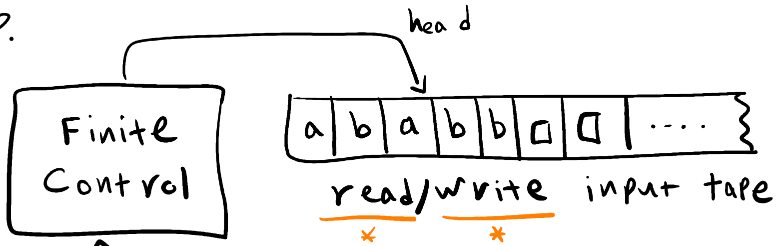
امید یعقوبی
جلسه‌ی 11و12 حل تمرین
گروه دوم



$\sqcup \notin \Sigma$

+ States
Transition Function

* می‌توان نمادهای روی نوار را تغییر داد. *
* هد می‌تواند به چپ و راست حرکت کند. *
* نوار از راست نامتناهی *
** می‌توان در هر زمان accept یا reject کرد. [لزومی ندارد که انتهای ورودی باشیم.] ← برخلاف DFA

مثال: $B = \{a^k b^k c^k \mid k \geq 0\}$

توصیف سطح بالا:

(۱) به سمت راست ورودی را اسکن کن تا به ☐ برسی و چک کن که ورودی به فرم $a^* b^* c^*$ باشد، در غیر این صورت reject کن. (No need to write / Finite Automaton)

(۲) به سمت چپ ترین خانه برگرد.

(۳) به سمت راست اسکن کن و در عین حال یک a، یک b و یک c خط بزن.

(۴) مرحله‌ی (۳) را تکرار کن، اگر در نهایت a و b و c خط خورده آخرین a و b و c بودند accept کن. اگر دست کم یکی از نمادها تمام شد و دوتای دیگر نبود، یعنی به فرم $a^k b^k c^k$ نیست. در این حالت reject کن.

$a \in \Sigma \implies a \in \Gamma, \cancel{a} \in \Gamma$ ← خط زدن

Yes No

تعریف رسمی: $M=(Q,\Sigma,\Gamma,\delta,q_0,q_{acc},q_{rej})$

Γ ⟵ Tape alphabet: $\Gamma \subseteq \Sigma$

$\delta: Q\setminus\{q_{acc},q_{rej}\}\times\Gamma \to Q\times\Gamma\times\{L,R\}$ [Partial *]

مثال: $\delta(q,a)=(r,b,R)$

ممکن است $\delta(p,c)$ تعریف نشده باشد.

Note: $\square \notin \Sigma$ و $\square \in \Gamma$ (Blank)

Note: روی ورودی w یک TM مانند M یا روی q_{acc} / q_{rej} متوقف می‌شود (Halt می‌کند) یا در Loop می‌افتد. [Halt نمی‌کند]

[Crash: روی $q\in Q$ که $q\neq q_{acc}$ و $q\neq q_{rej}$ متوقف شود [خوب نیست.]]

Accept w * ⟵ وارد q_{acc} می‌شود. $\forall a\in\Gamma.\ \delta(q_{acc},a)=\text{Undefined}$

Reject w by Halting ⟵ وارد q_{rej} می‌شود $\quad$ // $\delta(q_{rej},a)$ //

Reject w by Looping ⟵ اجرای w روی M تمام نشود.

NTM ⟵ $\delta: Q\times\Gamma \to \mathcal{P}(Q\times\Gamma\times\{L,R\})$

[$w\in L \iff$ $\exists$ Accepting branch In ComputatioTree T_w]

$L(M)=\{w\in\Sigma^* \mid M \text{ accept } w\}$

Deterministic

* زبان A Recursively Enumerable است، اگر ماشین M وجود داشته باشد که $A=L(M)$

[اصطلاحاً می‌گوییم M زبان A را تشخیص [Recognize] می‌دهد.]

→ Hopcraft

تعریف 9.2.1 ، زبان های recursive (decidable)

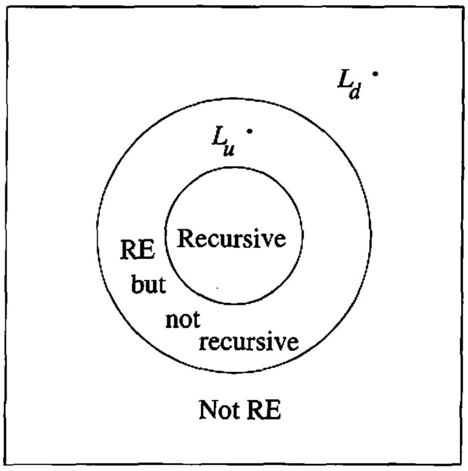
می گوییم زبان L تصمیم پذیر است اگر ماشین تورینگ M وجود داشته باشد که
$L = L(M)$ به شکلی که

① اگر $w \in L$ آن گاه M روی w ، accept کند [ودرنتیجه Halt می کند.]

② اگر $w \notin L$ آن گاه M روی w ، Halt کند [و به q_{rej} برود.]

A TM of this type corresponds to our informal notion of an "algorithm," a well-defined sequence of steps that always finishes and produces an answer. If we think of the language L as a "problem," as will be the case frequently, then problem L is called *decidable* if it is a recursive language, and it is called *undecidable* if it is not a recursive language.

P.373 edt 2001



توجه شود که معنای توقف به طور کلی گستره ی بزرگتری از توقف هاست.
ولی در این جا فرض می کنیم Halt همواره روی حالت های q_{acc} یا q_{rej} است.

Halting [در این جا] ← Halt را در این جا به معنای Halt روی q_{acc} یا q_{rej} در نظر می گیریم ، پس اگر به این معنا ماشین Halt کند ، در مورد $w \overset{?}{\in} L$ تصمیم گیری کرده است.

* هیچ مدل محاسباتی معقولی تا به حال پیدا نشده است که زبانی را توصیف کند که برای آن ماشین تورینگ وجود ندارد.

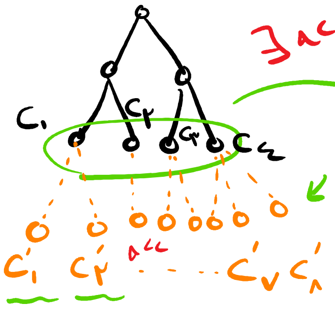
مثال از هم‌ارزی مدل‌های معقول (reasonable) محاسبه:

* شبیه‌سازی NTM با DTM ← ایده: BFS درخت محاسبه ← روش‌های پیاده‌سازی متفاوت

(Dovetailing)

← یک روش پیاده‌سازی BFS درخت محاسبه

$\exists acc?$



$\vdash \underbrace{q_i\,\dot{a}\,b \ldots}_{C_1} \# \underbrace{q_j\,\dot{b}\,b\,b \ldots}_{C_2} \# \ldots \underbrace{\# \ldots}_{C_3} \# \underbrace{q_k\,\dot{a}\,b\,a\,b \ldots}_{C_4} \# \square \ldots$

update

$\vdash \underbrace{q'_i\,\dot{a}\,a\,a \ldots}_{C'_1} \# \underbrace{\ldots - \ldots}_{C'_2 \# \ldots \# C'_v} - \# \underbrace{q'_j\,b\,a\,\dot{a} \ldots}_{C'_n} \#$

search for accepting branch

Question: Is C'_i an accepting Configuration $i \in \{1 \ldots n\}$

$b \in \Sigma \Rightarrow \dot{b} \in \Gamma$

$q \in Q \Rightarrow q \in \Gamma$

forking: ← اگر یک C_i، M انتخاب بعدی داشت M، کپی از آن ایجاد می‌کنیم و تغییرات را روی کپی‌ها اعمال می‌کنیم.

Defn: Configuration (Instantaneous Description)

← $\alpha q \beta$

← محتویات نوار ← $\alpha \beta \square^{\infty}$

← حالت فعلی ← q

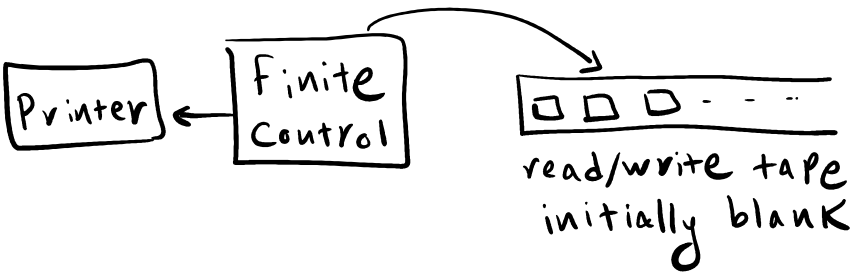
← Head Position ← اولین حرف β

(q, α, β)

Defn: Initial Configuration روی $w \in \Sigma^*$ ← (q_0, ε, w)

$$L(M) = \{ w \in \Sigma^* \mid \underbrace{(q_0, \varepsilon, w)}_{\text{Initial Configuration}} \vdash^* \underbrace{(q_{acc}, \alpha, \beta)}_{\text{Accepting Configuration}},\ \{\alpha, \beta\} \subseteq \Gamma^* \}$$

Enumerator



No input!

Defn: یک Enumerator یک $\underline{DTM}$ است که به یک Printer مجهز شده. روی نوار خالی شروع به کار می کند. هم چنین می تواند رشته هایی مانند $w_1, w_2, \ldots$ را چاپ کند و می تواند چاپ این رشته ها پایان ناپذیر باشد. [مثلا: $\delta(q,a) = (r,b,L,c)$

$c \longrightarrow$ output (write only)

زبان یک Enumerator: رشته هایی که آن ها را چاپ کرده.

$$E = (Q, \Sigma, \Gamma, \delta, q_0, q_{print}, q_{halt})$$

$$\hookrightarrow \delta: Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L,R\} \times \Sigma_\varepsilon$$

Note: $w \in L(M)$ می تواند به هر ترتیبی چاپ شود.

Note: $w \in L(M)$ می تواند بیش از یک بار چاپ شود.

$$\text{enumerator} \longrightarrow \underset{L(M)}{\underset{\cap}{w_1}} \# \underset{L(M)}{\underset{\cap}{w_2}} \# \cdots\cdots$$

$$L(E) = \{ w \in \Sigma^* \mid E \text{ prints } w \}$$

Note: Enumerator یک generator است نه یک Recognizer

$\hookrightarrow w \overset{?}{\in} L \begin{cases} w \in L & \text{yes} \\ w \notin L & \begin{cases} \text{No} \\ \text{Loop} \end{cases} \end{cases}$

$\hookrightarrow w \in L$ iff M prints w in some point

Thm : $A \in RE \iff A = L(E)$ که یک E Enumerator است.

Pf. $\Rightarrow$ فرض کنیم برای A یک Enumerator بنام E داریم.
ماشین M که Recognizer است را به شکل زیر تعریف می‌کنیم.

M = on input x:
1. Simulate E.
2. Whenever E print y, test $y \stackrel{?}{=} x$.
3. Accept if $y = x$

Note : اگر $w \notin A$ آنگاه M در Loop می‌افتد.

$\Leftarrow$ فرض کنیم M یک Recognize برای A است. یک Enumerator
با نام E برای A می‌سازیم.
E باید هر w را که M آن را accept می‌کند چاپ کند.

Twist : اگر M را به صورت Naive روی هر $w \in \Sigma^*$ شبیه‌سازی کنیم
ممکن است برای یک $w \in \Sigma^*$ ماشین M در Loop بیفتد.

Trick : Dovetailing !

ε را تا ۱ قدم شبیه‌سازی کن
روی M

Lexicographic

	Step۱	Step۲	Step۳	...	Step n
ε	(ε,۱)	(ε,۲)	(ε,۳)		
0	(0,۱)	(0,۲)	(0,۳)	..	
1	. .	.	. - -	..	
00	.	. .	.	.	
01					
⋮					

$\boxed{\text{If } \exists k \ (w, k) \text{ accept} \Rightarrow \text{Print } w}$

$w_i \in \Sigma^*$

۰ را تا ۳ قدم روی M شبیه‌سازی کن

سوال: اگر E را مطابق با روش اثبات ص قبل از روی M بسازیم، آیا $w \in L(E)$ ها با ترتیب lexicographic چاپ می‌شوند؟

نه. (٤, ١٥) و (٥١, ٢) [به تعداد قدم‌ها هم وابسته است.]

سوال: اگر در سوال بالا M را یک decider بگیریم، آیا می‌توان E را به شکلی ساخت که $w \in L(M)$ را با ترتیب Lexicographic چاپ کند؟

بله. $\{\varepsilon, 0, 1, \dots\}$ را به ترتیب روی M اجرا کنیم.

سوال: اگر یک Enumerator به نام E داشته باشیم که $w \in L(E)$ را با ترتیب lex چاپ می‌کند، آیا یک decider به نام M وجود دارد که $L(M) = E$؟

[برعکس بالا]. بله. اجرا می‌کنیم تا زمانی که $w_i <_{lex} x <_{lex} w_j$.

$\boxed{\text{Computation}}$ $\overset{?}{=}$ $\boxed{\text{Turing machine}}$

Proved

$\boxed{\lambda\text{-calculus}}$

Proved

$\boxed{\text{Recursive Functions}}$

reasonable models of computation ∞

Church-Turing thesis

CT

* Intuitive *

CT: Relation

* Formal *

ارتباط بین مفهوم محاسبه و فرمال‌سازی‌هایی که از آن وجود دارد.

جمله‌ی "اگر CT را بپذیریم آن‌گاه ..."

justify

* می‌توان آن را با شواهد فعلی توجیه کرد.

Note: مفهوم محاسبه شهودی است. پس CT غیرقابل اثبات است.

درستی ارتباط بین یک مفهوم شهودی و یک فرمال‌سازی از آن اثبات‌ناپذیر است.

توصیف سطح بالا: با پذیرش CT داریم:

« روی ورودی w

[توصیف فارسی الگوریتم بدون ابهام و با استفاده از اعمال به اندازه ی کافی پایه ای و پایان پذیر.»]

به سمت راست اسکن می کنیم تا به □ برسیم. مثلا

~~اگر M روی w متوقف نشود آن گاه...~~ ← نمی تواند decider باشد.

مثال: $L = \{a^k b^k c^k \mid k \geq 0\}$ [توصیف سطح بالا]

« روی ورودی w

1- چک کن $w \in a^* b^* c^*$ باشد، اگر نبود reject کن.

2- با استفاده از سه شمارنده تعداد a ها، b ها و c ها را بشمار.

3- اگر سه شمارنده در انتها مقادیری برابر داشتند، رشته را Accept کن در غیر این صورت reject کن.»

ALERT: اگر صورت سوال توصیف دقیق خواسته نمی توانید از توصیف سطح بالا استفاده کنید.

$A_{DFA} = \{\langle D, w\rangle \mid D \text{ is a DFA which accepts } w\}$

Thm: $A_{DFA} \in R$

Pf. "روی ورودی w

① چک کن $w \stackrel{?}{=} \langle D, w\rangle$ اگر نبود reject. [syntax]

② D را روی w شبیه سازی کن و جواب را خروجی بده. "

ورودی → $\underbrace{(Q, \Sigma, \delta, q_0, F)}_{D}, \underbrace{01\cdots1}$ ← Detail

متغیرها: $q_{current}$ و index

با δ دو متغیر را تا $w_{index=|w|}$ به روز می کنیم

Thm: $A_{NFA} \in R$

Pf. با استفاده از powerset construction یک نمونه از مسئله A_{NFA} را به نمونه ای از مسئله A_{DFA} تبدیل می کنیم. [note: powerset construction پایان پذیر است.] سپس الگوریتم بالا را روی آن اجرا می کنیم.

Thm: $E_{DFA} \in R$

ایده ی اثبات: آیا مسیری از q_0 به یک $q_f \in F$ وجود دارد.

[مسئله ی Graph ⇜ DFS]

$EQ_{DFA} = \{\langle D_1, D_2\rangle \mid D_1, D_2 \text{ are DFAs and } L(D_1) = L(D_2)\}$

$EQ_{DFA} \in R$

ایده اثبات .

$L(D_1) = L(D_2)$ iff $L(D_1) \Delta L(D_2) = \emptyset$

ساخت $A \Delta B$ ، DFA $\leadsto$ ① $A \Delta B = (A \cap \overline{B}) \cup (B \cap \overline{A})$

② $E_{DFA} \in R$ $\leadsto \underline{\langle D_{A\Delta B}\rangle \in E_{DFA}}$

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

Thm: $A_{TM} \notin$ Recursive

با برهان خلف فرض کنیم A_{TM} تصمیم پذیر است. پس وجود دارد ماشین تورینگ H به شکلی که $L(H) = A_{TM}$. پس داریم :

$$H \text{ on } \langle M, w\rangle \begin{cases} \text{Accept} & M \text{ accept } w \\ \text{Reject} & \text{o.w} \end{cases}$$

D = " On input $\langle M\rangle$

1. Simulate H on $\langle M, \langle M\rangle\rangle$
2. Accept if H rejects. Reject if H accept."

* D accepts $\langle M\rangle$ iff M doesn't accept $\langle M\rangle$.
* D accepts $\langle D\rangle$ iff D doesn't accept $\langle D\rangle$. $\times$

<u>Contradiction.</u>

Diagonalization

TMs	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	...	$\langle D \rangle$
M_1	acc	rej	acc	rej	...	
M_2	rej	acc	rej	rej	...	
M_3	acc	acc	acc	acc	...	
M_4	rej	acc	acc	rej	...	
⋮			⋮			
D	rej	rej	rej	acc	...	$\boxed{?}$

$A_{TM} \notin R$

$\{\langle M, w\rangle, M \text{ does not acc } w\} = \overline{A_{TM}} \notin RE$: Corollary

$\{L, \overline{L}\} \in RE \Rightarrow L \in R$: Pf. Idea

مفهوم کاهش پذیری: برای دو زبان A و B می‌گوییم A به B کاهش پذیر است یعنی می‌توان با استفاده از B، A را حل کرد.

مثال: محاسبه‌ی مساحت مستطیل قابل کاهش به مسائل اندازه‌گیری به ضلع‌های آن است.

مثال: A_{NFA} قابل کاهش به A_{DFA} است. کافی است NFA ورودی را به DFA متناظر تبدیل کنیم.

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

* A به B کاهش پذیر است اگر از حل B بتوان به حلی برای A رسید.

کاهش A_{TM} به E_{TM}، فرض کنیم E_{TM} دارای decider با نام R است.

S = "on input $\langle M, w\rangle$

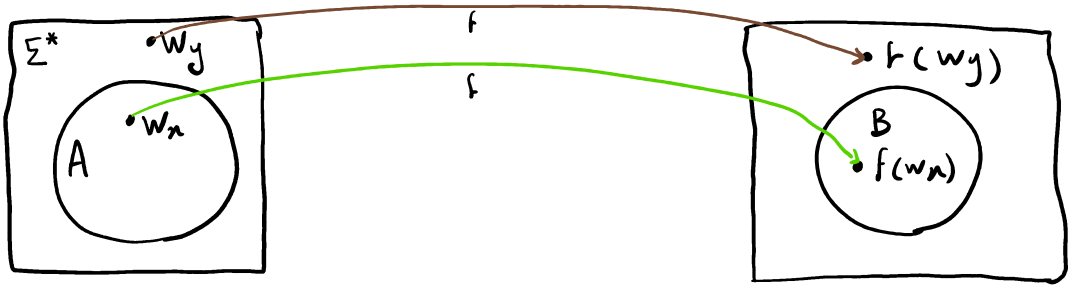
1. Transform M to new TM M_w = "on input x
   1. If $x \neq w$ rej
   2. else: run M on w
   3. Accept if M Accepts."
2. Use R to test whether $L(M_w) = \emptyset$
3. If "yes" then reject.
   If "No" then accept.

تعریف: تابع $f : \Sigma^* \to \Sigma^*$ محاسبه پذیر (Computable) است اگر TM با نام M_F داشته باشیم که M_F روی هر ورودی w توقف کند و f(w) را روی نوار بنویسد.

تعریف: می‌گوییم «A is mapping-reducible to B» است و

می‌نویسیم $A \leqslant_m B$ اگر تابع محاسبه‌پذیر f وجود داشته باشد که

$$w \in A \iff f(w) \in B$$



$$f : \langle M, w\rangle \longmapsto \langle M_w\rangle$$

$\in E_{TM}$

مثال: $A_{TM} \leqslant_m E_{TM}$ ⟵ f is $f(\langle M,w\rangle) = \langle M_w\rangle$

M_w on input x:
1. $x \neq w$ rej
2. run M on w
3. Accept if M Accept
4. o.w reject

$$\langle M,w\rangle \in A_{TM} \iff \langle M_w\rangle \in E_{TM}$$

Thm: $A \leqslant_m B$, $B \in R \Rightarrow A \in R$

Pf. S = "On input w:
1. Compute $f(w)$
2. Run M_B on $f(w)$
3. If M_B halts then output same result"

نتیجه: اگر $A \leq_m B$ و $A \notin R$ آنگاه $B \notin R$

Thm: اگر $A \leq_m B$ و $B \in RE$ آنگاه $A \in RE$

[اثبات مانند قبل]

نتیجه: اگر $A \leq_m B$ و $A \notin RE$ آنگاه $B \notin RE$

سوال: اگر $A \leq_m B$ آیا می‌توان نتیجه گرفت که $\overline{A} \leq_m \overline{B}$ ؟

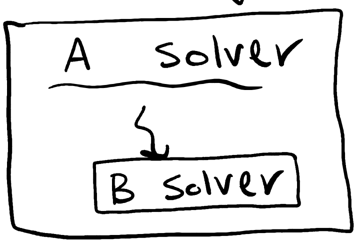
به روشنی بله: چون $[w \in A \iff f(w) \in B] \iff [w \notin A \iff f(w) \notin B]$

Note: Reduction یک مفهوم عمومی است و Mapping Reduction یک نمونه از Reduction عمومی است.

$$\leq_m \Longrightarrow \leq_{\underset{\text{oracle}}{T}}$$

(General) Reducibility of A to B:

* Use B solver to solve A *



Note: الزاماً هر نمونه از مسائل A به B ترجمه نمی‌شود.

شروع جلسه‌ی 12

[Mapping Reduction را نوعی ترجمه در نظر بگیریم]

* A به $\bar{A}$ کاهش پذیر است. [مفهوم عمومی کاهش پذیری]

* لزوماً A به $\bar{A}$ Mapping Reducible نیست.

مثال: $\overline{A_{TM}} \not\le_m A_{TM}$:

برای اولی، اگر بتوانیم $\bar{A}$ را حل کنیم به سادگی می‌توان جواب را برعکس کرد. [اگر حل به معنای Decide باشد.]

$\langle M,w\rangle \notin A_{TM}$ { rej simmulation / Loop

RE

ولی اگر با برهان خلف فرض کنیم $\overline{A_{TM}} \le_m A_{TM}$ ، آن گاه $A_{TM} \in R$.

$A_{TM} \notin R$ ✗

pf. می‌دانیم $A_{TM} \in RE$ [شبیه‌سازی به کمک UTM]، با فرض $\overline{A_{TM}} \le_m A_{TM}$ و بسته بودن RE نسبت به $\le_m$. $\overline{A_{TM}} \in RE$ پس

$\{\overline{A_{TM}}, A_{TM}\} \subseteq RE \Rightarrow A_{TM} \in R$ ✗. که این تناقض است. $A_{TM} \in R$

آیا $A_{TM} \leq_m \overline{A_{TM}}$ ؟ خیر.

ادعا: $A_{TM} \not\leq_m \overline{A_{TM}}$.

Pf. با برهان خلف فرض کنیم $A_{TM} \leq_m \overline{A_{TM}}$. می‌دانیم که

$A \leq_m B$ داریم $\overline{A} \leq_m \overline{B}$. پس $\overline{A_{TM}} \leq_m \overline{\overline{A_{TM}}}$

$\Leftarrow$ $\overline{A_{TM}} \leq_m A_{TM}$. که این تناقض است.

Terminology : $A \not\leq_m B$ و $B \not\leq_m A$ ، می‌گوییم

A و B ، incomparable هستند.

سوال: می‌دانیم که اگر $A \leq_m B$ و $B \in RE$ آن‌گاه $A \in RE$.
آیا می‌توان این حکم را برای مفهوم عموم کاهش هم نشان داد؟

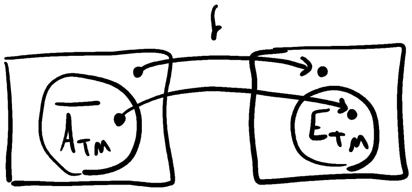
خیر. با برهان خلف فرض کنیم چنین باشد. می‌دانیم که $\overline{A_{TM}}$ قابل کاهش به A_{TM} است. چرا که اگر حلی برای A_{TM} داشته باشیم می‌توانیم جواب آن را به روشنی Flip کنیم. هم‌چنین A_{TM} یک زبان T-Recognizable است. پس $\overline{A_{TM}} \in RE$، که این تناقض است.

Reminder: $E_{TM} = \{\langle M\rangle \mid M \text{ is a TM and } L(M)=\emptyset\}$

Thm: می‌خواهیم نشان دهیم $E_{TM} \notin RE$.

کلاس RE نسبت به $\leq_m$ بسته است.

Pf. $\overline{A_{TM}} \leq_m E_{TM}$.



$f: \langle M,w\rangle \longmapsto \langle M_w\rangle$

$\langle M,w\rangle \in \overline{A_{TM}} \iff \langle M_w\rangle \in E_{TM}$ *

f, $\langle M,w\rangle$

TM M_w = " On input x

1. $x \neq w \rightsquigarrow$ reject
2. Run M on w
3. Accept if M accepts."

Σ^*

$\emptyset$

اگر $\langle M_w\rangle \in E_{TM}$ آن‌گاه $\langle M,w\rangle \in \overline{A_{TM}}$

$\langle M_w\rangle \in E_{TM} \Leftarrow \langle M,w\rangle \in \overline{A_{TM}}$

$L(M_w)=\emptyset \Leftarrow \underset{\text{rej}}{\overset{\text{Loop}}{}}$ (خط 2) $\leftarrow \langle M,w\rangle \notin A_{TM}$

$\langle M_w\rangle \in E_{TM} \Leftarrow \langle M,w\rangle \in \overline{A_{TM}}$

$EQ_{TM} = \{\langle M_1, M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$

Thm: هر دوی $\overline{EQ_{TM}}$ و EQ_{TM} ، T-unrecognizable هستند.

ادعای (1): $\overline{A_{TM}} \leq_m EQ_{TM}$

ادعای (2): $\overline{A_{TM}} \leq_m \overline{EQ_{TM}}$

برای هر $w \in \Sigma^*$ ماشین M_w این گونه تعریف می‌شود:

$\forall w \in \Sigma^* \rightsquigarrow M_w =$ "On input x
1. ignore x
2. Simmulate M on w."

M_{reject} : ماشین تورینگی است که هر ورودی را reject می‌کند.

اگر $\langle M,w\rangle \notin A_{TM}$ آنگاه $L(M_w) = L(M_{reject})$

اگر $\langle M,w\rangle \in A_{TM}$ آنگاه $L(M_w) \neq L(M_{reject})$

پس $\langle M,w\rangle \in \overline{A_{TM}} \iff \langle M_w, M_{reject}\rangle \in EQ_{TM}$

پس $\overline{A_{TM}} \leq_m EQ_{TM}$. پس $EQ_{TM} \notin RE$

* به همین ترتیب M_{accept} تعریف می‌کنیم.

$\langle M,w\rangle \in \overline{A_{TM}} \iff \langle M_w, M_{accept}\rangle \in \overline{EQ_{TM}}$

پس $\overline{A_{TM}} \leq_m \overline{EQ_{TM}}$ و در نتیجه $\overline{EQ_{TM}} \notin RE$

سوال: نشان دهید $A \in R$ iff $A \leq_m 0^*1^*$

$\Leftarrow$: چون $A \leq_m 0^*1^*$ و $0^*1^* \in R$ پس $A \in R$

$\Rightarrow$: از آن جا که $A \in R$ پس TM با نام M داریم که $L(M)=A$ و M یک Deterministic Decider است.

پس

$$f(w) \begin{cases} 01 & w \in A \\ 10 & w \notin A \end{cases}$$

Note: $w \overset{?}{\in} A$ توسط M انجام می‌شود و M یک Decider است. پس f محاسبه پذیر است.

$A \leq_m 0^*1^*$

سوال: $\underset{\sim}{A \in RE} \iff \underset{\sim}{A \leq_m A_{TM}}$

$\Leftarrow$ $A_{TM} \in RE$ پس $A \in RE$

$\Rightarrow$ $A \in RE$ پس یک Recognizer با نام M برای آن وجود دارد.

پس تعریف می‌کنیم $f(w) = \langle M, w \rangle$

recognized by M

$w \in A \iff \langle M, w \rangle \in A_{TM}$

سوال: $\underset{\sim}{L_K \overset{?}{\in} R}$ $\leftarrow$ $L_K \in R$

$L_K = \{\langle M, w, K \rangle \mid \underline{M \text{ uses at most } K \text{ tape square on input } w}\}$

$\#\text{Configurations} \leq \underbrace{|Q| \times |\Gamma|^K \times K}_{\alpha_K}$

decider $\leftarrow$ M'_K

- شبیه‌سازی را تا α_K قدم انجام می‌دهیم
- محدود کردن نوار ماشین مورد شبیه‌سازی (M) به K خانه با استفاده از

$\vdash \notin \Gamma$ [M الفبای Γ]

اگر $\underline{\text{Loop} \leq K}$ [abuse Notation]، تعداد کانفیگوریشن‌های

حتما نیز برابر α است. درنتیجه اگر اجرای M روی w بیش
از α طول کشید طبق اصل لانه کبوتری یک کانفیگوریشن تکرار
شده و loop داریم و acc می‌کنیم.

سوال: $L_{FinMem} \overset{?}{\in} R$

$L_{FinMem} = \{ \langle M,W\rangle \mid$ Does there exist a K s.t M uses at most K tape squares on input w $\}$

Halting Problem
$\leq$

جواب. $L_{fm} \notin R$.

ادعا: $L_{fm} \in R \Rightarrow HP \in R$ ✗ پس $L_{fm} \notin R$

Deterministic

مشاهده: اگر $L_{fm} \in R$ پس یک Decider با نام M' داریم که آن را تصمیم می‌گیرد. اگر $\langle M,w\rangle \notin L_{fm}$ پس M روی w در loop افتاده. [در غیر این صورت ماشین Halt کرده (روی w) و محتوای نوار در آخرین کانفیگوریشن را در نظر بگیرد. $|C_f| = K$]

M_{HP}:

" روی ورودی w

① اگر $w \neq \langle M,w\rangle$ ، reject

② اگر $\langle M,W\rangle \notin L_{fm}$ [به کمک M' بازش خلف]، reject

③ در غیر این صورت K را به روش زیر پیدا می‌کنیم:

foreach $i \in \{1, 2, \dots \underline{K}\}$ $\begin{cases} \text{Run TM } M'_K \text{ on } \langle M,w,i\rangle \\ \text{If accept } K = i \\ \text{If reject } i = i+1 \end{cases}$

④ α_K را حساب کن و ماشین M را روی w تا α_K شبیه‌سازی کن

اگر تا α_K، Halt کرد accept در غیر این صورت reject

$$L_{f_c} := \left\{ \text{on input } w : \begin{cases} \text{If } |Loop(L(M_{1000}))| \text{ is } finite \text{ Output: } |Loop(L(M_{1000}))| \\ \text{If } |Loop(L(M_{1000}))| = \infty \text{ Output: } -1 \end{cases} \right.$$

محاسبه‌گر

$Loop(L(M)) = \{w \in \Sigma^* \mid M$ does not halt on input $w\}$ سوال $L_{F_c} \overset{?}{\in} R$

$$\left.\begin{array}{l} |Loop(L(M_{1...}))| = K \in \mathbb{N} \ (1) \\ |Loop(L(M_{1...}))| = \infty \ (2) \end{array}\right\}$$ ۲ حالت داریم

در هر دو حالت F_c محاسبه‌پذیر است. [Note: نیازی به تست $w \overset{?}{\in} Loop(L(M_{1...}))$ نیست.]

F_c ورودی را ignore می‌کند و یک ثابت است را می‌نویسد.

$$L_{tat} := \left\{ \text{on input } w : \begin{cases} \text{If } \langle M_{1000}\rangle \in Halt(L(M_{1000})) \text{ Accept} \\ \text{If } \langle M_{1000}\rangle \notin Halt(L(M_{1000})) \text{ Reject} \end{cases} \right.$$

$Halt(L(M)) = \{w \in \Sigma^* \mid M$ Halts on input $w\}$. سوال: $L_{tat} \overset{?}{\in} R$

$$\left.\begin{array}{ll} L_{tat} = \Sigma^* \Leftarrow & \langle M_{1...}\rangle \in Halt(L(M_{1...})) \ (1) \\ L_{tat} = \emptyset & \langle M_{1...}\rangle \notin Halt(L(M_{1...})) \ (2) \end{array}\right\}$$ دو حالت داریم

در هر دو حالت $L_{tat} \in R$.

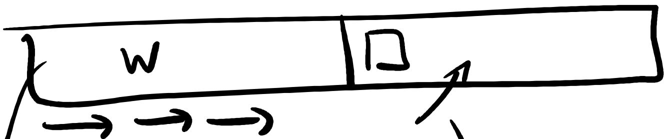
$L_{EpsilonLeft} = \{\langle M\rangle | M \text{ ever moves left while computing } \epsilon\}$ need $\in R$ ✓

کافیست M را تا ۱+|Q| قدم روی ε شبیه سازیم.

سوال: $L_{EpsilonLeft} \overset{?}{\in} R$

ادعا: هر TM با نام M روی ورودی w اگر حرکت به چپ داشته باشد، این حرکت در دست بالا در $|W|+|Q|+1$ قدم شبیه سازی انجام می شود.

رایس
↓
خصوصیت معنایی
↓
زبان M



$\underbrace{\delta(q_i, \square)}, \; q_i \in Q \Rightarrow |Q|+1$
$|W|$

$\underbrace{\delta(q_i, a)} \longrightarrow$ اگر دائم به راست برود w را پشت سر ما گذارد [پس از |w| قدم]

$L_2 = \{\langle M\rangle : |L(M)| \geq 2\}$

سوال: $L_2 \overset{?}{\in} RE$

$L_2 \in RE$. با استفاده از Dovetailing

M'

Lexicographic

	Step1	Step2	Step3	...	Step n
ε	(ε,1)	(ε,2)	(ε,3)		
0	(0,1)	(0,2)	(0,3)	..	
1	..	.	. . -	. .	
00	.	. .	. .	.	
01					

$\langle M\rangle \in L_2 \Rightarrow$ Halt.accept

$\langle M\rangle \notin L_2 \Rightarrow M'\ \langle M\rangle$ Reject by Looping

$L_{2'} = \{ \langle M \rangle : |L(M)| \leqslant 2 \}$

سوال : $L_{2'} \overset{?}{\in} RE$ [می‌دانیم $\overline{HP} \notin RE$]

ادعا: اگر $L_{2'} \in RE \Leftarrow \overline{HP} \in RE$ ، زیرا ادعا $\leadsto$ $\overline{HP} \leqslant_m L_{2'}$

$f : \langle M, x \rangle \mapsto \langle M' \rangle \qquad \langle M, x \rangle \in \overline{HP} \iff \langle M' \rangle \in L_{2'}$

M' :

" روی ورودی w
① ورودی را پاک می‌کند
② M را روی x شبیه‌سازی می‌کند
③ اگر M متوقف شد accept . "

$$L(M') = \begin{cases} \emptyset \\ \Sigma^* \end{cases}$$

$\langle M, x \rangle \in \overline{HP} \implies$ M روی x در loop می‌افتد

$\implies$ M' هیچ ورودی‌ای را نمی‌پذیرد.

$\implies L(M') = \emptyset \implies |L(M')| = 0$

$\implies |L(M')| \leqslant 2 \implies \langle M' \rangle \in L_{2'}$

Contrapositive (عکس نقیض)

$\langle M, x \rangle \notin \overline{HP} \implies$ M روی x حتماً Halt می‌کند

$\implies$ M' همه‌ی ورودی‌ها را می‌پذیرد

$\implies L(M') = \Sigma^* \implies |L(M')| = \infty$

$\implies |L(M')| > 2 \implies \langle M' \rangle \notin L_{2'}$

$L_{Fin} = \{\langle M\rangle : L(M) \text{ is Finite}\}$

سوال $L_{Fin} \overset{?}{\in} RE$

اثبات مانند سوال قبل. $[\emptyset , \Sigma^*]$

$L_c = \{\langle M\rangle \mid M \text{ is a TM}, \exists w\in\Sigma^* \text{ s.t. } M \text{ on } w \text{ halts at most in } |\langle M\rangle| \text{ steps}\}$

سوال $L_c \overset{?}{\in} R$

$L_c \in R$. تمام رشته های ما به شکلی که $|w| \le |\langle M\rangle|$ تا $|\langle M\rangle|$ قدم شبیه سازی کنیم. درصورتی که دست کم یکی از آن ها Halt کرد، accept می کنیم.

Twist : اگر $c = |\langle M\rangle|$ ، از کجا معلوم رشته‌ی w وجود نداشته باشد، $|w| > c$ و M در c قدم روی w متوقف شود؟

اگر روی w در کمتر از c قدم Halt کند آن گاه

$$w = \underbrace{a_1 a_2 \cdots a_c}_{w'} \underbrace{a_{c+1} \cdots a_{|w|}}_{w''}$$

روی w' که $|w'| = c$ در کمتر مساوی c قدم Halt می شد.

$L_{100step} = \{\langle M \rangle \mid M$ is a TM and $\exists w \in \Sigma^*$, M on w halts at most in 100 step$\}$

سوال $L_{100step} \overset{?}{\in} R$.

اثبات. مانند سوال قبل.

فرض کنیم h یک homomorphism به شکل $h: \Sigma^* \to \Delta^*$ باشد.

فرض کنیم $L \in RE$ ، آیا $\underline{h(L) \in RE}$ ؟

hint : یک NTM برای $h(L)$ می‌توان ساخت. برای ورودی $x \overset{?}{\in} h(L)$.

① حدس w و بعد

✓ $\underline{h(w) = x}$ ①

✓ $\underline{\text{recognizer } L}$ ⟵ $w \overset{?}{\in} L$ ②

$\exists w \in \Sigma^* \longrightarrow x \in h(L)$

* فرض کنیم $L \in R$ ، آیا $h(L) \in R$ ؟ [نه]

$\underline{HP = \{\langle M, w \rangle \mid M \text{ halts on } w\}}$ → (Σ)

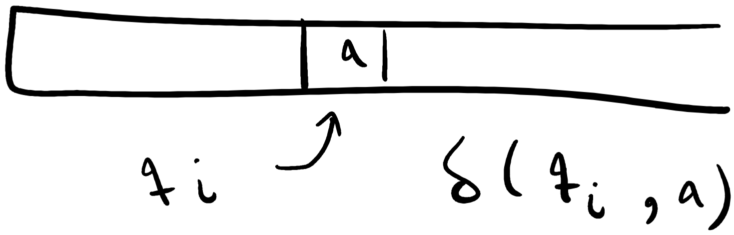
→ $\Sigma' = \Sigma \cup \{c\}$, $c \notin \Sigma$

$HP' = \{\langle M, w, c^i \rangle : M$ halts on w in at most i step$\}$

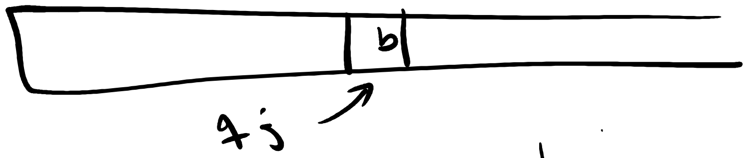
$$HP' \overset{?}{\in} R, \quad h(a) \begin{cases} a & a \in \Sigma \\ \varepsilon & a = c \end{cases} \Rightarrow h(HP') = HP$$

$HP' \overset{?}{\in}, HP \overset{?}{\in} R$

سوال: اگر به غیر از L و R، بی حرکتی یا stay را داشته باشیم قدرت محاسباتی افزایش پیدا می کند؟؟



$$\delta(q_i, a) = (b, q_j, S)$$



$$\delta(q_i, a) = \begin{cases} \text{① طا را بنویس برو به راست و برو به } q_s \notin Q \\ \text{② در } q_s \text{ برای هر ورودی برو به چپ} \end{cases}$$