# Bilgisayar Bilimlerinde Hesaplama Kuramı

Hüseyin Hışıl

Bilgisayar Mühendisliği Bölümü
Mühendislik Fakültesi
Yaşar Üniversitesi

8 Mart 2012 / İzmir

# Bilgisayar için en basit model

# Alfabe, sözcük, kurallı dil (formal language)

## Tanım

- $\Sigma$ alfabesi : Sonlu sayıda elemana (karaktere) sahip küme
- Sözcük : $\varepsilon$ (boş sözcük) de dahil olmak üzere $\Sigma$'dan sıralı olarak seçilen sonlu sayıda eleman
- Kurallı dil : $\Sigma$ üzerine tanımlı sözcüklerden oluşan küme
- $\Sigma^*$ : $\Sigma$ üzerine tanımlı tüm sözcüklerin oluşturduğu kurallı dil
- $\Sigma^n$ : $\Sigma$ üzerine tanımlı $n$ tane karakterden oluşan tüm sözcüklerin oluşturduğu kurallı dil

## Örnek

$\Sigma = \{a, b\}$ olsun. $\Sigma$ üzerine $a$ ile başlayıp en fazla üç karakterden oluşan tüm sözcükleri içeren $L_1$ dili tanımlanmıştır. $L_1$ dilini belirtiniz.

$$L_1 = \{a, aa, ab, aaa, aab, aba, abb\}.$$

# Yeni dillerin türetilmesi

## Tanım

$\Sigma$ bir alfabe olsun. $\Sigma$ üzerine tanımlanmış $L$ ve $K$ dilleri için;

- Birbirine bağlama :

$$LK = \{xy \mid x \in L \text{ and } y \in K\}.$$

- Birleştirme :

$$L \cup K = \{x \mid x \in L \text{ or } x \in K\}.$$

- Kleene tamamlaması :

$$L^* = \{x \mid x, L \text{ dilinden birbiriyle bağlanmış sonlu sayıda sözcük}\}.$$

# Yeni dillerin türetilmesi

## Örnek

$L_1 = \{♡♡, ♣, ♣♡\}$, $L_2 = \{♠, ♣♠\}$ olsun.

1. $L_1 L_2 = ?$
2. $L_1 \cup L_2 = ?$
3. $(L_1 \cup L_2)^* = ?$

# Yeni dillerin türetilmesi

## Örnek

$L_1 = \{♡♡, ♣, ♣♡\}$, $L_2 = \{♠, ♣♠\}$ olsun.

1. $L_1L_2 = ?$
2. $L_1 \cup L_2 = ?$
3. $(L_1 \cup L_2)^* = ?$

1. $L_1L_2 = \{♡♡♠, ♡♡♣♠, ♣♠, ♣♣♠, ♣♡♠, ♣♡♣♠\}$.

# Yeni dillerin türetilmesi

## Örnek

$L_1 = \{♡♡, ♣, ♣♡\}$, $L_2 = \{♠, ♣♠\}$ olsun.

1. $L_1 L_2 = ?$
2. $L_1 \cup L_2 = ?$
3. $(L_1 \cup L_2)^* = ?$

1. $L_1 L_2 = \{♡♡♠, ♡♡♣♠, ♣♠, ♣♣♠, ♣♡♠, ♣♡♣♠\}$.
2. $L_1 \cup L_2 = \{♡♡, ♣, ♣♡, ♠, ♣♠\}$.

# Yeni dillerin türetilmesi

**Örnek**

$L_1 = \{\heartsuit\heartsuit, \clubsuit, \clubsuit\heartsuit\}$, $L_2 = \{\spadesuit, \clubsuit\spadesuit\}$ olsun.

1. $L_1 L_2 = ?$
2. $L_1 \cup L_2 = ?$
3. $(L_1 \cup L_2)^* = ?$

1. $L_1 L_2 = \{\heartsuit\heartsuit\spadesuit, \heartsuit\heartsuit\clubsuit\spadesuit, \clubsuit\spadesuit, \clubsuit\clubsuit\spadesuit, \clubsuit\heartsuit\spadesuit, \clubsuit\heartsuit\clubsuit\spadesuit\}$.
2. $L_1 \cup L_2 = \{\heartsuit\heartsuit, \clubsuit, \clubsuit\heartsuit, \spadesuit, \clubsuit\spadesuit\}$.
3. $(L_1 \cup L_2)^* =$

   $\{x \mid x, L_1 \cup L_2$ dilinden birbiriyle bağlanmış sonlu sayıda sözcük$\}$.

# Hesaplayıcı (Finite state machine - FSM)

Hesaplayıcının özellikleri :

- Kural bağımlı / Güdümlü (Deterministic)
- Araç / Düzenek / Makine (Machine)
- Aşama / Evre (State)
- Sonlu (Finite)

## Tanım

Hesaplayıcı $K$

- $Q$ : Sonlu sayıda eleman (evre) içeren bir küme,
- $\Sigma$ alfabesi : Sonlu sayıda karakterden oluşan bir küme,
- $\delta : Q \times \Sigma \to Q$ şeklinde tanımlanmış bir fonksiyon,
- $q_0 \in Q$ : başlangıç evresi,
- $F \subseteq Q$ : Sonlu sayıda bitiş evresi

olmak üzere $K := (Q, \Sigma, \delta, q_0, F)$ ile tanımlanan soyut bir makinedir.

# Hesaplayıcı nasıl çalışır?

## Örnek

$\delta$ parçalı fonksiyonu

- $\delta(q_0,0) = q_0$,
- $\delta(q_0,1) = q_1$,
- $\delta(q_1,0) = q_0$,
- $\delta(q_1,1) = q_2$,
- $\delta(q_2,0) = q_2$,
- $\delta(q_2,1) = q_1$

olmak üzere

$$K = (\{q_0,q_1,q_2\},\{0,1\},\delta,q_0,\{q_1\})$$

olsun. $K$ hesaplayıcısı 01 sözcüğünü kabul eder mi?

1. $\delta(q_0,0) = q_0$,
2. $\delta(q_0,1) = q_1$.

$q_1$ bitiş evresi olduğu için 01 sözcüğü kabul edilir.

# Hesaplayıcı nasıl çalışır?

## Örnek

$\delta$ parçalı fonksiyonu

- $\delta(q_0,0) = q_0$,
- $\delta(q_0,1) = q_1$,
- $\delta(q_1,0) = q_0$,
- $\delta(q_1,1) = q_2$,
- $\delta(q_2,0) = q_2$,
- $\delta(q_2,1) = q_1$

olmak üzere

$$K = (\{q_0,q_1,q_2\},\{0,1\},\delta,q_0,\{q_1\})$$

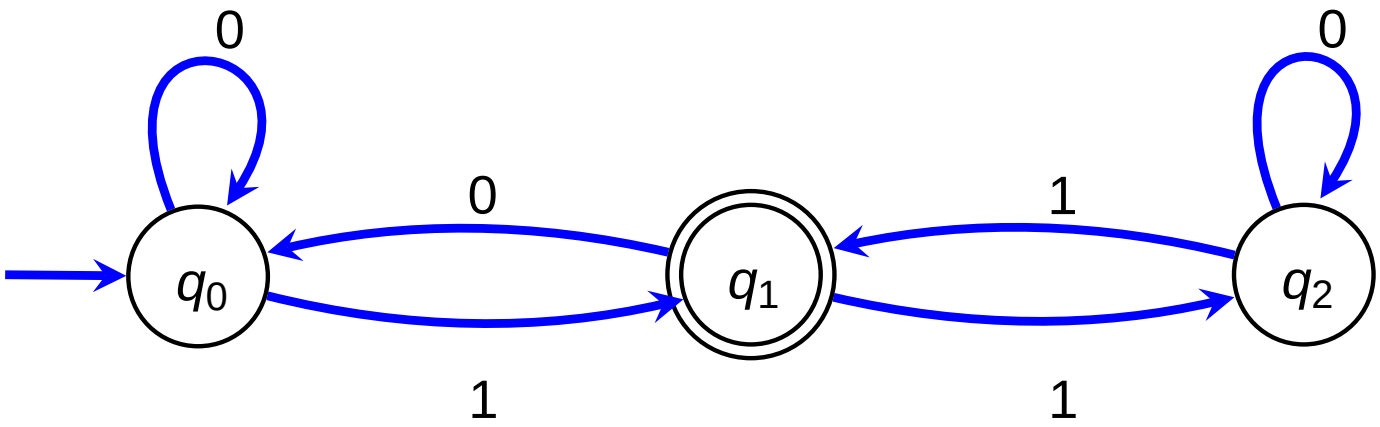
olsun. $K$ hesaplayıcısı 110 sözcüğünü kabul eder mi?

1. $\delta(q_0,1) = q_1$,
2. $\delta(q_1,1) = q_2$,
3. $\delta(q_2,0) = q_2$.

$q_2$ bitiş evresi olmadığı için 110 sözcüğü kabul edilmez.

# Hesaplayıcı nasıl çalışır?

$K$ hesaplayıcısı 10011101 sözcüğünü kabul eder mi?




$q_0\ 1\ q_1\ 0\ q_0\ 0\ q_0\ 1q_1\ 1q_2\ 1\ q_1\ 0\ q_0\ 1\ q_1$

Kabul edilir.

# Serbest Hesaplayıcı (Non-deterministic FSM)

Serbest hesaplayıcının özellikleri :

- Kural bağımsız / Güdümsüz / Serbest (Non-Deterministic)
- Sonlu
- Aşama / Evre
- Araç / Makine

**Tanım**

Serbest hesaplayıcı $S$

- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$ şeklinde tanımlanmış bir fonksiyon

olmak üzere

$$S := (Q, \Sigma, \delta, q_0, F)$$

ile tanımlanan soyut bir makinedir.

Not : $2^Q$ ifadesi $Q$ kümesinin özaltkümesini temsil etmektedir.

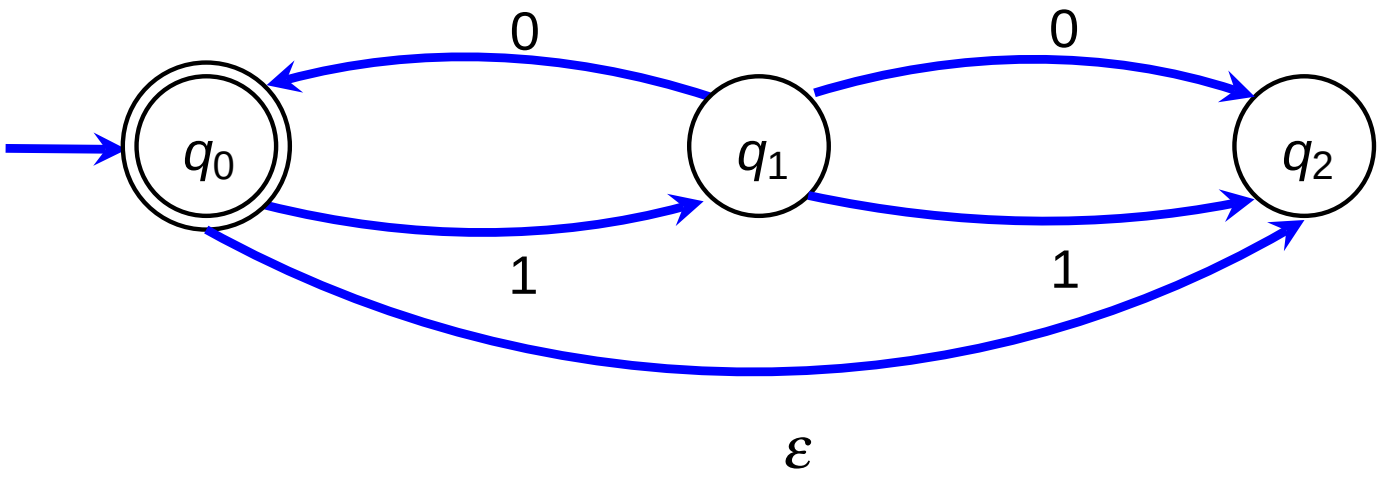
# Serbest hesaplayıcının (kural-bağımlı) hesaplayıcıdan farkı nedir?

1. $\delta$'nın değer kümesi $Q$'dan seçilen bir eleman olarak değil $Q$'nun bir altkümesi olarak verilmiştir. Örneğin,

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

2. $\delta$ fonksiyonunun ikinci parametresi olarak $\varepsilon$ verilebilir. Bu serbest hesaplayıcının karakter getirmeden bir evreden başka bir evreye geçebileceğine ya da aynı evrede kalmaya devam edebileceğine işaret eder.
3. $\emptyset \in 2^Q$ bulunulan evreden geçiş yapılabilecek hiçbir evre olmayabileceğini gösterir.
4. Verilen bir sözcük, bitiş evresine ulaşmanın bir yolu varsa kabul edilir.

# Serbest hesaplayıcı nasıl çalışır?




## Örnek

Verilen serbest hesaplayıcı tarafından

- $\varepsilon$, 10, 1010, 101010 sözcükleri kabul edilir.
- 110, 10100 sözcükleri kabul edilmez.

# Hesaplayıcı için genişletilmiş gösterim

$\delta$ fonksiyonunun yardımı ile $\delta^*$ fonksiyonunu tanımlayalım.

## Tanım

$\delta^* : Q \times \Sigma^* \to Q,$

$$\begin{aligned} \delta^*(q, \varepsilon) &= q, && \textbf{[TEMEL]} \\ \delta^*(q, wa) &= \delta(\delta^*(q, w), a) && \textbf{[ÖZYİNELEME]} \end{aligned}$$

öyleki $q \in Q, w \in \Sigma^*, a \in \Sigma$ şeklinde tanımlanmış bir fonksiyon olsun.

# Hesaplayıcı için genişletilmiş gösterim

Tanımladığımız $\delta^*$ fonksiyonunu ile $K$ hesaplayıcısını kullanalım.

$$
\begin{aligned}
\delta^*(q, 11001) &= \delta(\delta^*(q_0, 1001), 1) \\
&= \delta(\delta(\delta^*(q_0, 001), 1), 1) \\
&= \delta(\delta(\delta(\delta^*(q_0, 01), 1), 1), 0) \\
&= \delta(\delta(\delta(\delta(\delta^*(q_0, 1), 1), 1), 0), 0) \\
&= \delta(\delta(\delta(\delta(\delta(\delta^*(q_0, \varepsilon), 1), 1), 0), 0), 1) \\
&= \delta(\delta(\delta(\delta(\delta(q_0, 1), 1), 0), 0), 1) \\
&= \delta(\delta(\delta(\delta(q_1, 1), 0), 0), 1) \\
&= \delta(\delta(\delta(q_2, 0), 0), 1) \\
&= \delta(\delta(q_2, 0), 1) \\
&= \delta(q_2, 1) \\
&= q_1 \qquad \text{(11001 sözcüğü kabul edilir.)}
\end{aligned}
$$

# Hesaplayıcının kabul ettiği dil

Kural-bağımlı ya da serbest

$$M := (Q, \Sigma, \delta^*, q_0, F)$$

hesaplayıcısının kabul ettiği dili tanımlayalım.

**Tanım ($M$ hesaplayıcısının kabul ettiği dil)**

$$L(M) := \{w \in \Sigma^* : \delta^*(q_0, w) \in F\}.$$

# Hesaplayıcıların eşdeğerliği

**Tanım**

$M_1$ ve $M_2$ hesaplayıcıları verilsin.

$L(M_1) = L(M_2)$ ise $M_1$ ve $M_2$ eşdeğer hesaplayıcılar olarak nitelenir.

**Kuram**

*$S$ serbest hesaplayıcı ve $L(S)$ verilsin.*

*$L(S) = L(M)$ olacak şekilde bir kural-bağımlı hesaplayıcı $M$ vardır.*

Sonuç : Kural-bağımlı hesaplayıcı ve serbest hesaplayıcı eşdeğerdir.

# Düzenli dil (Regular language)

**Tanım**

$L$ bir dil olmak üzere eğer

$L$ dilini kabul eden bir hesaplayıcı tanımlanabiliyor

ise

$L$ düzenli bir dildir.

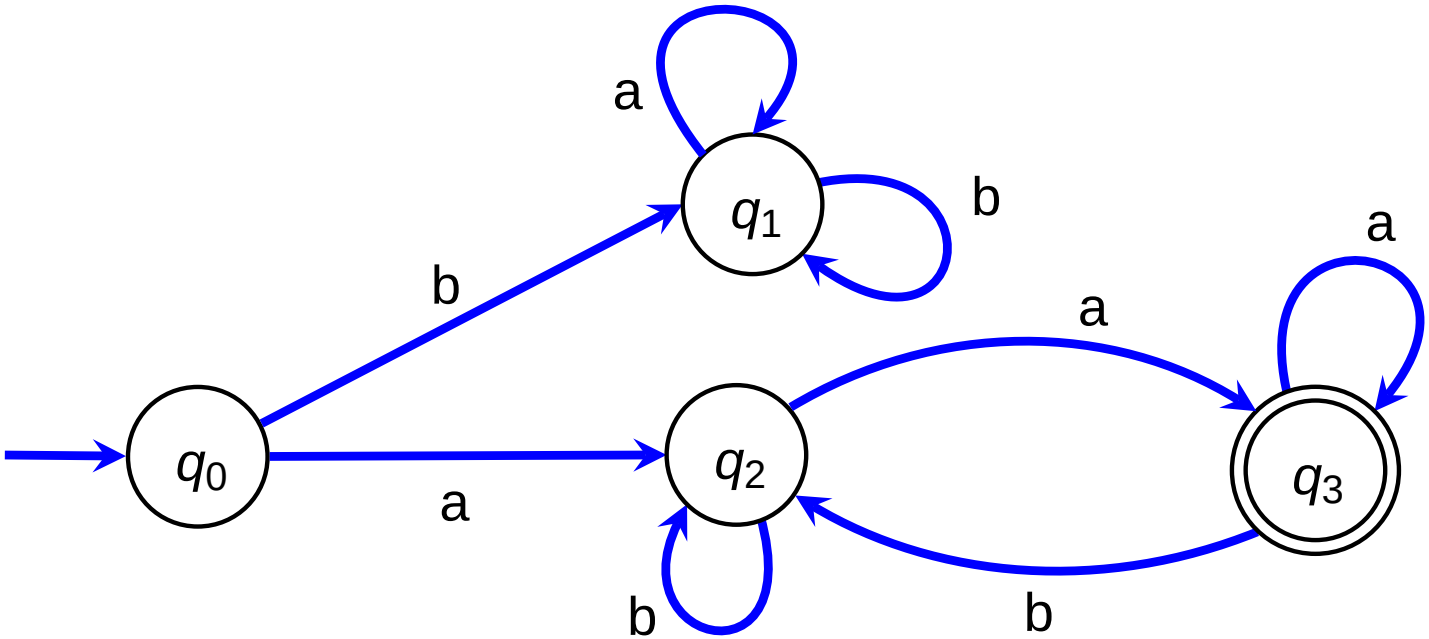
**Not :** $p \rightarrow q$ şeklindeki mantıksal ifadeler matematiksel tanımlara özel olarak $p \leftrightarrow q$ olarak yorumlanır.

# Düzenli dili tanımlayan hesaplayıcı(lar)

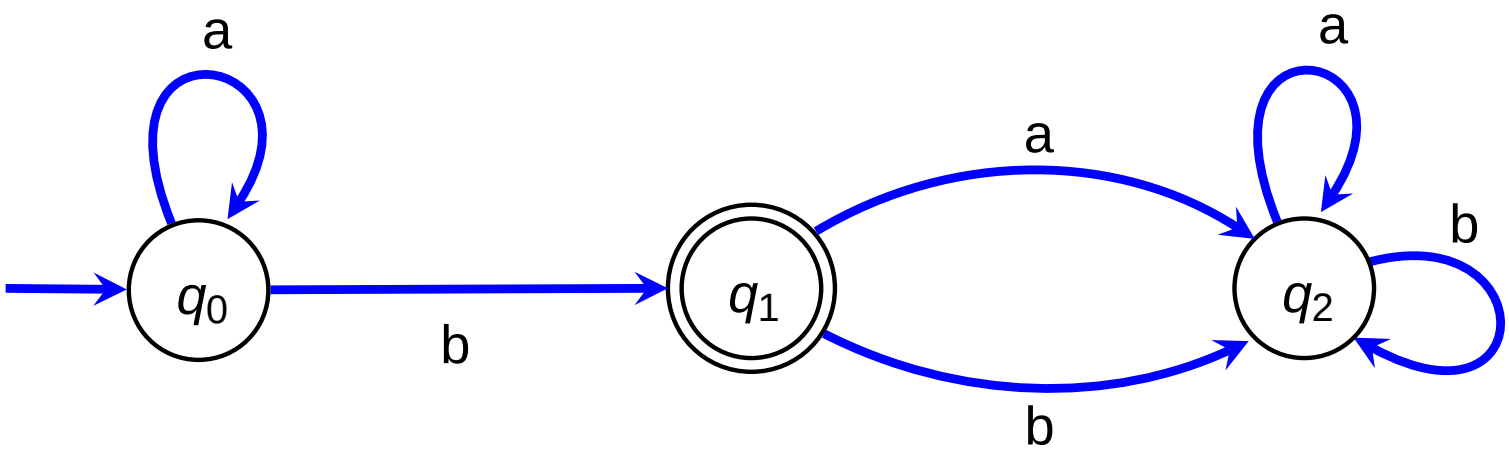
## Örnek

$L := \{awa : w \in \{a,b\}^*\}$ verilsin.

$L$ dilinin düzenli olduğunu gösteriniz.



# Hesaplayıcının kabul ettiği (düzenli) dil




## Örnek

Şekilde verilen $K$ hesaplayıcısının kabul ettiği (düzenli) dili bulunuz.

$$L(K) = L(a^*b).$$

Not : $a^*b$ şeklinde bir anlatımı henüz tanımlamadık.

# Düzenli anlatım (Regular expression)

## Tanım

$\Sigma$ bir alfabe olsun.

1. **[TEMEL]** $\emptyset$, $\varepsilon$, $a \in \Sigma$ düzenli anlatımdır.
2. **[ÖZYİNELEME]** Eğer $r_1$ ve $r_2$ düzenli anlatım ise
   - $(r_1|r_2)$ ile gösterilen $r_1$ ya da $r_2$,
   - $(r_1 r_2)$ ile gösterilen $r_1$ 'i takip eden $r_2$,
   - $(r_1^*)$ ile gösterilen değişken sayıda $r_1$'in birbirini takip etmesi

   anlatımları da düzenli anlatımdır.
3. **[SINIRLAMA]** Bu iki kural dışında kalan hiçbir anlatım düzenli anlatım değildir.

# Düzenli anlatım ile tanımlanan diller

## Tanım

Düzenli anlatım $q$ ile tanımlanan $L(q)$ dili aşağıda belirtilen kurallara göre oluşturulur.

1. **[TEMEL]** $L(\emptyset) = \emptyset, \quad L(\varepsilon) = \{\varepsilon\}, \quad \forall a \in \Sigma, L(a) = \{a\}.$
2. **[ÖZYİNELEME]**
   1. $L(qr) = L(q)L(r),$
   2. $L(q|r) = L(q) \cup L(r),$
   3. $L(q^*) = (L(q))^*.$

# Düzenli anlatım ile tanımlanan diller

**Örnek**

$\{a,b\}$ alfabesi üzerine tanımlanmış $L(a^*b(a|b)^*)$ dilindeki sözcükleri örnekleyiniz.

$$b, ab, abbb, abaaa, ababba.$$

$$\begin{aligned} L(a^*b(a|b)^*) &= L(a^*)L(b)L((a|b)^*) \\ &= (L(a))^*L(b)(L(a|b))^* \\ &= (L(a))^*L(b)(L(a) \cup L(b)))^* \\ &= \{a\}^*\{b\}(\{a\} \cup \{b\})^*. \end{aligned}$$

# Düzenli anlatım, düzenli dil ve hesaplayıcıların ilişkisi

**Kuram**

*$r$ bir düzenli anlatım olsun.*

*$L(r)$ dilini kabul eden bir hesaplayıcı vardır.*

*Dolayısıyla, $L(r)$ düzenli bir dildir.*

**Kuram**

*$T$ bir düzenli dil olsun.*

*$T = L(r)$ olacak şekilde bir düzenli anlatım vardır.*

# Genel bakış

Geride bıraktığımız sunularda

- Kurallı ve düzenli dili tanımladık.
- Bilgisayar için soyut bir model olabileceğine inandığımız hesaplayıcıyı tanımladık.
- Hesaplama yapmanın anlamına dair sormak istediğimiz sorulara cevap araken mantık ilkelerini kullanabileceğimiz bir altyapı hazırladık.

# Temel Kuramlar

**Kuram**

*$L/\Sigma$, düzenli anlatımla ya da hesaplıcıyla tanımlanan düzenli bir dil olsun.*

*Seçilen $w \in \Sigma^*$ sözcüğü $L$ dilinin elemanı olup olmadığını belirleyecek sonlu sayıda adımla ifade edilebilen bir yöntem (algoritma) vardır.*

**Kanıt.**

$L$ dilini tanımlayan hesaplayıcı ile $w$ sözcüğünün kabul edilip edilmeğini sınanır. □

**Kuram**

*$L/\Sigma$ herhangi bir düzenli dil olsun.*

*$L$ dilinin boş, sonlu veya sonsuz küme olduğunu sınacak bir algoritma vardır.*

**Kanıt.**

- $L$ dilini tanımlayan hesaplayıcı geçiş diyagramı olarak temsil edilir.
- Eğer başlangıç evresinden bitiş evresine bir bağlantı bulunmuyorsa $L = \emptyset$ kanıtlanmış olur.
- Eğer başlangıç evresinden bitiş evresine bir bağlantı bulunuyorsa $L \neq \emptyset$ kanıtlanmış olur.
- $L \neq \emptyset$ durumunda geçiş diyagramı bir döngü içeriyorsa $L$ sonsuz kümedir; döngü içermiyorsa $L$ sonlu kümedir.

□

# Hayallerimizin yıkıldığı an

$L = \{a^n b^n : n \geq 0\}$ bir düzenli dil midir? Cevap : Düzenli dil değildir !

# Hayallerimizin yıkıldığı an

$L = \{a^n b^n : n \geq 0\}$ bir düzenli dil midir? Cevap : Düzenli dil değildir !

## Olmayana ergi yöntemi ile ispatlayalım.

1. $L$ dilinin düzenli olduğunu varsayalım. O halde $M = (Q, a, b, \delta, q_0, F)$ gibi tanımlabilecek bir hesaplayıcı vardır.

# Hayallerimizin yıkıldığı an

$L = \{a^n b^n : n \geq 0\}$ bir düzenli dil midir? Cevap : Düzenli dil değildir !

## Olmayana ergi yöntemi ile ispatlayalım.

1. $L$ dilinin düzenli olduğunu varsayalım. O halde $M = (Q, a, b, \delta, q_0, F)$ gibi tanımlabilecek bir hesaplayıcı vardır.
2. $i = 1, 2, 3, \ldots$ için $\delta^*(q_0, a^i)$ ifadesini inceleyelim. $M$ hesaplayıcısında sonlu sayıda evre bulunmasına karşılık sınırsız sayıda $i$ değeri bulunmaktadır.

# Hayallerimizin yıkıldığı an

$L = \{a^n b^n : n \geq 0\}$ bir düzenli dil midir? Cevap : Düzenli dil değildir !

## Olmayana ergi yöntemi ile ispatlayalım.

1. $L$ dilinin düzenli olduğunu varsayalım. O halde $M = (Q, a, b, \delta, q_0, F)$ gibi tanımlabilecek bir hesaplayıcı vardır.
2. $i = 1, 2, 3, \ldots$ için $\delta^*(q_0, a^i)$ ifadesini inceleyelim. $M$ hesaplayıcısında sonlu sayıda evre bulunmasına karşılık sınırsız sayıda $i$ değeri bulunmaktadır.
3. O halde $m \neq n$ olacak şekilde $\delta^*(q_0, a^n) = q$ ve $\delta^*(q_0, a^m) = q$ durumunun mutlaka oluşacaktır.

# Hayallerimizin yıkıldığı an

$L = \{a^n b^n : n \geq 0\}$ bir düzenli dil midir? Cevap : Düzenli dil değildir !

## Olmayana ergi yöntemi ile ispatlayalım.

1. $L$ dilinin düzenli olduğunu varsayalım. O halde $M = (Q, a, b, \delta, q_0, F)$ gibi tanımlabilecek bir hesaplayıcı vardır.
2. $i = 1, 2, 3, \ldots$ için $\delta^*(q_0, a^i)$ ifadesini inceleyelim. $M$ hesaplayıcısında sonlu sayıda evre bulunmasına karşılık sınırsız sayıda $i$ değeri bulunmaktadır.
3. O halde $m \neq n$ olacak şekilde $\delta^*(q_0, a^n) = q$ ve $\delta^*(q_0, a^m) = q$ durumunun mutlaka oluşacaktır.
4. $M$ hesaplayıcısı $a^n b^n$ sözcüğünü kabul ettiğine göre $\delta^*(q, b^n) = q_f \in F$.

# Hayallerimizin yıkıldığı an

$L = \{a^n b^n : n \geq 0\}$ bir düzenli dil midir? Cevap : Düzenli dil değildir !

## Olmayana ergi yöntemi ile ispatlayalım.

1. $L$ dilinin düzenli olduğunu varsayalım. O halde $M = (Q, a, b, \delta, q_0, F)$ gibi tanımlabilecek bir hesaplayıcı vardır.
2. $i = 1, 2, 3, \ldots$ için $\delta^*(q_0, a^i)$ ifadesini inceleyelim. $M$ hesaplayıcısında sonlu sayıda evre bulunmasına karşılık sınırsız sayıda $i$ değeri bulunmaktadır.
3. O halde $m \neq n$ olacak şekilde $\delta^*(q_0, a^n) = q$ ve $\delta^*(q_0, a^m) = q$ durumunun mutlaka oluşacaktır.
4. $M$ hesaplayıcısı $a^n b^n$ sözcüğünü kabul ettiğine göre $\delta^*(q, b^n) = q_f \in F$. O halde $\delta^*(q_0, a^m b^n) = \delta^*(\delta^*(q_0, a^m), b^n) = \delta^*(q, b^n) = q_f$.

# Hayallerimizin yıkıldığı an

$L = \{a^n b^n : n \geq 0\}$ bir düzenli dil midir? Cevap : Düzenli dil değildir !

## Olmayana ergi yöntemi ile ispatlayalım.

1. $L$ dilinin düzenli olduğunu varsayalım. O halde $M = (Q, a, b, \delta, q_0, F)$ gibi tanımlabilecek bir hesaplayıcı vardır.
2. $i = 1, 2, 3, \ldots$ için $\delta^*(q_0, a^i)$ ifadesini inceleyelim. $M$ hesaplayıcısında sonlu sayıda evre bulunmasına karşılık sınırsız sayıda $i$ değeri bulunmaktadır.
3. O halde $m \neq n$ olacak şekilde $\delta^*(q_0, a^n) = q$ ve $\delta^*(q_0, a^m) = q$ durumunun mutlaka oluşacaktır.
4. $M$ hesaplayıcısı $a^n b^n$ sözcüğünü kabul ettiğine göre $\delta^*(q, b^n) = q_f \in F$. O halde $\delta^*(q_0, a^m b^n) = \delta^*(\delta^*(q_0, a^m), b^n) = \delta^*(q, b^n) = q_f$. $M$ hesaplayıcısının $a^m b^n$ sözcüğünü kabul etmesi bir çelişkidir.

□

# Nereye vardık?

Hesaplamanın doğasına ilişkin birçok soruyu cevaplayabilmesine karşın

hesaplayıcılar,

bilgisayarı soyut olarak modellemek için

yetersizdir !

# Peki şimdi ne olacak?

Bilgisayarın soyut modeli var mıdır?

Geliştirilen diğer modeller :

- Düzenli gramer. YETERSİZ
- Bağlam-bağımsız diller (Context-free languages). YETERSİZ
- Bellekli hesaplayıcılar (Pushdown automata). YETERSİZ
- ...
- Turing makinesi. YETERSİZLİĞİ İSPATLANAMAMIŞTIR !
- ...

Daha fazla bilgi için bkz. Chomsky'nin sınıflandırması.

# Alan Turing (1912-1954)

Matematik, mantık, kriptografi, hesaplama alanlarında çalışmış İngiliz bilimadamı

http://upload.wikimedia.org/wikipedia/en/c/c8/Alan_Turing_photo.jpg

# Turing Makinesi

## Tanım

Turing Makinesi $T$

- $Q$ : Sonlu sayıda eleman (evre) içeren bir küme,
- $\Gamma$ alfabesi : Sonlu sayıda karakterden oluşan bir küme,
- $\Sigma$ alfabesi : Sonlu sayıda karakterden oluşan bir küme

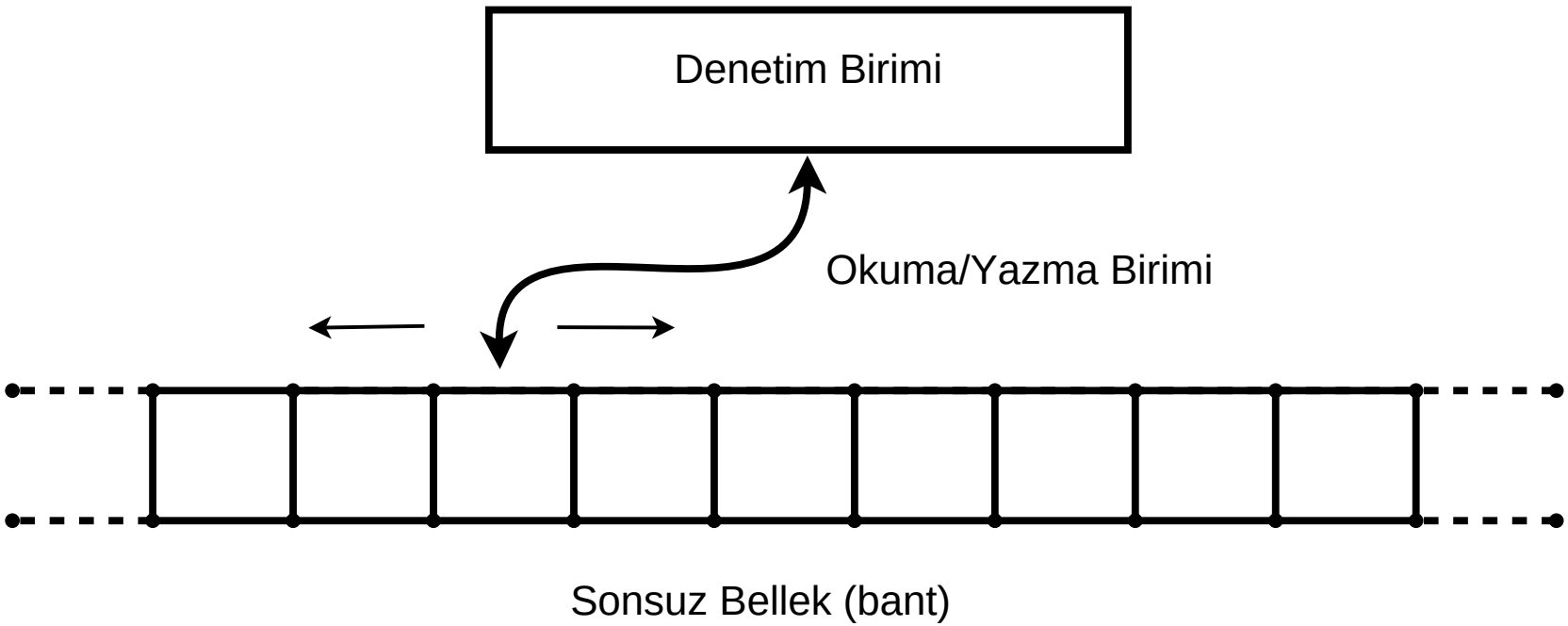
$$\text{öyleki } \Sigma \subseteq \Gamma - \{\square\},$$

- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ şeklinde tanımlanmış bir fonksiyon,
- $\square \in F$ : özel olarak tanımlanmış boşluk karakteri,
- $q_0 \in Q$ : başlangıç evresi,
- $F \subseteq Q$ : Sonlu sayıda bitiş evresi olmak üzere

$$T := (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$$

ile tanımlanan soyut bir makinedir.

# Turing Makinesi



# Turing Makinesi - Bilgisayar Benzerliği

| Turing Makinesi | Bilgisayar |
|---|---|
| $\Gamma$ | Olası atama değerleri |
| $\delta$ | Program |
| $q_0$ | Boot sector |
| $L, R$ | İşlemci komutları |
| Sözcük bandı | Bellek ve Giriş/Çıkış Aygıtı |

# Turing Makinesi Nasıl Çalışır?

$\{a,b\}$ alfabesi üzerine tanımlanan $\{a^n b^n : n \geq 1\}$ dilini kabul eden Turing makinesi aşağıda örneklenmiştir :

$$\begin{array}{lll} \delta(q_0,a) = (q_1,x,R), & \delta(q_2,y) = (q_2,y,L), & \delta(q_3,y) = (q_3,y,R), \\ \delta(q_1,a) = (q_1,a,R), & \delta(q_2,a) = (q_2,a,L), & \delta(q_3,\Box) = (q_4,\Box,R). \\ \delta(q_1,y) = (q_1,y,R), & \delta(q_2,x) = (q_0,x,R), & \\ \delta(q_1,b) = (q_2,y,L), & \delta(q_0,y) = (q_3,y,R), & \end{array}$$

olmak üzere

$$T := \{\{q_0,q_1,q_2,q_3,q_4\},\{a,b\},\{a,b,x,y,\Box\},\delta,q_0,\{q_4\}\}.$$

$T$ makinesini sınamak için $aabb$ sözcüğünü kullanalım :

$q_0aabb \vdash xq_1abb \vdash xaq_1bb \vdash xq_2ayb \vdash q_2xayb \vdash xq_0ayb \vdash xxq_1yb \vdash xxyq_1b \vdash xxq_2yy \vdash xq_2xyy \vdash xxq_0yy \vdash xxyq_3y \vdash xxyq_3\Box \vdash xxyy\Box q_4\Box.$

# Hesaplama tanımı ve Church-Turing Tezi

## Tanım

$T$ bir Turing makinesi olsun.

$T$'ti bitiş evresive getiren bir sözcük için evreler arasında yapılan geçişlerin tamamına hesaplama denir.

## Church-Turing tezi.

Mekanik olarak yapılabilen her hesaplamayı yapabilecek bir Turing makinesi vardır. □

Not : Church-Turing tezi hesaplamanın tanımı gereği bir tezden ziyade tanım hatta aksiyom olarak değerlendirilmektedir.

# Turing Makinesi Neyi Hesaplayamaz?

**Kuram**

*Bir Turing makinesinin verilen bir sözcük ile bitiş evresine ulaşıp ulaşamayacağını belirleyebilecek bir Turing makinesi yoktur.*

**Not :** Turing makinesinin sınırsız işlem gücüyle çözemeyeceği birçok problem tanımlanmıştır.

# Diğer konular

- Hesaplama Analizi
  - $O(g(n))$, $\Omega(g(n))$, $\Theta(g(n))$.
- Karmaşıklık Teorisi
  - P, NP, NP-Bütün (NP-Complete), NP-Zor (NP-Hard)
- P sınıfı NP sınıfına eşdeğer midir?

# NP-Bütün Prolemlerin Tanımlandığı Bazı Alanlar

- Graph theory (Graph coloring)
- Network design (Traveling salesman)
- Storage and retrieval (Dynamic storage allocation)
- Sequencing and scheduling (Multiprocessor scheduling)
- Mathematical programming (Knapsack)
- Algebra and number theory (Algebraic equations over any finite field.)
- Games and puzzles (Sudoku, Tetris)
- Logic (Boolean satisfiability)
- Automata and language theory (Reduction of incompletely specified automata)
- Program optimization (Register sufficiency for loops)

Kaynakça :

Peter Linz, *“An Introduction to Formal Languages and Automata”*, Jones and Barlett Publishers, 2001.