بسم الله الرحمن الرحیم

نظریه زبان‌ها و ماشین‌ها

جلسه ۲۳

مجتبی خلیلی
دانشکده برق و کامپیوتر
دانشگاه صنعتی اصفهان

ماشین تورینگ

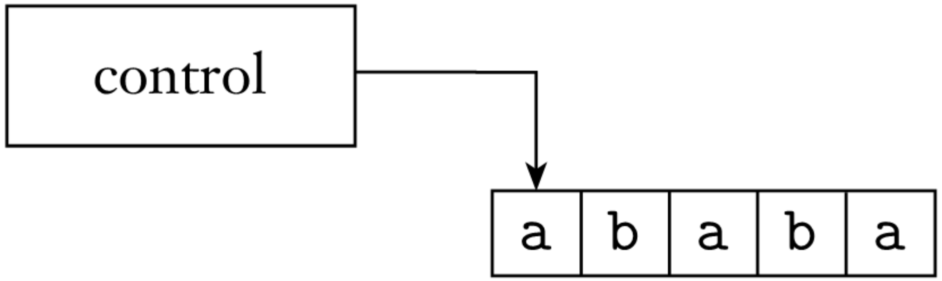
- تعریف ماشین تورینگ را دیدیم (با یک نوار نامتناهی و حرکتهای قطعی یکی یکی به چپ یا راست).
- آیا میتوان ماشینی همانند آن اما با طول نوار متناهی (برابر طول ورودی) در نظر گرفت؟ آیا هم قدرتند؟

linear bounded automaton

- ماشین کراندار خطی

DEFINITION 5.6

A ***linear bounded automaton*** is a restricted type of Turing machine wherein the tape head isn't permitted to move off the portion of the tape containing the input. If the machine tries to move its head off either end of the input, the head stays where it is—in the same way that the head will not move off the left-hand end of an ordinary Turing machine's tape.



linear bounded automaton

- مطابق معمول، زبان ماشین مجموعه همه رشته‌هایی است که پذیرش میکند.

- آیا این ماشین از PDA قویتر است؟
- آیا از ماشین تورینگ ضعیفتر است؟
- آیا هم ارز یا معادلند؟

Equivalence of Classes of Automata

DEFINITION 10.1

Two automata are equivalent if they accept the same language. Consider two classes of automata C_1 and C_2. If for every automaton M_1 in C_1 there is an automaton M_2 in C_2 such that

$$L(M_1) = L(M_2),$$

we say that C_2 is at least as powerful as C_1. If the converse also holds and for every M_2 in C_2 there is an M_1 in C_1 such that $L(M_1) = L(M_2)$, we say that C_1 and C_2 are equivalent.

Equivalence of Classes of Automata

There are many ways to establish the equivalence of automata. The construction of Theorem 2.2 does this for dfa's and nfa's. For demonstrating the equivalence in connection with Turing's machines, we often use the important technique of **simulation**.

Equivalence of Classes of Automata

IUT-ECE

- مثال:

THEOREM 10.1

The class of Turing machines with a stay-option is equivalent to the class of standard Turing machines.

$$\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{\mathrm{L}, \mathrm{R}\}$$

TM

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

TM with stay

Equivalence of Classes of Automata

IUT-ECE

- مثال:

Proof: Since a Turing machine with a stay-option is clearly an extension of the standard model, it is obvious that any standard Turing machine can be simulated by one with a stay-option.

$$\boxed{\delta: Q \times \Gamma \longrightarrow Q \times \Gamma \times \{\mathrm{L}, \mathrm{R}\}} \Longrightarrow \boxed{\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R, S\}}$$

TM

TM with stay

Equivalence of Classes of Automata

IUT-ECE

Sipser: $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$,

○ مثال:

To show the converse, let $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$ be a Turing machine with a stay-option to be simulated by a standard Turing machine $\widehat{M} = \left(\widehat{Q}, \Sigma, \Gamma, \widehat{\delta}, \widehat{q}_0, \square, \widehat{F}\right)$. For each move of M, the simulating machine $\widehat{M}$ does the following.

$$\boxed{\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{\mathrm{L}, \mathrm{R}\}} \longleftarrow \boxed{\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R, S\}}$$

TM

TM with stay

Equivalence of Classes of Automata

IUT-ECE

- مثال:

To show the converse, let $M = (Q, \Sigma, \Gamma, \delta, q_0, \Box, F)$ be a Turing machine with a stay-option to be simulated by a standard Turing machine $\widehat{M} = \left(\widehat{Q}, \Sigma, \Gamma, \widehat{\delta}, \widehat{q}_0, \Box, \widehat{F}\right)$. For each move of M, the simulating machine $\widehat{M}$ does the following.

$$\delta(q_i, a) = (q_j, b, L \text{ or } R), \quad \longrightarrow \quad \widehat{\delta}(\widehat{q}_i, a) = (\widehat{q}_i, b, L \text{ or } R).$$

$$\delta(q_i, a) = (q_j, b, S), \quad \longrightarrow \quad \left\{ \begin{array}{l} \widehat{\delta}(\widehat{q}_i, a) = (\widehat{q}_{jS}, b, R), \\ \widehat{\delta}(\widehat{q}_{jS}, c) = (\widehat{q}_j, c, L), \\ \text{for all } c \in \Gamma. \end{array} \right.$$

linear bounded automaton

- مطابق معمول، زبان ماشین مجموعه همه رشته‌هایی است که پذیرش میکند.

- آیا این ماشین از PDA قویتر است؟
- آیا از ماشین تورینگ ضعیفتر است؟
- آیا هم ارز یا معادلند؟

linear bounded automaton

- ثابت میتوان کرد که از TM ضعیفتر است.

- از طرفی LBA زبان‌های زیر را تشخیص میدهد:

$$B = \{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n \mid n \geq 0\}$$

$$A = \{ww \mid w \in \Sigma^*\}$$

زبانهای حساس به متن

DEFINITION 11.4

A grammar $G = (V, T, S, P)$ is said to be **context sensitive** if all productions are of the form

$$x \to y,$$

where $x,\ y \in (V \cup T)^+$ and

$$|x| \leq |y|\,. \tag{11.15}$$

زبانهای حساس به متن

It is less obvious why such grammars should be called context sensitive, but it can be shown (see, for example, Salomaa 1973) that all such grammars can be rewritten in a normal form in which all productions are of the form

$$xAy \to xvy.$$

زبانهای حساس به متن

DEFINITION 11.5

A language L is said to be context sensitive if there exists a context-sensitive grammar G, such that $L = L(G)$ or $L = L(G) \cup \{\lambda\}$.

زبانهای حساس به متن

EXAMPLE 11.2

The language $L = \{a^n b^n c^n : n \geq 1\}$ is a context-sensitive language. We show this by exhibiting a context-sensitive grammar for the language. One such grammar is

$$\begin{aligned} S &\rightarrow abc|aAbc, \\ Ab &\rightarrow bA, \\ Ac &\rightarrow Bbcc, \\ bB &\rightarrow Bb, \\ aB &\rightarrow aa|aaA. \end{aligned}$$

We can see how this works by looking at a derivation of $a^3b^3c^3$.

$$\begin{aligned} S &\Rightarrow aAbc \Rightarrow abAc \Rightarrow abBbcc \\ &\Rightarrow aBbbcc \Rightarrow aaAbbcc \Rightarrow aabAbcc \\ &\Rightarrow aabbAcc \Rightarrow aabbBbccc \\ &\Rightarrow aabBbbccc \Rightarrow aaBbbbccc \\ &\Rightarrow aaabbbccc. \end{aligned}$$

CSG & LBA

THEOREM 11.8

For every context-sensitive language L not including λ, there exists some linear bounded automaton M such that $L = L(M)$.

تعریف فرمال گرامر

DEFINITION 1.1

A grammar G is defined as a quadruple

$$G = (V, T, S, P),$$

where V is a finite set of objects called **variables**,
T is a finite set of objects called **terminal symbols**,
$S \in V$ is a special symbol called the **start** variable,
P is a finite set of **productions**.

It will be assumed without further mention that the sets V and T are nonempty and disjoint.

$$\boxed{P \;:\; (V \cup T)^{+} \longrightarrow (V \cup T)^{*}.}$$

انواع گرامرها

$$P \;:\quad x \to y,$$

Type 0

Type 1

CSG

$$x,\ y \in (V \cup T)^+$$

$$|x| \le |y|\,.$$

Type 2

CFG

$$y \in (V \cup T)^*$$

$$x \in V$$

Type 3

Regular

$$A \to xB,\ A \to Bx,$$

$$A \to x,$$

$$A,\ B \in V,\ \text{and}\ x \in T^*.$$

Mojtaba Khalili

گرامرهای بدون محدودیت

DEFINITION 11.3

A grammar $G = (V, T, S, P)$ is called **unrestricted** if all the productions are of the form

$$u \rightarrow v,$$

where u is in $(V \cup T)^+$ and v is in $(V \cup T)^*$.

A Grammar Generating $\{a^{2^k} \mid k \in \mathcal{N}\}$

Let $L = \{a^{2^k} \mid k \in \mathcal{N}\}$. L can be defined recursively by saying that $a \in L$ and that for every $n \geq 1$, if $a^n \in L$, then $a^{2n} \in L$. Using this idea to obtain a grammar means finding a way to double the number of a's in the string obtained so far. The idea is to use a variable D that will act as a "doubling operator." D replaces each a by two a's, by means of the production $Da \to aaD$. At the beginning of each pass, D is introduced at the left end of the string, and we think of each application of the production as allowing D to move past an a, doubling it in the process. The complete grammar has the productions

$$S \to LaR \qquad L \to LD \qquad Da \to aaD \qquad DR \to R \qquad L \to \Lambda \qquad R \to \Lambda$$

Beginning with the string LaR, the number of a's will be doubled every time a copy of D is produced and moves through the string. Both variables L and R can disappear at any time. There is no danger of producing a string of a's in which the action of one of the doubling operators is cut short, because if R disappears when D is present, there is no way for D to be eliminated. The string $aaaa$ has the derivation

$$\begin{aligned} S &\Rightarrow LaR \Rightarrow LDaR \Rightarrow LaaDR \Rightarrow LaaR \Rightarrow LDaaR \\ &\Rightarrow LaaDaR \Rightarrow LaaaaDR \Rightarrow LaaaaR \Rightarrow aaaaR \Rightarrow aaaa \end{aligned}$$

$$S \rightarrow ACaB$$
$$Ca \rightarrow aaC$$
$$CB \rightarrow DB|E$$
$$aD \rightarrow Da$$
$$AD \rightarrow AC$$
$$aE \rightarrow Ea$$
$$AE \rightarrow \varepsilon$$

$$S \Rightarrow$$
$$ACaB \Rightarrow$$
$$AaaCB \Rightarrow$$
$$AaaDB \Rightarrow$$
$$AaDaB \Rightarrow$$
$$ADaaB \Rightarrow$$
$$ACaaB \Rightarrow$$
$$AaaCaaB \Rightarrow$$
$$AaaaaCB \Rightarrow$$
$$AaaaaE \Rightarrow$$
$$AaaaEa \Rightarrow$$
$$AaaEaa \Rightarrow$$
$$AaEaaa \Rightarrow$$
$$AEaaaa \Rightarrow$$
$$aaaa$$

A Grammar Generating $\{a^n b^n c^n \mid n \geq 1\}$

The previous example used the idea of a variable moving through the string and operating on it. In this example we use a similar left-to-right movement, although there is no explicit "operator" like the variable D, as well as another kind arising from the variables rearranging themselves. Like the previous example, this one uses the variable L to denote the left end of the string.

We begin with the productions

$$S \to SABC \mid LABC$$

which allow us to obtain strings of the form $L(ABC)^n$, where $n \geq 1$. Next, productions that allow the variables A, B, and C to arrange themselves in alphabetical order:

$$BA \to AB \qquad CB \to BC \qquad CA \to AC$$

Finally, productions that allow the variables to be replaced by the corresponding terminals, *provided* they are in alphabetical order:

$$LA \to a \qquad aA \to aa \qquad aB \to ab \qquad bB \to bb \qquad bC \to bc \qquad cC \to cc$$

Although nothing forces the variables to arrange themselves in alphabetical order, doing so is the only way they can ultimately be replaced by terminals.

A CSG Generating $L = \{a^n b^n c^n \mid n \geq 1\}$

The unrestricted grammar we used in Example 8.12 for this language is not context-sensitive, because of the production $LA \to a$, but using the same principle we can easily find a grammar that is. Instead of using the variable A as well as a separate variable to mark the beginning of the string, we introduce a new variable to serve both purposes. It is not hard to verify that the CSG with productions

$$S \to SABC \mid \mathcal{A}BC \qquad BA \to AB \qquad CA \to AC \qquad CB \to BC$$
$$\mathcal{A} \to a \qquad aA \to aa \qquad aB \to ab \qquad bB \to bb \qquad bC \to bc \qquad cC \to cc$$

generates the language L.

انواع گرامرها (چامسکی)

IUT-ECE

$$P \ : \quad x \rightarrow y,$$

Type 0

unrestricted

$u \rightarrow v,$

u is in $(V \cup T)^{+}$

v is in $(V \cup T)^{*}.$

Type 1

CSG

$x, y \in (V \cup T)^{+}$

$|x| \leq |y|.$

Type 2

CFG

$y \in (V \cup T)^{*}$

$x \in V$

Type 3

Regular

$A \rightarrow xB, \ A \rightarrow Bx,$

$A \rightarrow x,$

$A, \ B \in V$, and $x \in T^{*}.$

گرامرهای بدون محدودیت و زبانهای تورینگ تشخیص‌پذیر

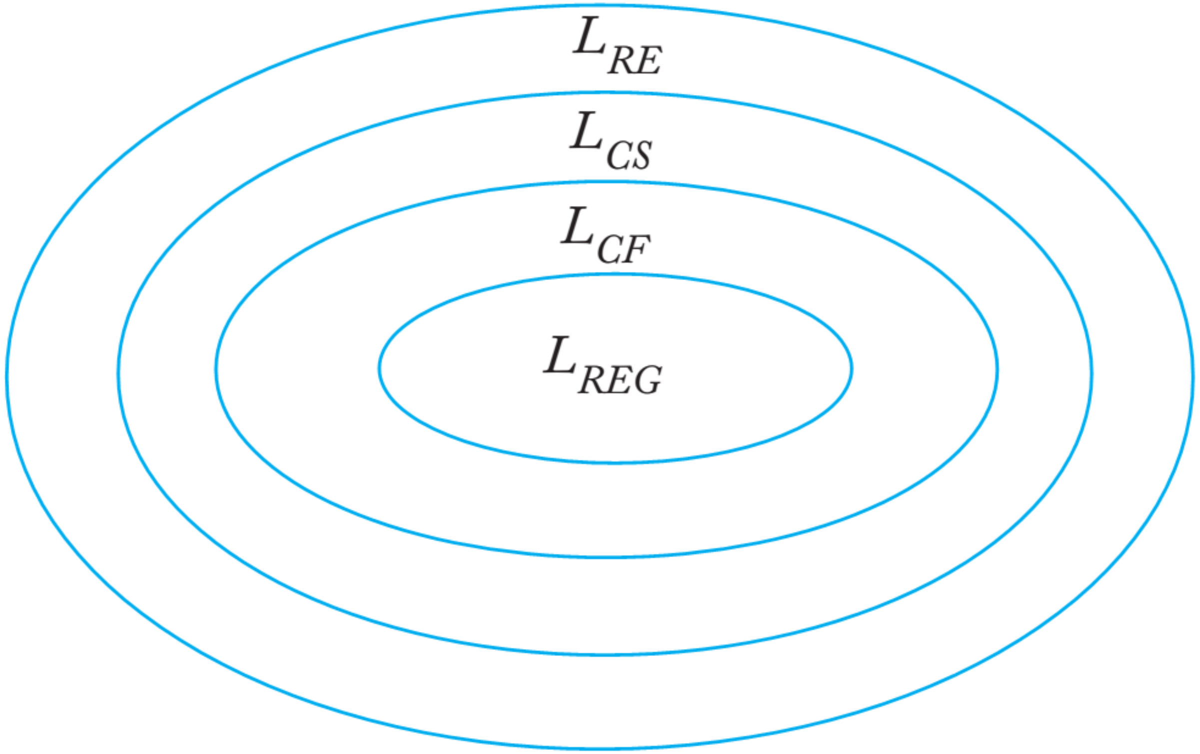
THEOREM 11.6

Any language generated by an unrestricted grammar is recursively enumerable.

سلسله مراتب زبانها

Type	Language	Grammar	Automaton
0	recursively enumerable	unrestricted	TM
1	context-sensitive	context-sensitive	LBA
2	context-free	context-free	PDA
3	regular	regular	DFA

سلسله مراتب زبانها



سلسله مراتب زبانها