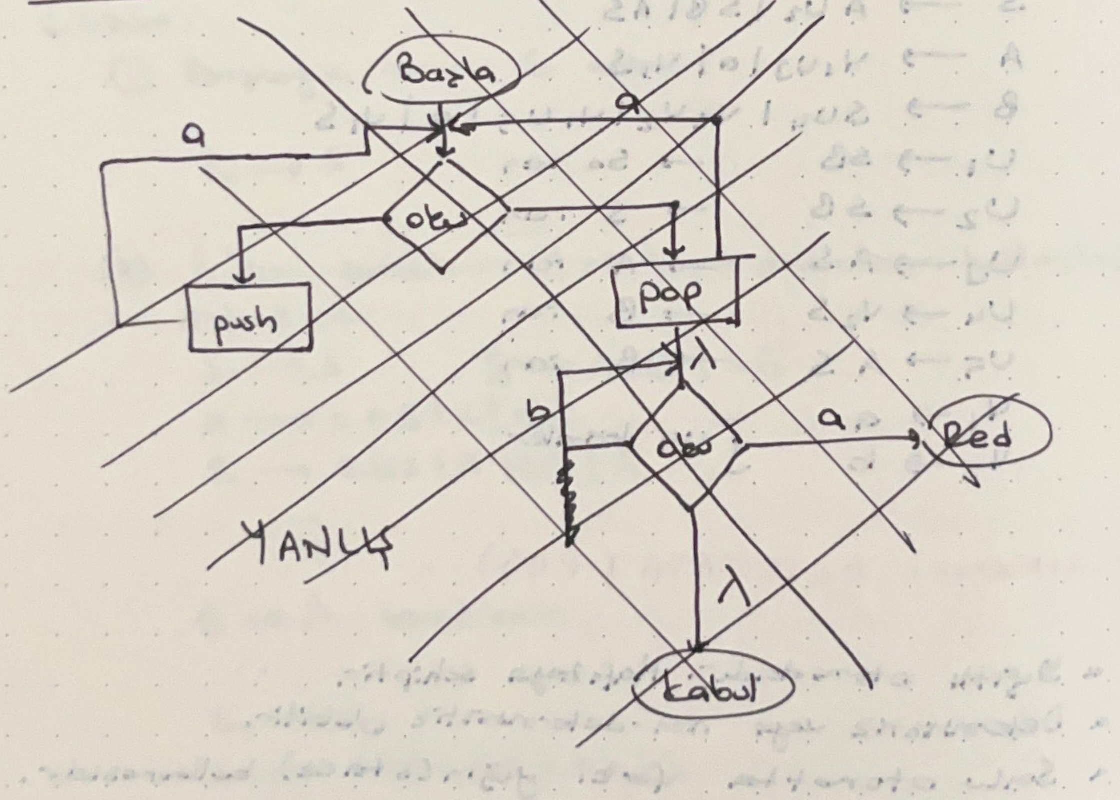
Q: durumlar
Σ: alfebenin kümesi, λ ile de gösterilir
Γ: yığın alfebesi
δ: durumlar arası geçişlerin kümesi
q_0: başlangıç durumu (S_0 da olabilir)
F: Bitiş durumu

* LIFO mantığı ile çalışır.
* Stack PDA'nın düzenli olmayan dillerin bazılarının anlaşılmasını sağlar.



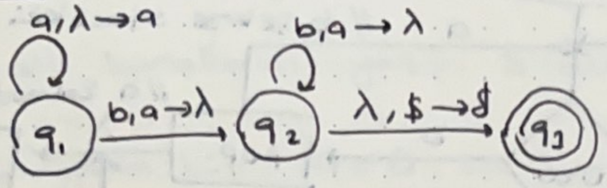
* CFG'ler PDA tarafından tanınır.

~~ÖRNEK: $L = \{a^n b^m \mid n \geq 0, m > n\}$ PDA'da tanımla~~

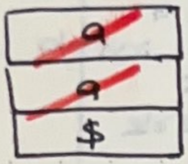


* Pumping lemma örneğinde $a^n b^n$ $n \geq 1$ ifadesinin düzenli olmadığını ispatladık.
* Bu dil otomata tarafından kabul edilmiyordu.
* Çünkü sözde a'ların kaç tane old. ve buna göre kaç b olacağı hesaplayamıyordu.
* Bu sorunu hafıza ile çözebiliriz. Yani PDA kullanırız.

ÖRNEK: $L(M) = \{a^n b^n : n \geq 1\}$ PDA'sını çiz.



$ veya z stack'in bitiş noktasını göster.

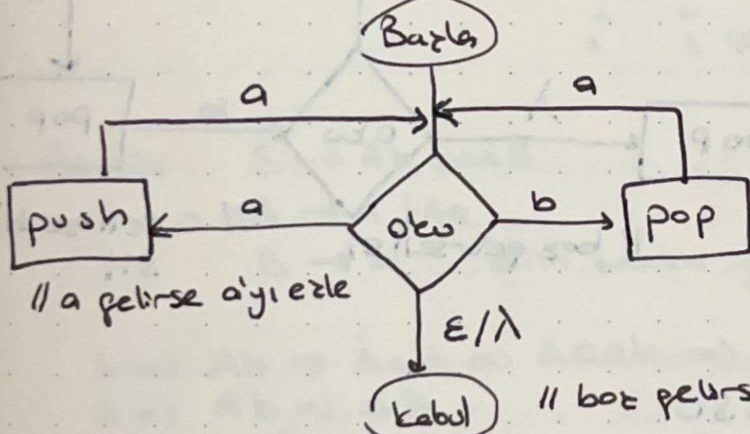


stack

n = 2 olsun

aabb

a ezledi sonra bir daha a ezledi b'yi gördü b, a → λ, a'yı sildi bir şey yapmadı yine b'yi gördü a'yı sildi bir şey yazmadı. λ geldi $ gördü bitti.



// b gelirse a'yı sil

// a gelirse a'yı ezle

// boş gelirse bitir.

ÖRNEK: $L(M) = \{a^n b^m c^k$ $n \geq 1$ $m \geq 2$ $k \geq 1$ $m = n + k\}$ dilini ve PDA'sını tanımla.

Diyelim ki; $n = 1$ $k = 1$ $m = 2$ → abbc

$n = 2$ $k = 3$ $m = 5$ → aabbbbbccc

O zaman; $G = \langle V_N, V_T, P, S \rangle$

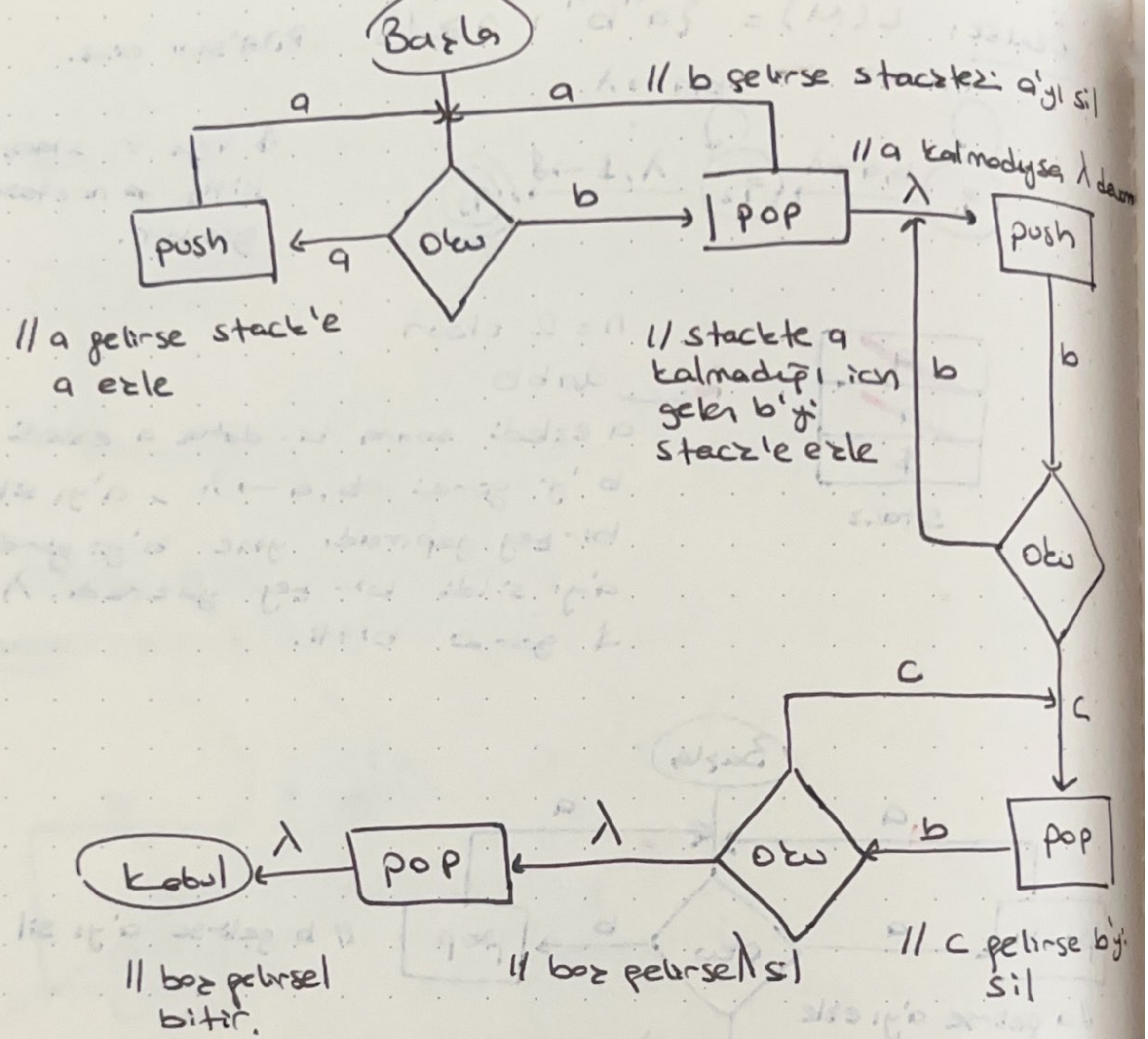
⇒ $V_N = \{S, A, B\}$

⇒ $V_T = \{a, b, c\}$

⇒ $P : S \rightarrow AB$

$A \rightarrow aAb \mid ab$

$B \rightarrow bBc \mid bc$



ÖRNEK: $S \rightarrow 01B0 \mid 0S0$
$B \rightarrow 1B \mid 1$ olabilecek dilin kuralını yaz. $L(G) = ?$

$S \Rightarrow 01B0 \Rightarrow 011B0 \Rightarrow \underline{01110}$

$S \Rightarrow 0S0 \Rightarrow 00S00 \Rightarrow 0001B000 \Rightarrow \underline{00011000}$

$S \Rightarrow 01B0 \Rightarrow 011B0 \Rightarrow 0111B0 \Rightarrow \underline{011110}$

$S \Rightarrow 0S0 \Rightarrow 00S00 \Rightarrow 0001B000 \Rightarrow 00011B000$
$\Rightarrow \underline{000111000}$

Her seferinde 0'ların sayısı eşit ve 1'ler ortada yer alıyor. 1'ler her zaman en az 2 tane oluyor. O zaman;

$$L(G) = \{ 0^n 1^m 0^n \mid n \geq 1 , m \geq 2 \}$$

ÖRNEK: $L = \{ (ab)^n c^m \mid 0 \le m,n \mid 3n \le m \}$

dil kurallarını yaz. S değeri ve P değerlerini.

Diyelim ki $m = 0 \quad n = 0$ olsun : λ

$m = 7 \quad n = 2$ olsun $abab\underbrace{mmmmmmm}_{c}$

$m = 5 \quad n = 0$ olsun $ccccc$

$m = 4 \quad n = 1$ olsun $abcccc$

$m = 8 \quad n = 2$ olsun $ababcccccccc$

$m = 10 \quad n = 3$ olsun $abababccccccc$
ccc

$$S \rightarrow \lambda \mid Sc \mid abSccc$$

$3n \le m$
$\downarrow \quad \downarrow$
$1 \quad 3$ (c en az 3 tane olmalı)

ÖRNEK: $S \rightarrow Ab \mid aaB$
$A \rightarrow a \mid Aa$
$B \rightarrow b$ dilin kuralını yaz. $L(G) = ?$

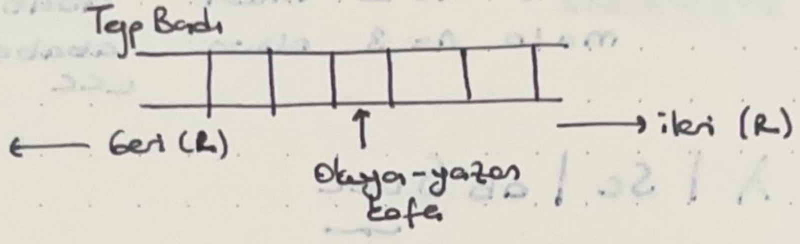
$S \Rightarrow Ab \Rightarrow Aab \Rightarrow Aaab \Rightarrow aaab$
$S \Rightarrow Ab \Rightarrow ab$
$S \Rightarrow aaB \Rightarrow aab$
$S \Rightarrow Ab \Rightarrow Aab \Rightarrow aab$

Her zaman b en sonda ve tek. a ise en az bir tane var.
O zaman;
$L(G) = \{ a^n b, \ n \ge 1 \}$

TURING MAKİNESİ:

* Bir Turing makinesi her bir adımda sonlu sayıdaki durumlardan birinde olan denetleme birimi ve her iki yönde sonsuz sayıda hücrelere bölünmüş teypten oluşur.

* Basitçe bir kafa (head) ve teyp bandından (tape) oluşur.



* Bilgisayarların yapabileceği tüm hesaplamaları yapabilecek kapasitededir.
* PDA'ya benzer bir belleğe sahiptir.
* PDA'dan farklı ileri-geri yapabilir. Mevcut konumdaki değeri silip onun yerine yenisini yazabilir.

* Turing makinesi aynı zamanda sonlu durumda sonsuz hafızaya sahip otomata olarak tanımlanır.

* Turing makinesi $(Q, \Sigma, \Gamma, \delta, q_0, B, F)$ yedilisi ile tanımlanır.

Q = Sonlu durum kümesi
Σ = sonlu sayıdaki giriş alfabesindeki semboller
Γ = şerit üzerindeki sayılar / simgeler
B = bitiş sembolü veya ◊ ile de gösterilir.
q_0 = başlangıç durumu
F = uç durumları içeren sonlu durum

$$\delta = Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$$

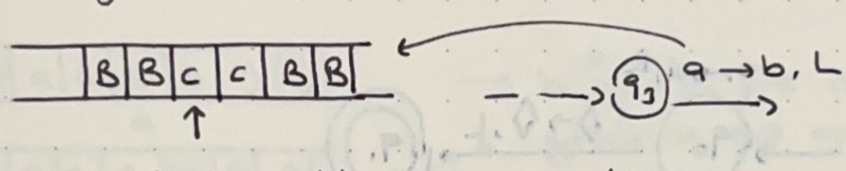
kafanın şerit üzerinde hangi yöne gideceğini belirler

Halt

* Bitme veya çalışmama durumu 2 türlüdür.

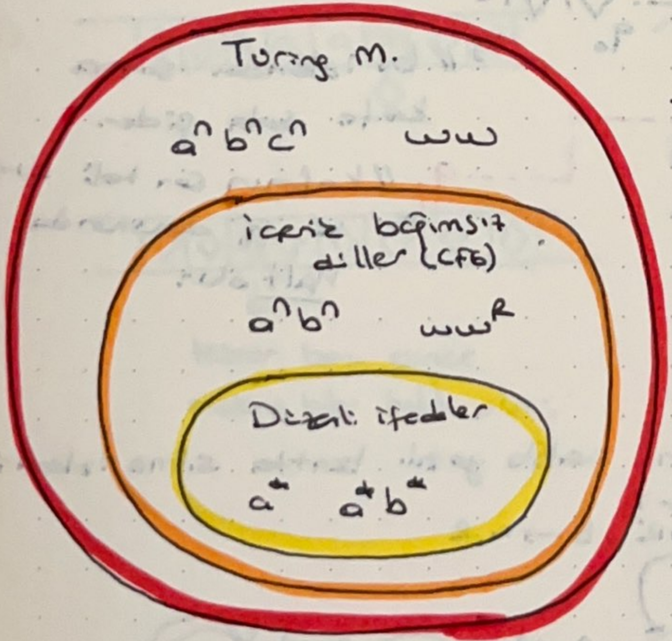
① Gidecek başka durum kalmamıştır. ---→ q_5

② Kafanın bulunduğu yerde istenen işlemi yapacak eleman yoktur.



Kafanın old. yerde a yoktur.
O zaman yine halt olur.

Turing makinesinin kabul ettiği diller.

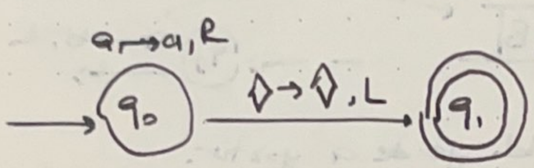


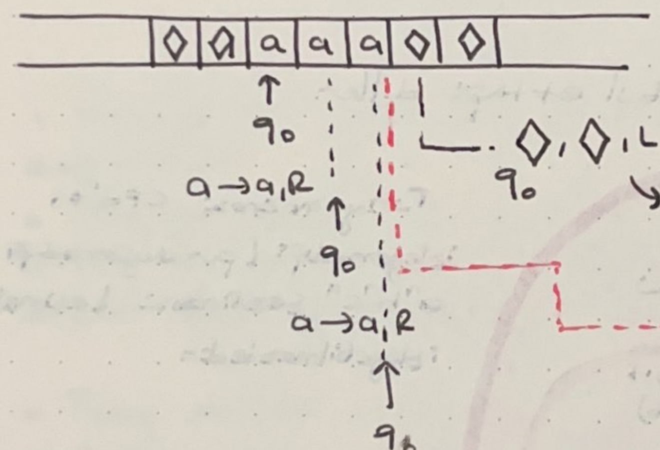
Turing makinesi CFG'nin işleyemediği (parçalayamadığı) $a^n b^n c^n$ şeklindeki kelimeleride işleyebilmektedir.

ÖRNEK: a^n dizeli ifadesini Turing makinesi ile göster.

aaa şeklinde ifadenin makine tarafından kabul edilip edilmediğine bak.

$$M = \{\underbrace{\{q_0, q_1\}}_{Q}, \underbrace{\{a\}}_{\Sigma}, \underbrace{\{a, x\}}_{\Gamma}, \underbrace{\{q_0\ a \rightarrow a, R,\ q_0\ x \rightarrow x, L\}}_{\delta}, q_0, x, q_1\}$$



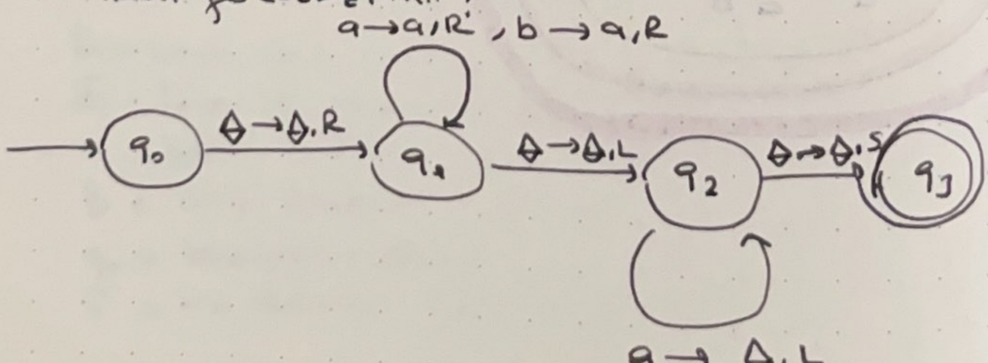


→ Bantta 3 a old. kabul edilsin

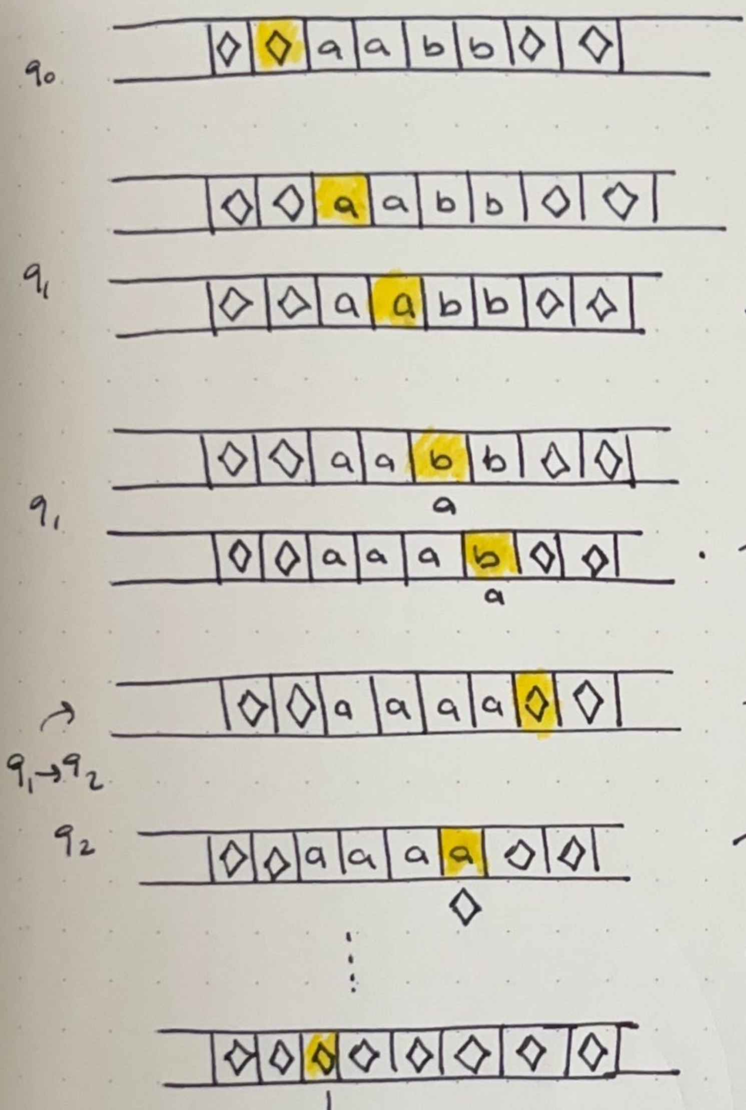
// bu işlemden sonra kafa sola gider.

// kafanın son hali burada ve q_1 durumunda kabul <u>halt</u> olur.

ÖRNEK: Aşağıdaki makinenin aabb yazılı bantta silme işlemini nasıl gerçekleştirilir?



Not: Son resimde olan kutular kafanın en bulunduğu kutudur.



→ bantta aabb old. kabul edelim.

◇ old. için sağ sarılır.

→ okunan a görüldüğü yerde a değeri kalır ve sağ gidilir.

→ b okunan görülen yere a yazılır ve sağ gidilir.

→ durum q_2'ye geçer. ◇ görülünce sola gider.

→ a okununca ◇ yazar ve sola gider

Bu işlem tüm a'lar için devam eder ve sonuç olarak tümü silinmiş boş kalan bellek kalır.

tezren boş görünce makine biter halt olur.