بسم الله الرحمن الرحيم

نظریه زبان‌ها و ماشین‌ها

جلسه ۲۶

مجتبی خلیلی
دانشکده برق و کامپیوتر
دانشگاه صنعتی اصفهان

تصمیم‌پذیری

- تاکنون ماشین تورینگ را به عنوان مدلی برای یک کامپیوتر (محاسبه کننده) عام بیان کرده‌ایم و همچنین مفهوم الگوریتم را به طور دقیق معرفی کرده‌ایم (تز چرچ-تورینگ).

- حال سوال این است برای چه مسائلی الگوریتم داریم؟ و برای چه مسائلی الگوریتم نداریم؟

تصمیم‌پذیری

In this chapter we begin to investigate the power of algorithms to solve problems. We demonstrate certain problems that can be solved algorithmically and others that cannot. Our objective is to explore the limits of algorithmic solvability. You are probably familiar with solvability by algorithms because much of computer science is devoted to solving problems. The unsolvability of certain problems may come as a surprise.

زبان‌های تصمیم‌پذیر

- در ادامه قصد داریم مثالهایی را بررسی کنیم که توسط ماشین تورینگ تصمیم‌پذیر هستند (الگوریتم دارند- توصیف سطح بالا از ماشین تورینگ).

زبان‌های تصمیم‌پذیر

- چگونه نشان دهیم یک زبان تصمیم‌پذیر است؟

زبان‌های تصمیم‌پذیر

- با فرض DFA D و رشته w، آیا D رشته w را میپذیرد؟
- میدانیم میتوانیم توصیف یک ماشین را به عنوان ورودی به ماشین تورینگ دهیم.

$$A_{\mathsf{DFA}} = \{\langle B, w\rangle \mid B \text{ is a } \mathsf{DFA} \text{ that accepts input string } w\}.$$

زبان‌های تصمیم‌پذیر

- آیا این زبان تصمیم‌پذیر است؟

$$A_{\mathsf{DFA}} = \{\langle B, w\rangle \mid B \text{ is a } \mathsf{DFA} \text{ that accepts input string } w\}.$$

THEOREM 4.1

A_{DFA} is a decidable language.

زبان‌های تصمیم‌پذیر

$$A_{\mathsf{DFA}} = \{\langle B, w\rangle |\ B \text{ is a } \mathsf{DFA} \text{ that accepts input string } w\}.$$

PROOF IDEA We simply need to present a TM M that decides A_{DFA}.

$M =$ "On input $\langle B, w\rangle$, where B is a DFA and w is a string:

1. Simulate B on input w.
2. If the simulation ends in an accept state, *accept*. If it ends in a nonaccepting state, *reject*."

PROOF We mention just a few implementation details of this proof. For those of you familiar with writing programs in any standard programming language, imagine how you would write a program to carry out the simulation.

First, let's examine the input $\langle B, w\rangle$. It is a representation of a DFA B together with a string w. One reasonable representation of B is simply a list of its five components: Q, Σ, δ, q_0, and F. When M receives its input, M first determines whether it properly represents a DFA B and a string w. If not, M rejects.

Then M carries out the simulation directly. It keeps track of B's current state and B's current position in the input w by writing this information down on its tape. Initially, B's current state is q_0 and B's current input position is the leftmost symbol of w. The states and position are updated according to the specified transition function δ. When M finishes processing the last symbol of w, M accepts the input if B is in an accepting state; M rejects the input if B is in a nonaccepting state.

زبان‌های تصمیم‌پذیر (درباره زبانهای منظم)

$A_{\mathsf{DFA}} = \{\langle B, w\rangle |\ B \text{ is a } \mathsf{DFA} \text{ that accepts input string } w\}.$

$A_{\mathsf{NFA}} = \{\langle B, w\rangle |\ B \text{ is an } \mathsf{NFA} \text{ that accepts input string } w\}.$

$A_{\mathsf{REX}} = \{\langle R, w\rangle |\ R \text{ is a regular expression that generates string } w\}.$

$E_{\mathsf{DFA}} = \{\langle A\rangle |\ A \text{ is a } \mathsf{DFA} \text{ and } L(A) = \emptyset\}.$

$EQ_{\mathsf{DFA}} = \{\langle A, B\rangle |\ A \text{ and } B \text{ are } \mathsf{DFAs} \text{ and } L(A) = L(B)\}.$

زبان‌های تصمیم‌پذیر (درباره زبان مستقل از متن)

$A_{\mathsf{CFG}} = \{\langle G, w\rangle \mid G \text{ is a } \mathsf{CFG} \text{ that generates string } w\}.$

$E_{\mathsf{CFG}} = \{\langle G\rangle \mid G \text{ is a } \mathsf{CFG} \text{ and } L(G) = \emptyset\}.$

زبان‌های تصمیم‌پذیر

- مثال: زبان زیر را در نظر بگیرید:

$$A_{\mathsf{NFA}} = \{\langle B, w\rangle |\ B \text{ is an NFA that accepts input string } w\}.$$

THEOREM 4.2

A_{NFA} is a decidable language.

زبان‌های تصمیم‌پذیر

- اثبات:

$N =$ "On input $\langle B, w\rangle$, where B is an NFA and w is a string:

1. Convert NFA B to an equivalent DFA C, using the procedure for this conversion given in Theorem 1.39.
2. Run TM M from Theorem 4.1 on input $\langle C, w\rangle$.
3. If M accepts, *accept*; otherwise, *reject*."

زبان‌های تصمیم‌پذیر

- مثال: زبان زیر را در نظر بگیرید:

$$A_{\mathsf{REX}} = \{\langle R, w\rangle |\ R \text{ is a regular expression that generates string } w\}.$$

زبان‌های تصمیم‌پذیر

- اثبات:

PROOF The following TM P decides A_{REX}.

$P =$ "On input $\langle R, w\rangle$, where R is a regular expression and w is a string:

1. Convert regular expression R to an equivalent NFA A by using the procedure for this conversion given in Theorem 1.54.
2. Run TM N on input $\langle A, w\rangle$.
3. If N accepts, *accept*; if N rejects, *reject*."

زبان‌های تصمیم‌پذیر

- مثال: زبان زیر را در نظر بگیرید:

$$E_{\mathsf{DFA}} = \{\langle A\rangle |\ A \text{ is a } \mathsf{DFA} \text{ and } L(A) = \emptyset\}.$$

زبان‌های تصمیم‌پذیر

- اثبات:

$T =$ "On input $\langle A \rangle$, where A is a DFA:

1. Mark the start state of A.

2. Repeat until no new states get marked:

3. Mark any state that has a transition coming into it from any state that is already marked.

4. If no accept state is marked, *accept*; otherwise, *reject*."

زبان‌های تصمیم‌پذیر

- مثال: زبان زیر را در نظر بگیرید:

$$EQ_{\mathsf{DFA}} = \{\langle A, B\rangle \mid A \text{ and } B \text{ are } \mathsf{DFAs} \text{ and } L(A) = L(B)\}.$$

زبان‌های تصمیم‌پذیر

IUT-ECE

- اثبات:

DFA C : C accepts only those strings that are accepted by either A or B but not by both.

$F =$ "On input $\langle A, B\rangle$, where A and B are DFAs:

1. Construct DFA C as described.

$$L(C) = \left(L(A) \cap \overline{L(B)}\right) \cup \left(\overline{L(A)} \cap L(B)\right).$$

$$L(C) = \emptyset \text{ iff } L(A) = L(B).$$

We can construct C from A and B with the constructions for proving the class of regular languages closed under complementation, union, and intersection. These constructions are algorithms that can be carried out by turing machines.

زبان‌های تصمیم‌پذیر

- اثبات:

$F =$ "On input $\langle A, B\rangle$, where A and B are DFAs:

1. Construct DFA C as described.

2. Run TM T from Theorem 4.4 on input $\langle C\rangle$.

3. If T accepts, *accept*. If T rejects, *reject*."

زبان‌های تصمیم‌پذیر (درباره مستقل از متن ها)

- مسائلی مشابهی مثل قبل: پذیرش، تهی بودن و ...

زبان‌های تصمیم‌پذیر

- زبان زیر را در نظر بگیرید:

$$A_{\mathsf{CFG}} = \{\langle G, w\rangle \mid G \text{ is a CFG that generates string } w\}.$$

THEOREM 4.7

A_{CFG} is a decidable language.

زبان‌های تصمیم‌پذیر

IUT-ECE

- اثبات:

PROOF The TM S for A_{CFG} follows.

$S =$ "On input $\langle G, w \rangle$, where G is a CFG and w is a string:

Idea 1: check all derivations?

If G does not generate w, this algorithm would never halt => RE

زبان‌های تصمیم‌پذیر

- اثبات:

PROOF The TM S for A_{CFG} follows.

$S =$ "On input $\langle G, w\rangle$, where G is a CFG and w is a string:

Idea 2: Use CNF.
any derivation of w has 2n − 1 steps, where n is the length of w.

زبان‌های تصمیم‌پذیر

IUT-ECE

- اثبات:

PROOF The TM S for A_{CFG} follows.

$S =$ "On input $\langle G, w\rangle$, where G is a CFG and w is a string:

1. Convert G to an equivalent grammar in Chomsky normal form.

We can convert G to Chomsky normal form by using the procedure given in Section 2.1.

زبان‌های تصمیم‌پذیر

- اثبات:

PROOF The TM S for A_{CFG} follows.

$S =$ "On input $\langle G, w\rangle$, where G is a CFG and w is a string:

1. Convert G to an equivalent grammar in Chomsky normal form.
2. List all derivations with $2n-1$ steps, where n is the length of w; except if $n = 0$, then instead list all derivations with one step.
3. If any of these derivations generate w, *accept*; if not, *reject*."

زبان‌های تصمیم‌پذیر

- زبان زیر را در نظر بگیرید:

$$E_{\mathsf{CFG}} = \{\langle G\rangle |\ G \text{ is a CFG and } L(G) = \emptyset\}.$$

THEOREM 4.8

E_{CFG} is a decidable language.

زبان‌های تصمیم‌پذیر

- اثبات:

PROOF

$R =$ "On input $\langle G \rangle$, where G is a CFG:

1. Mark all terminal symbols in G.
2. Repeat until no new variables get marked:
3. Mark any variable A where G has a rule $A \rightarrow U_1 U_2 \cdots U_k$ and each symbol $U_1, \ldots, U_k$ has already been marked.
4. If the start variable is not marked, *accept*; otherwise, *reject*."

زبان‌های تصمیم‌پذیر

- زبان زیر را در نظر بگیرید:

$$E_{\mathsf{CFG}} = \{\langle G\rangle \mid G \text{ is a CFG and } L(G) = \emptyset\}.$$

THEOREM 4.8

E_{CFG} is a decidable language.

زبان‌های تصمیم‌پذیر

IUT-ECE

- اثبات:

PROOF

$R =$ "On input $\langle G \rangle$, where G is a CFG:

1. Mark all terminal symbols in G.

2. Repeat until no new variables get marked:

3. Mark any variable A where G has a rule $A \to U_1 U_2 \cdots U_k$ and each symbol $U_1, \ldots, U_k$ has already been marked.

4. If the start variable is not marked, *accept*; otherwise, *reject*."

زبان‌های تصمیم‌پذیر

- مثال آخر

THEOREM 4.9

Every context-free language is decidable.

PROOF Let G be a CFG for A and design a TM M_G that decides A. We build a copy of G into M_G. It works as follows.

M_G = "On input w:

1. Run TM S on input $\langle G, w \rangle$.
2. If this machine accepts, *accept*; if it rejects, *reject*."

تصمیم‌ناپذیری

- تاکنون ماشین تورینگ را به عنوان مدلی برای یک کامپیوتر (محاسبه کننده) عام بیان کرده‌ایم و همچنین مفهوم الگوریتم را به طور دقیق معرفی کرده‌ایم (تز چرچ-تورینگ).

- مسائل تصمیم‌پذیری دیدیم.

- برای چه مسائلی الگوریتم نداریم؟
- آیا صرفا برخی مسائل تئوری هستند؟

تصمیم‌ناپذیری

- اکنون قصد داریم نشان دهیم برخی مسائل حل نشدنی هستند (الگوریتمی برای آنها نداریم).
- اغلب این مسائل را میتوان به سادگی بیان کرد.
- مثال

زبان‌های تصمیم‌ناپذیر

THEOREM 12.8

There exists no algorithm for deciding whether any given context-free grammar is ambiguous.

- این یک مسئله حل‌ناپذیر است. ماشین تورینگی نداریم که برای هر ورودی دلخواه (CFG) متوقف شود.

تصمیم‌ناپذیری

Why should you study unsolvability? After all, showing that a problem is unsolvable doesn't appear to be of any use if you have to solve it. You need to study this phenomenon for two reasons. First, knowing when a problem is algorithmically unsolvable *is* useful because then you realize that the problem must be simplified or altered before you can find an algorithmic solution. Like any tool, computers have capabilities and limitations that must be appreciated if they are to be used well. The second reason is cultural. Even if you deal with problems that clearly are solvable, a glimpse of the unsolvable can stimulate your imagination and help you gain an important perspective on computation.

زبان‌های تصمیم‌ناپذیر

$$A_{\mathsf{TM}} = \{\langle M, w\rangle |\ M \text{ is a } \mathsf{TM} \text{ and } M \text{ accepts } w\}.$$

$$HALT_{\mathsf{TM}} = \{\langle M, w\rangle |\ M \text{ is a } \mathsf{TM} \text{ and } M \text{ halts on input } w\}.$$

$$E_{\mathsf{TM}} = \{\langle M\rangle |\ M \text{ is a } \mathsf{TM} \text{ and } L(M) = \emptyset\}.$$

$$REGULAR_{\mathsf{TM}} = \{\langle M\rangle |\ M \text{ is a } \mathsf{TM} \text{ and } L(M) \text{ is a regular language}\}.$$

$$EQ_{\mathsf{TM}} = \{\langle M_1, M_2\rangle |\ M_1 \text{ and } M_2 \text{ are } \mathsf{TM}\text{s and } L(M_1) = L(M_2)\}.$$

$$ALL_{\mathsf{CFG}} = \{\langle G\rangle |\ G \text{ is a } \mathsf{CFG} \text{ and } L(G) = \Sigma^*\}.$$

زبان‌های تصمیم‌ناپذیر

- چگونه نشان دهیم یک زبان تصمیم‌پذیر نیست؟

زبان‌های تصمیم‌ناپذیر

- زبان زیر را در نظر بگیرید:

$$A_{\mathsf{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}.$$

THEOREM 4.11 ....

A_{TM} is undecidable.

زبان‌های تصمیم‌ناپذیر

IUT-ECE

- زبان زیر را در نظر بگیرید:

$$A_{\mathsf{TM}} = \{\langle M, w\rangle \mid M \text{ is a } \mathsf{TM} \text{ and } M \text{ accepts } w\}.$$

Note that this machine loops on input $\langle M, w\rangle$ if M loops on w, which is why this machine does not decide A_{TM}. If the algorithm had some way to determine that M was not halting on w, it could *reject* in this case. However, an algorithm has no way to make this determination, as we shall see.