Subject
Date

دو تعریف برای زبان یک PDA:

① $\mathcal{L}_1(P) := \{ \omega \in \Sigma^* \mid (q_0, \omega, z_0) \vdash^* (q, \Lambda, \alpha),\ q \in F,\ \alpha \in \Gamma^* \}$

② $\mathcal{L}_2(P) := \{ \omega \in \Sigma^* \mid (q_0, \omega, z_0) \vdash^* (q, \Lambda, \Lambda),\ q \in Q \}$

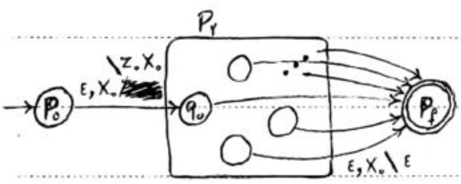
در برخی منابع، علامت کف stack را عضو Γ فرض می‌کنند لذا رسیدن به آن علامت در stack (یعنی هر زمان که بالای stack آن علامت بود.)

به معنای خالی بودن stack است. اما در متن ما (Martin)، $z_0 \in \Gamma$ است؛ لذا می‌توان در وسط stack هم z_0 داشت و لذا این که بالای stack

علامت z_0 باشد، به معنای کف stack نیست. لذا در این مواقع هنگامی به کف stack می‌رسیم که Λ در کف باشد.

می‌خواهیم نشان دهیم دو تعریف بالا به معنای زیر معادل اند:

$\forall P_1 : \exists P_2 : \mathcal{L}_1(P_1) = \mathcal{L}_2(P_2)$ (I) , $\forall P_2 : \exists P_1 : \mathcal{L}_2(P_2) = \mathcal{L}_1(P_1)$ (II)



برای نشان دادن (II)، با داشتن یک P_2، از روی آن یک P_1 معادل می‌سازیم:

P_2 را ماشینی با زبان $\mathcal{L}_2(P_2)$ فرض می‌کنیم و فرض می‌کنیم $X_0 \notin \Gamma_2$. P_1 را به صورت زیر تعریف می‌کنیم:

$P_1 := (Q_2 \cup \{P_0, P_f\},\ \Sigma_2,\ \Gamma_2 \cup \{X_0\},\ P_0,\ X_0,\ \{P_f\},\ \delta_1)$

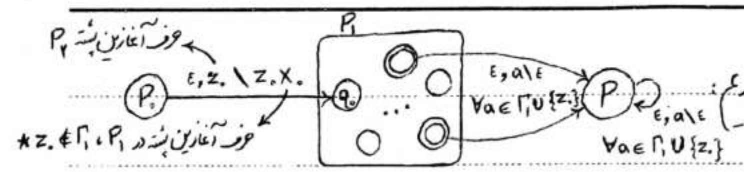
وضعیت آغازین حرف آغازین پشته وضعیت های پذیرش

و δ_1 را به صورت زیر تعریف می‌کنیم: $\forall q \in Q_2,\ a \in \Sigma_2,\ x \in \Gamma_2 : \delta_1(q, a, x) = \delta_2(q, a, x)$

$\delta_1(P_0, \varepsilon, X_0) = (q_0, z_0 X_0)$, $\forall q \in Q_2 : \delta(q, \varepsilon, X_0) = (P_f, \varepsilon)$

Subject
Date

برای نشان دادن ①، با داشتن یک P_1، از روی آن یک P_2 معادل می‌سازیم:



با توجه به شکل می‌توان P_2 را به طور دقیق برحسب P_1 تعریف کرد.

قضیه: به ازای هر PDA مانند P، یک گرامر مستقل از متن G وجود دارد که $\mathcal{L}(P)=\mathcal{L}(G)$ و برعکس.

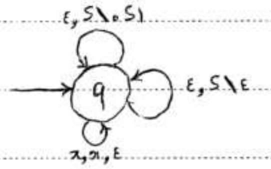
اثبات: $P \leftarrow G$ $\quad$ $G=(V,T,S,P)\ ,\ \forall(A\to B)\in P: A\in V, B\in (V\cup T)^*$

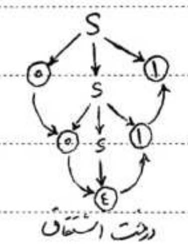
ماشین پشته‌ای P معادل با گرامر G را طبق تعریف $\mathcal{L}_2$، به صورت روبرو ارائه می‌دهیم: $P_1=(\{q\}, T, V\cup T, q, S, \delta)$

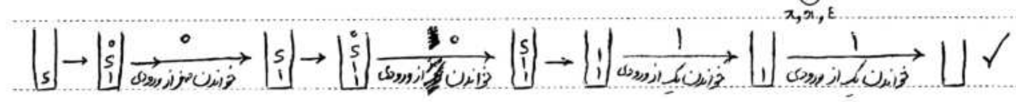
الفبای ورودی — الفبای پشته — وضعیت آغازین — حرف آغازین پشته

و تابع انتقال δ را چنین تعریف می‌کنیم: $\forall(A\to B)\in P: \delta(q,\varepsilon,A)=(q,B)\quad , \quad \forall x\in T: \delta(q,x,x)=(q,\varepsilon)$

مثال: $S\to 0S1 \mid \varepsilon \implies S \to 0S1 \to 00S11 \xrightarrow{\text{عضو گرامر}} 0011$







ثابت می‌کنیم $\mathcal{L}_2(P)=\mathcal{L}(G)$. فرض می‌کنیم $w\in\mathcal{L}(G)$. پس $S \vdash \gamma_1 \vdash \gamma_2 \vdash \dots \vdash \gamma_k \vdash w$ که $\gamma_i\in(V\cup T)^*$ وجود دارد که به گام‌های آن

سمت چپ ترین اشتقاق هستند؛ یعنی $\gamma_i \vdash \gamma_{i+1}$ که $\alpha A\beta \vdash \alpha B\beta$ و $(A\to B)\in P$ و $\alpha\in T^*$ و $\beta\in(V\cup T)^*$.

این قسمت نشان دهنده سمت چپ ترین اشتقاق است. معنای شهودی آن این است که فقط متغیری که در سمت چپ آن متغیر دیگری نیست، مشتق می‌شود

ادامه اثبات این قسمت ($P \leftarrow G$) به عهده شما! (با استفاده از استقرا روی تعداد گام‌های تولید رشته با گرامر)

Subject

Date

$G \leftarrow P$

فرض کنیم P یک PDA است. گرامر مستقل از متن G را می‌سازیم به گونه ای که: $\mathcal{L}(G) = \mathcal{L}_2(P)$

$P = (Q, \Sigma, \Gamma, q_0, z_0, \delta)$ $\quad\quad G := (V, T, S, P)$

$V = \{S\} \cup \{(q, a, q_0) \mid q, q_0 \in Q, a \in \Gamma\}$ $\quad\quad T = \Sigma$

$P:$ ⓪ $\forall q \in Q: S \rightarrow (q_0, z_0, q)$

① $\forall (P, B_1 B_2 \dots B_m) \in \delta(q, x, A), \forall q_2, \dots, q_{m+1} \in Q: (q, A, q_{m+1}) \rightarrow x (P, B_1, q_2)(q_2, B_2, q_3) \dots (q_m, B_m, q_{m+1})$

② $\forall q, q_1 \in Q, \forall a \in \Sigma \cup \{\Lambda\}, \forall A \in \Gamma, \forall (q_1, \Lambda) \in \delta(q, a, A): (q, A, q_1) \rightarrow a$

(و حذف می شود!)

deterministic PDA: $\forall q \in Q, x \in \Sigma \cup \{\varepsilon\}, A \in \Gamma: |\delta(q, x, A)| \leq 1, \forall x \in \Sigma: |\delta(q, x, A)| + |\delta(q, \varepsilon, A)| \leq 1$

این شرط جامع شرط قبلی است.

زبان های deterministic pushdown automata را DCFL (deterministic context-free language) می‌گوییم.

مثال: $\mathcal{L}_1 = \{w \mid N_0(w) = N_1(w)\}$

deterministic

$\delta(q_0, 0, z_0) = (q_1, 0z_0)$ $\quad \delta(q_0, 1, z_0) = (q_1, 1z_0)$ $\quad \delta(q_1, 0, 0) = (q_1, 00)$ $\quad \delta(q_1, 0, 1) = (q_1, \varepsilon)$

$\delta(q_1, 1, 1) = (q_1, 11)$ $\quad \delta(q_1, 1, 0) = (q_1, \varepsilon)$ $\quad \delta(q_1, \varepsilon, z_0) = (q_0, z_0)$ $\quad$ state های پذیرش: $\{q_0\}$

Subject

Date

$L_2 = \{ w \mid$ و در هر پیشوند از w تعداد ۰ ها نابیشتر از تعداد یک هاست , $N_0(w) = N_1(w) \}$

deterministic

پذیرش: $\{q_0\}$ $\delta(q_0, 1, z_0) = \delta(q_1, 1z_0)$ $\delta(q_1, 1, 1) = (q_1, 11)$ $\delta(q_1, 0, 1) = (q_1, \varepsilon)$ $\delta(q_1, \varepsilon, z_0) = (q_0, z_0)$

$L = \{ ww^R \mid w \in \{0,1\}^* \}$

اثبات DCFL نبودن L: ← قضیه 5.16 کتاب

فرض می‌کنیم تابع انتقالی DPDA وجود داشته باشد که معادل این زبان باشد. در این صورت تابع انتقال به صورت مجموعه زیر قابل بیان است:

$\delta(q, x, A) = (p, \varepsilon)$ یا $\delta(q, x, A) = (p, \alpha A)$ یا $\delta(q, x, A) = \text{undefined}$

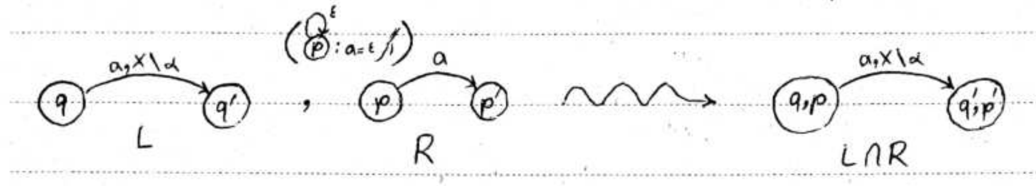
(در صورتی که داشتیم $\delta(q, x, A) = (p, B)$، state جدید q' را می‌سازیم و به جای رابطه قبلی، دو رابطه $\delta(q, x, A) = (q', \varepsilon)$ و $\forall T \in \Gamma: \delta(q', \varepsilon, T) = (p, BT)$)

v_w: رشته‌ای که پشته تشکیل شده در هنگام خواندن wv_w کمینه طول ممکن را در بین همه wv ها (همه رشته‌هایی که پیشوند w دارند) دارد.

PAPCO

Subject
Date

قضیه: اشتراک یک زبان مستقل از متن و یک زبان منظم، یک زبان مستقل از متن است. L CFL , R regular $\Rightarrow L\cap R$ CFL



همچنین اشتراک یک DCFL و regular، یک DCFL است.

- اشتراک دو CFL لزوماً CFL نیست. (اثبات جلسه بعد) مثال: $0^n1^n2^m \cap 0^m1^n2^n$

قضیه: مکمل یک DCFL، DCFL است.

زبان $L=\{0^n1^n2^n \mid n\geq 0\}$، CFL نیست، درحالی که مکمل آن CFL است. بنابراین CFL نسبت به مکمل گیری بسته نیست.

بررسی تصمیم پذیری چند مسئله: (یک مسئله تصمیم پذیر است اگر بتوان آن را با یک ماشین یا یک گرامر جواب داد.)

برای L (زبان) DCFL و R regular:

1) L CFL , R regular $\Rightarrow L\cap R$ CFL L DCFL , R regular $\Rightarrow L\cap R$ DCFL

2) L DCFL $\Rightarrow L'$ DCFL   ★(این رابطه برای CFL ها برقرار نیست.)

منظور از ✓ تصمیم پذیر بودن و منظور از X تصمیم ناپذیر بودن است.   ✓. $R\subseteq L$?   ✓. $L\subseteq R$?   ✓. $R=L$?

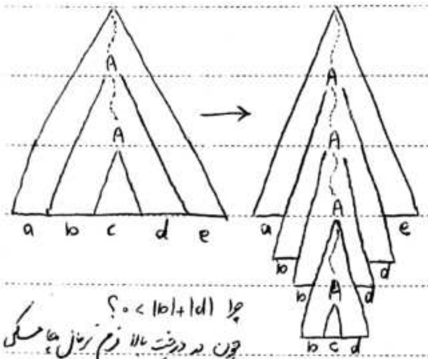
با بررسی گرامر یک CFL، می‌توان تهی بودن زبان آن را تصمیم گیری کرد. ⟵ $R=L \Leftrightarrow \underbrace{\underbrace{(L'\cap R)}_{\text{طبق 1 و 2 DCFL}}\cup\underbrace{(L\cap R')}_{\text{طبق 1 DCFL}}}_{CFL}=\emptyset$

X. $L\cup L_1$ is DCFL?   X. $L_1\cap L$ is CFL?   ✓. $L=\emptyset$? $\Rightarrow$ ✓. $L=\Sigma^*$?   ✓. is L regular?

X ∩   با بررسی گرامر   اثبات نمی‌کنیم. (اختیاری)

Subject
Date

Pumping Lemma: لم پمپاژ:



اگر L یک زبان مستقل از متن باشد، $n\in\mathbb{N}$ موجود است که هر $w\in L$ و $|w|>n$ را می‌توان به فرم $w=abcde$ نوشت که $0<|b|+|d|$ و $|bcd|<n$ و به ازای هر $i\in\mathbb{N}\cup\{0\}$:

$$ab^icd^ie\in L$$

چرا $0<|b|+|d|$؟ چون در درخت بالا فرم زبان‌ها مسکلی را در نظر می‌گیریم.

مثال: نشان می‌دهیم هیچ کدام مستقل از متن نیستند.

$L_1=\{0^n1^n2^n|n\geq 0\}$ $L_2=\{0^n1^m2^n3^m|n,m\geq 0\}$ $L_3=\{0^p|p \text{ is prime}\}$

L_1: n در لم پمپاژ را در نظر بگیرید: $0^n1^n2^n=abcde$ و $|bcd|<n$. چون bcd حداکثر شامل یک یا دو حرف از $\{0,1,2\}$ است بنابراین: $abcde\in L$: لم پمپاژ
* $ab^2cd^2e\notin L$

L_2: به طور مشابه با L_1

L_3: $p>n$ وجود دارد که $abcde=0^p$. می‌گیریم $i=|ace|$. در این صورت $|ab^icd^ie|=i(1+|bd|)$. بنابراین: $ab^icd^ie\notin L_3$ *

$L_4=\{ww^Rw'w'^R|w,w'\in\Sigma^*\}$: $w=0^n1^n\Rightarrow 0^n1^{2n}0^{2n}1^{2n}0^n\in L_4$ $L_5=\{ww|w\in\Sigma^*\}$; $w=0^n1^n\Rightarrow 0^n1^n0^n1^n\in L_5$

$L_6=\{0^i1^j2^k|i\leq j<k\}$ $L_7=\{0^i1^j2^k|i\neq j\neq k\neq i\}$?

L_6: $abcde=0^n1^{n+1}2^{n+2}$ اگر bcd شامل 0 و 1 باشد، آن را به توانی می‌رسانیم که از k بیشتر شود، اگر شامل 2 و 1 باشد، آن را به توان صفر می‌رسانیم تا کمتر از i شود.

مثال: $L=\{0^i1^j2^k|j\in\{i,k\}\}$: $L=0^i1^i2^k\cup 0^i1^k2^k$ بنابراین CFL است. می‌خواهیم نشان دهیم DCFL نیست. اگر DCFL باشد،

طبق گزاره 1 و 2، $L_7=\{0^i1^j2^k|j\notin\{i,k\}\}=L'\cap \underbrace{0^*1^*2^*}_{\text{regular}}$ ،DCFL است. اگر ثابت کنیم L_7، CFL نیست کافیست.

Subject
Date

$\forall b \in \Sigma^*: b^* :=$ تعداد حروف برچسب دار b

لم Ogden (تعمیم لم پمپاژ):

اگر L مستقل از متن باشد، عدد n موجود است که هر رشته به طول دلخواه و شامل حداقل n حرف برچسب دار (هر حرف را به دلخواه می توان برچسب دار کرد یا نکرد) مثل w را می توان به فرم $w=abcde$ نوشت که $b^*+d^*>0$ و $b^*+c^*+d^*<n$ که داریم: $\forall i: ab^icd^ie \in L$

ادامه مثال قبل با استفاده از لم Ogden:

می خواستیم نشان دهیم $L=\{0^i1^j2^k \mid j \notin \{i,k\}\}$، CFL نیست. اگر CFL باشد، n را مقدار به اندازه کافی بزرگ در لم Ogden می گیریم.

رشته $x=0^n1^{n+n!}2^{n+n!}$ در L قرار دارد. پس x را می توان به فرم $x=abcde$ نوشت به طوری که حداقل یکی از b و d شامل ۰ باشند
(* حروف برچسب دار: 0^n)

(چون فقط ۰ ها برچسب دارند) و $\forall i\geq 0: ab^icd^ie \in L$

اگر b یا d حاوی حداقل دو حرف مختلف از الفبا باشند به وضوح $ab^2cd^2e \notin L$ (اصلاً به فرم $0^*1^*2^*$ نیست.)

لذا چند حالت زیر ممکن است: ① $b\in 0^*$ و $d\in 0^*$ ② $b\in 0^*$ و $d\in 1^*$ ③ $b\in 0^*$ و $d\in 2^*$

حالت اول: $|b|+|d|=m<n: ab^{\frac{n!}{m}+1}cd^{\frac{n!}{m}+1}e=0^{n+n!}1^{n+n!}2^{n+2n!}\in L$ ※

حالت دوم: اگر $d=\varepsilon$ به حالت اول ارجاع می دهیم. $|d|=m<n: ab^{\frac{n!}{m}+1}cd^{\frac{n!}{m}+1}e=0^*1^{n+2n!}2^{n+2n!}\in L$ ※

حالت سوم: اگر $d=\varepsilon$ به حالت اول ارجاع می دهیم. $|b|=m<n: ab^{\frac{n!}{m}+1}cd^{\frac{n!}{m}+1}e=0^{n+n!}1^{n+n!}2^{*}\in L$ ※

- مثال روبرو نشان می دهد که یک CFL که DCFL نیست، ممکن است مبهم نباشد. بنابراین deterministic بودن به معنی غیرمبهم بودن نیست: $S\to 0S0\mid 1S1\mid \varepsilon$

ماشین تورینگ

$T = (Q, \Sigma, \Gamma, B, F, \delta, q_0, z_0)$

حروف الفبا — حروف نوار

$q \xrightarrow{x \backslash y, D} q'$ $\quad D \in \{Right, Left\}$

$\Updownarrow$

$\delta(q, x) = (q', y, D)$

$z_0\ w_1\ w_2 \dots w_n$ نوار tape

$\Sigma \subseteq \Gamma \qquad B \in \Gamma \setminus \Sigma \qquad$ وضعیت‌های پذیرش $F \subseteq Q \qquad z_0 \notin \Gamma$

z_0 را نمی‌توان به نوار اضافه یا کم کرد $\longrightarrow$ $x = z_0 \overset{*}{\Leftrightarrow} y = z_0$

$\delta : Q \times (\Gamma \cup \{z_0\}) \longrightarrow Q \times (\Gamma \cup \{z_0\}) \times \{L, R\}$

در نوار فقط یک z_0 در کنار دیواره وجود دارد.

در نظریه محاسبه یک تابع ممکن است روی برخی از اعضای دامنه تعریف نشده باشد به چنین تابعی یک تابع جزئی می‌گویند (total func. ≠ partial func.=)

از ماشین تورینگ به دو طریق می‌توان استفاده کرد: • ورودی گرفته شود و ماشین بگوید پذیرفته است یا نه. • ورودی گرفته شود و یک خروجی عضو Γ^* داده شود (رشته موجود در نوار)

همچنین پذیرفته شدن را در ماشین تورینگ طبق دو استاندارد می‌توان معلوم کرد: • زمانی ورودی پذیرفته می‌شود که به یک وضعیت در F برسیم.

• زمانی ورودی پذیرفته می‌شود که تابع δ تعریف نشده باشد.

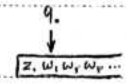
و زمان خروجی دادن را نیز طبق دو استاندارد می‌توان تعیین کرد: • وقتی به یک وضعیت در F می‌رسیم، رشته موجود در نوار را به عنوان خروجی بده.

• وقتی به جائی برسی که تابع δ تعریف نشده، رشته موجود در نوار را به عنوان خروجی بده.

B، بلانک است و در ابتدا درون نوار، z_0، سپس ورودی $w_1 w_2 \dots w_n$ وجود دارد و بقیه مقادیر نوار همگی B هستند. (توجه کنید که B جزو حروف الفای ورودی نیست.)

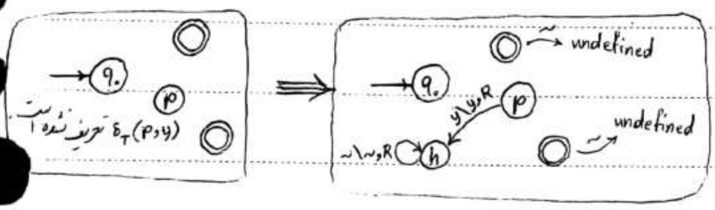
$z_0\ w_1\ w_2\ \overset{q_7}{\downarrow} w_3\ w_4\ w_5$: چنین وضعیتی در یک ماشین تورینگ را به صورت قراردادی با $w_1 w_2 q_7 w_3 w_4 w_5$ مشخص می‌کنیم.

زبان معادل ماشین تورینگ:



$L(T) = \{ \omega \in \Sigma^* \mid q_0\omega \vdash^* \alpha q \beta ,\ \alpha,\beta \in \Gamma^*,\ q \in F \}$: پذیرش رشته با رسیدن به وضعیت پذیرش

$H(T) = \{ \omega \in \Sigma^* \mid q_0\omega \vdash^* \alpha q x \beta ,\ \alpha,\beta \in \Gamma^*,\ x \in \Gamma ,\ q \in Q ,\ \delta(q,x) = \text{undefined} \}$: پذیرش رشته با توقف



قضیه: برای هر T، یک T' وجود دارد که: $L(T) = H(T')$

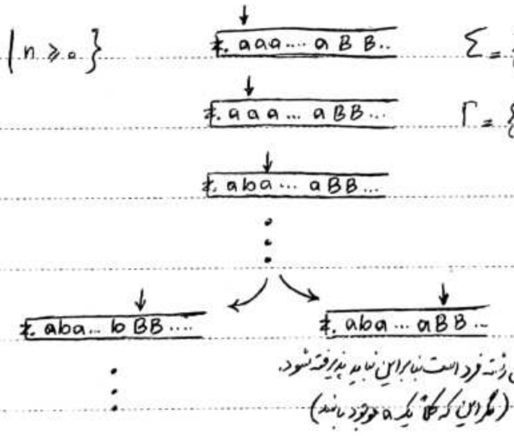
$Q_{T'} := Q_T \cup \{h\}\ (h \notin Q_T)$

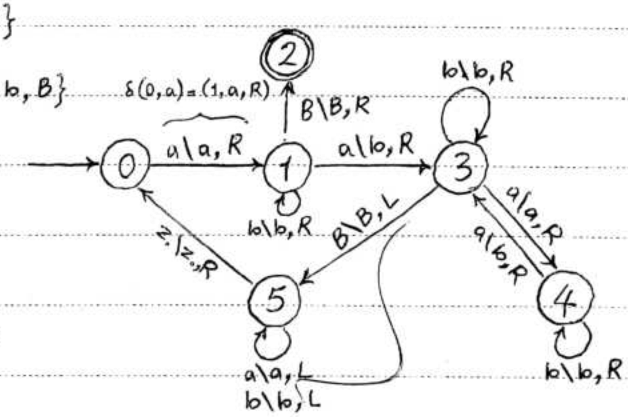
$$\delta'(q,x) = \begin{cases} \text{undefined} : q \in F,\ x \in \Gamma \\ \delta(q,x) : q \notin F,\ q \neq h,\ \delta(q,x)\ \text{defined} \\ (h,x,R) : q \notin F,\ q \neq h,\ \delta(q,x)\ \text{undefined} \\ (h,x,R) : q = h \end{cases}$$

قضیه: برای هر T، یک T' وجود دارد که: $H(T) = L(T')$

اثبات: کافیست یک state جدید h تعریف کنیم و هرجا δ تعریف نشده بود در δ' در آن شرایط به h برویم و قرار دهیم: $F := \{h\}$

مثال: $\{ a^{2^n} \mid n \geq 0 \}$ $\Sigma = \{a\}$ $\Gamma = \{a, b, B\}$





$q_0 a^8 \vdash a q_1 a^7 \vdash ab q_3 a^6 \vdash^* abababab q_3 \vdash^* q_0 abababab \vdash a q_1 bababab \vdash ab q_4 ababab$

$\vdash abb q_3 babab \vdash abbb q_3 abab \vdash^* abbbabbb q_3 \vdash^* q_0 abbbabbb \vdash^* ab^7 q_3 \vdash^* q_0 ab^7$

$\vdash a q_1 b^7 \vdash^* ab^7 q_1 \vdash ab^7 q_2$ ✓

$q_0a^4 \vdash^* q_0ababab \vdash aq_1babab \vdash abq_1abab \vdash abbq_3bab \vdash abbbq_3ab \vdash abbbaq_4b$

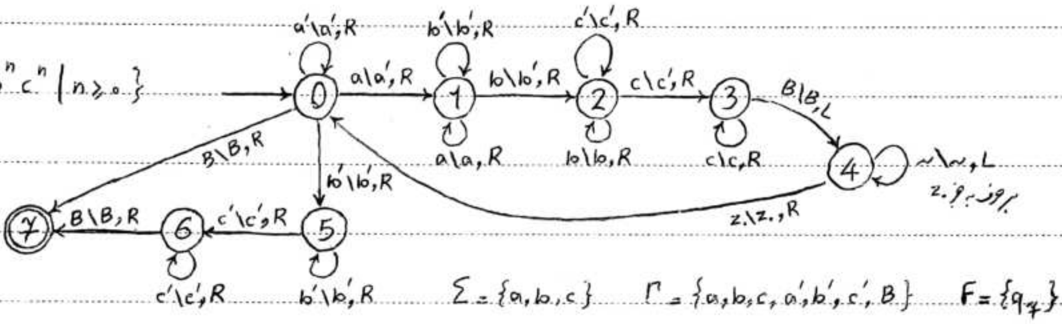
$\vdash abbbabq_4$ X

چون $\delta(q_4,B)$ تعریف نشده، این رشته پذیرفته نیست.

★ ماشین تورینگ درباره همه رشته‌ها لزوماً تصمیم‌گیری نمی‌کند. در واقع یعنی هر ماشین تورینگ در مورد هر رشته ورودی ممکن است به ۳ وضعیت دچار شود:

۱) به یک final state برسد ۲) توقف یابد (به وضعیت undefined برسد.) ۳) تا ابد ادامه یابد. (این لزوماً به معنی افتادن در یک دور نیست.)

مثال: $\{a^nb^nc^n \mid n \geq 0\}$



$\Sigma=\{a,b,c\}$ $\Gamma=\{a,b,c,a',b',c',B\}$ $F=\{q_7\}$

مثال: روش مورد نظر را در مثال زیر از تصمیم‌گیری ماشین‌ها، روشن می‌کنیم!!! $\{a^p \mid p \text{ is prime}\}$

$aaaaaaa \vdash^* aaaaaaaccbbbbbb \vdash^* aa'aa'aa'accbbbbbb \vdash^* aaaaaaacccbbbbb \vdash^* aaa'aaa'acccbbbbb$

$\vdash^* aaaaaaaccccbbbb \vdash^* aaaa'aaaccccbbb \vdash^* \ldots \vdash^* aaaaaaaccccccc$ ✓

روش دوم: ① ماشین ابتدا در انتهای رشته cc بنویسد ② سپس به اولین c برود و آن را به b تبدیل کند ③ سپس به ابتدای رشته برود و اولین a را به e تبدیل کند.

④ سپس به c بعدی برود و اگر آخرین c نیست، فرآیندهای ② و ③ را تکرار کند ⑤ اگر به آخرین c رسیده بود، به ابتدای رشته برود و اولین a را به f تبدیل کند.

⑥ حالا همه c ها را b کن و به مرحله ② برو. ⑦ اگر پس از تمام شدن a ها، آخرین a به f تبدیل شد، و f های دیگری هم وجود داشت به یک state جدید برود و از آن‌جا به تابع undefined شود.

و اگر فقط همان یک f (در جایگاه آخرین a) وجود داشت به یک state پذیرش برود. در حالتی که آخرین a به e تبدیل شده بود: - همه حروف را به حالت اولیه a و c برگردان

- یک c به انتها اضافه کن. - به مرحله ② برو.

Subject
Date

$\overbrace{\vdash w_1 w_2 w_3 \ldots w_k}^{w} \ldots$

ماشین تورینگ با خروجی رشته: $f: \Sigma_1^* \longrightarrow \Sigma_2^*$ $\quad (B \notin (\Sigma_1 \cup \Sigma_2))$

طبق تعریف اول برای زمان توقف: اگر $f(w)$ تعریف نشده است هیچ وقت به state پذیرش نرسیم و اگر $f(w) = u$، اولین بار که به یک state پذیرش می‌رسیم، داریم:

طبق تعریف دوم برای زمان توقف: اگر $f(w)$ تعریف نشده است هیچ وقت متوقف نشود و اگر $f(w) = u$، ماشین متوقف شود و در لحظ توقف، داریم:

$\vdash \underbrace{u_1 u_2 u_3 \ldots u_k}_{u = f(w)}\ B \ldots$ $\rightarrow$ اولین B

- ماشین تورینگ عادی (که هر رشته را پذیرش یا رد می‌کرد) را می‌توان به صورت $f: \Sigma^* \to \{0,1\}$ نشان داد که خروجی 0 به معنای رد شدن و 1 به معنای پذیرش است.

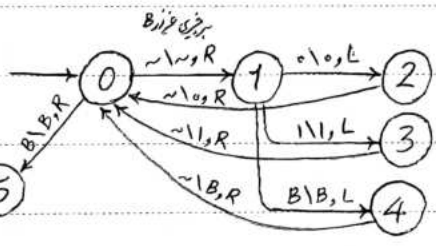
$f_1: \{0,1\}^* \longrightarrow \{0,1\}^*$ $\qquad$ $f_2: \{0,1\}^* \longrightarrow \{0,1\}^*$ $\qquad$ $f_3: \{0,1\}^* \longrightarrow \{0,1\}^*$

$w \longmapsto w^R$ $\qquad$ $w_1 w_2 \ldots w_n \longmapsto w_2 \ldots w_n \ (\varepsilon \mapsto \varepsilon)$ $\qquad$ $w_1 w_2 \ldots w_n \longmapsto 0 w_1 w_2 \ldots w_n$

$f_4: \{0\}^* \longrightarrow \{0,1\}^*$ $\qquad$ $f_5: \{0,1\}^* \longrightarrow \{0\}^*$

$0^n \longmapsto$ n در مبنای دو $\qquad$ n در مبنای دو $\longmapsto 0^n$

T_3:



تمرین تحویلی: (۲۰) T_5 توصیفی (۲۱) T_3 دقیق (۲۲) T_1 دقیق مهلت: ۳ دی

PAPCO

Subject

Date

بررسی وجود زبان غیرقابل محاسبه با ماشین تورینگ:

برابر بودن تعداد اعضای $\{a, b, c\}$ و {ج، ب، الف} به معنای وجود یک تناظر یک به یک بین اعضای دو مجموعه است. (شمردن یعنی ایجاد تناظر یک به یک بین $\{1, 2, \dots, k\}$ با مجموعه مورد نظر!)

سؤال: آیا برای هر دو مجموعه نامتناهی X و Y، تعداد اعضای X و Y برابرند؟ یعنی تابع $f: X \xrightarrow[\text{پوشا}]{\text{۱به۱}} Y$ وجود دارد؟

- نامتناهی بودن X، یعنی به ازای هیچ $k \in \mathbb{N}$، X در تناظر یک به یک با اعضای $\{1, 2, \dots, k\}$ نمی‌تواند باشد.

جواب: نه! نشان می‌دهیم که هیچ تابع پوشای $f: \mathbb{N} \to [0, 1)$ وجود ندارد. فرض خلف می‌کنیم. عدد $0 \le x < 1$ را به این صورت معرفی می‌کنیم:

رقم اول اعشار x، رقمی غیر از رقم اول اعشار $f(1)$ است، رقم دوم اعشار x، رقمی غیر از رقم دوم اعشار $f(2)$ است، ... بنابراین: $\nexists k \in \mathbb{N} : f(k) = x$

$|\mathbb{N}| = \aleph_0$ الف صفر $\qquad |\mathbb{R}| = 2^{\aleph_0} = \aleph_1$ الف یک

$|\mathbb{N}| = |\mathbb{Z}| = |\mathbb{Q}| \qquad f: \mathbb{N} \to \mathbb{Q}$

مخرج \ صورت	۱	۲	۳	۴
۱	۱/۱	۲/۱	۳/۱	۴/۱
۲	۱/۲	۲/۲	۳/۲	۴/۲
۳	۱/۳	۲/۳	۳/۳	۴/۳
۴	۱/۴	۲/۴	۳/۴	۴/۴

...

شمارا $\equiv \aleph_0$

$|\mathbb{R}| = |[0, 1)| = |\mathbb{Q}^c|$

اندازه مجموعه همه زبان‌ها با الفبای $\{0, 1\}$:

هر کدام از مقادیر $\{0, 1\}^*$ را متناظر با یک عدد طبیعی می‌گیریم، به صورت روبرو:

$$\begin{array}{cccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & \\ \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \\ \varepsilon & 0 & 1 & 00 & 01 & 10 & 11 & 000 & 001 & \dots \end{array}$$

بنابراین می‌توان عددی $0 \le x < 1$ (به صورت) را به هر زبانی با الفبای $\{0, 1\}$ متناظر کرد:

$L = \{\varepsilon, 0, 11\} \longleftrightarrow \{1, 2, 7\} \longleftrightarrow 0/1100001000\dots$

در نتیجه: $|\{0, 1\}^*| = |\mathbb{N}| = \aleph_0 \qquad |\underset{\{0,1\}}{\text{مجموعه زبان‌ها با الفبای}}| = |\{L \mid L \subset \{0, 1\}^*\}| = |\text{Powerset}(\{0, 1\}^*)| = |[0, 1)| = |\mathbb{R}| = \aleph_1$

Subject
Date

به طور کلی تعداد اعضای هیچ مجموعه‌ای با مجموعه زیرمجموعه‌های آن برابر نیست: $\forall X: P(X)=\{Y \mid Y\subseteq X\}$ $\quad \nexists f: f: X \xrightarrow[\text{Onto}]{1-1} P(X)$

برای اثبات فرض خلف می‌کنیم. حال مجموعه $A=\{x\in X \mid x\notin f(x)\}$ را در نظر بگیرید: $\exists a\in X: f(a)=A$ $\quad (a\in f(a)=A \overset{*}{\Longleftrightarrow} a\notin f(a)=A)$

● تعداد ماشین‌های تورینگ شمارا است: $\mathcal{T}=\{T \mid$ ماشین تورینگ $T\}$ $\quad |\mathcal{T}|=\aleph_0$

برای اثبات شمارا بودن ماشین‌های تورینگ، یک کدینگ یکتا برای همه ماشین‌های تورینگ در نظر بگیرید. در این کدینگ هر ماشین، یک رشته متناهی از الفبای کدینگ است. بنابراین مثل Σ^* شمارا است. (همین طور می‌توان نشان داد که تعداد همه برنامه‌هایی که با زبان جاوا می‌توان نوشت، شمارا است.)

همان طور که قبلاً گفتیم، مجموعه همه زبان‌ها ($=P(\Sigma^*)$) ناشمارا است؛ یعنی: $|P(\Sigma^*)| > |\mathcal{T}|$ بنابراین بعضی از زبان‌ها ماشین تورینگ ندارند.

$A=$ مجموعه اعداد جبری $:=$ ریشه‌های حقیقی همه چندجمله‌ای‌های صحیح $\qquad \mathbb{R}\setminus A=$ مجموعه اعداد متعالی $:=$ اعداد حقیقی که ریشه هیچ چندجمله‌ای صحیح نیستند.

$x^2-2=0 \Rightarrow \sqrt{2}\in A \qquad \sqrt[3]{\dfrac{\sqrt{5}-\sqrt{2}}{4\sqrt{4}}}\in A$

A شمارا است: برای اثبات کافیست همه چندجمله‌ای‌های صحیح را بشماریم. برای این کار به این صورت عمل می‌کنیم که ابتدا تعداد چندجمله‌ای‌هایی که مجموع قدرمطلق ضرایبشان یک است را می‌شماریم سپس تعداد چندجمله‌ای‌های صحیحی که مجموع قدرمطلق ضرایبشان ۲ است را می‌شماریم، سپس ...

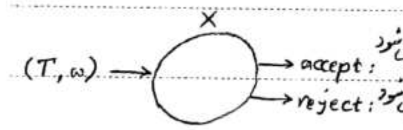
(توجه کنید که این راه که اول تعداد چندجمله‌ای‌های درجه یک را بشماریم، سپس درجه ۲ و ... نادرست است؛ چون چندجمله‌ای‌های درجه یک هیچ وقت تمام نمی‌شود.)

بنابراین چون $\mathbb{R}$ ناشمارا است، پس اعداد متعالی وجود دارند. (نه تنها وجود دارند، بلکه ناشمارا هستند!)

عدد لیوویل ($=0.1101\underbrace{00}_{2!}1\underbrace{000000}_{3!}1\underbrace{00\ldots00}_{4!}1000\ldots$) یک عدد متعالی است. همچنین اعداد π، e، $\ln 2$ و ... همگی متعالی هستند.

حال با توجه به این که ثابت کردیم زبان فاقد ماشین تورینگ وجود دارد، می‌خواهیم یک زبان فاقد ماشین تورینگ ارائه دهیم:

$H := \{ (T, \omega) \mid$ ماشین تورینگ T با ورودی ω متوقف می‌شود $\}$



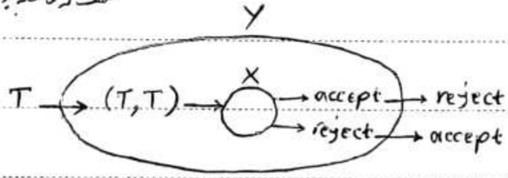
با برهان خلف فرض می‌کنیم ماشین تورینگ X وجود دارد که زبان H را تصمیم می‌گیرد.

و این تناظر قابل محاسبه است (با ساز و کار تورینگ)

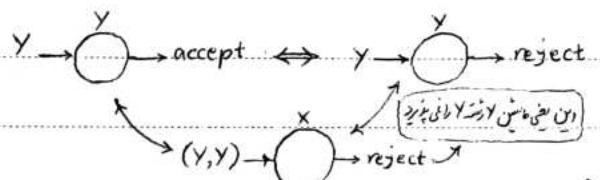
توجه کنید که چون بین مجموعه T و $\mathbb{N}$ تناظر یک به یک وجود دارد و نیز بین Σ^* و $\mathbb{N}$ تناظر یک به یک وجود دارد، بنابراین بین هر ماشین تورینگ T

این تناظر هم محاسبه‌پذیر است. البته تناظرهای یک به یکی هم بین مجموعه‌های شمارا و $\mathbb{N}$ هست که محاسبه‌پذیر نباشند.

و هر رشته ω یک تناظر یک به یک وجود دارد. درنتیجه (T,T) معنی‌دار است و منظور (T,ω) است که ω رشته متناظر با T است.



حال ماشین تورینگ Y را به صورت روبرو ارائه می‌دهیم: (طبق توضیحات بالا Y واقعاً تورینگ است.)



(چرا؟؟؟): حال داریم: [redacted]

به طور کلی این دست تناقض‌ها که شبیه به پارادوکس راسل هستند، تکنیک قطری‌سازی نام دارد.

قطری‌سازی را در این مسئله به این صورت نشان می‌دهیم:

$f: \mathbb{N} \xrightarrow[\text{onto}]{1-1} T$ محاسبه‌پذیر $\qquad g: \mathbb{N} \xrightarrow[\text{onto}]{1-1} \Sigma^*$ محاسبه‌پذیر

$\Sigma^* \backslash T$	f(1)	f(2)	f(3)	f(4)	...
g(1)	×	✓	✓	×	
g(2)	×	×	✓	✓	
g(3)	×	×	✓	×	
g(4)	✓	✓	×	×	
⋮					⋱

در جدول روبرو در درایه (i,j)، $T = f(i)$ و $\omega = g(j)$، ✓ قرار می‌دهیم اگر و تنها اگر ماشین $f(i)$ با ورودی $g(j)$ متوقف شود.

توجه کنید که چون فرض کرده‌ایم X وجود دارد، هر خانه این جدول را با X می‌توان پر کرد.

حال Y به صورت عکس عملکرد قطر این ماتریس کار می‌کند، یعنی:

Y:	1	2	3	4	...
	✓	✓	×	✓	

حال عددی که با Y متناظر است را در نظر می‌گیریم؛ مثلاً $Y = f(n)$، $n \in \mathbb{N}$. در این صورت عملکرد ماشین Y [redacted] برای ورودی متناظر با n [redacted]، متناقض است.

(یعنی اگر ✓ باشد، می‌توان نتیجه گرفت که باید × باشد و اگر × باشد، می‌توان نتیجه گرفت که باید ✓ باشد.)

نتیجه این است که این فرض که هر خانه از جدول را به وسیله X می‌توان پر کرد، غلط بوده و X در مورد برخی خانه‌ها سکوت می‌کند!

Subject
Date

و رشته‌ای خارج از زبان را تصمیم نمی‌گیرد.

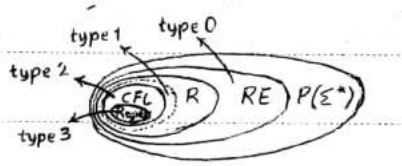
(Acceptor)
RE: مجموعه زبان‌هایی که ماشین تورینگ دارند = $\{L \subseteq \Sigma^* \mid \exists T: L = L(T)\}$ (منظور ماشین تورینگی است که اعضای زبان را تشخیص می‌دهد)
Recursively Enumerable

در تعریف اول (که پذیرش با رسیدن به final state اتفاق می‌افتاد)، می‌توان زیرمجموعه $Q_{Reject} \subset Q$ را در نظر گرفت که $\overbrace{Q_{Accept}}^{F} \cap Q_{Reject} = \emptyset$ که با رسیدن به این stateها، رشته ورودی رد می‌شود. در این صورت ماشین به ازای هر ورودی یا می‌پذیرد یا رد می‌کند یا توقف نمی‌کند. (δ در هیچ جا تعریف نشده نیست.)

(Decider)
R: مجموعه زبان‌های دارای الگوریتم ← الگوریتم: یک ماشین تورینگ که به ازای هر ورودی متوقف می‌شود. (می‌پذیرد یا رد می‌کند.) $R \subseteq RE$
Recursive

- دقت کنید که قدرت ماشین تورینگی که Q_{Reject} دارد، با ماشین تورینگ عادی یکسان است. (کافیست در ماشین تورینگ عادی، تعریف نشده‌ها را به یک $q_1 \in Q_{Reject}$ ببریم.)

$CFL \subsetneq R$, $R \subsetneq RE$



شمارا ناشمارا
$RE \subsetneq P(\Sigma^*)$

$H := \{(T, w) \mid$ ماشین تورینگ T رشته w را می‌پذیرد $\}$ ⟶ Halting Problem

در صفحه قبل نشان دادیم
$H \notin R$ $H \in RE$ (چرا؟؟)

مسأله تناظر Post: متناهی کارت مانند $A_i = \boxed{\begin{matrix} w_i \\ u_i \end{matrix}}$ ، $1 \le i \le k$ داریم. آیا n وجود دارد که: $w'_1 w'_2 \dots w'_n = u'_1 u'_2 \dots u'_n$

به طوری که: $\forall 1 \le j \le n: \exists 1 \le i \le k: \begin{cases} w'_j = w_i \\ u'_j = u_i \end{cases}$

کل این مجموعه را به صورت یک رشته در زبان کد می‌کنیم.

زبان PCP = $\{\overbrace{[\square, \square, \dots, \square]} \mid$ برای این مجموعه کارت تناظر Post وجود دارد $\}$

برای زبان PCP، یک ماشین تورینگ وجود دارد که رشته‌ای عضو زبان را تشخیص دهد. به این صورت که ابتدا می‌بیند جواب برای $n=1$ یافت می‌شود یا خیر، سپس برای $n=2$، سپس $n=3$ و ... اگر رشته عضو زبان باشد، این ماشین بالاخره متوقف می‌شود. (اما اگر عضو زبان نباشد، هیچ وقت متوقف نمی‌شود.) بنابراین: $PCP \in RE$

در ادامه نشان می‌دهیم که هیچ ماشین تورینگی وجود ندارد که به ازای هر ورودی هم در مورد عضو PCP بودن، نظر قطعی دهد و هم در مورد عضو PCP نبودن؛ یعنی:
$PCP \notin R$

Subject
Date

برای این که نشان دهیم زبان PCP، تصمیم ناپذیر است (دارای الگوریتم نیست / $\notin R$) نشان می دهیم که اگر تصمیم پذیر باشد، زبان H هم تصمیم پذیر خواهد بود.

(Halting Problem) برای این کار به ازای هر (T, w) که T ماشین تورینگ و w رشته است، یک مجموعه متناهی از کارت ها ارائه می کنیم به طوری که:

L_1 := زیرمجموعه ای از ها که در آن اولین کارت نوع ۱ است. ماشین تورینگ T ورودی w را بپذیرد. $\Longleftrightarrow C_{T,w} \in L_1 = (\boxed{\,}, \boxed{\,}, \dots, \boxed{\,})$

این مجموعه متناهی کارت ها را به صورت زیر ارائه می دهیم:

① $\left[\dfrac{\#}{\# q_0 w_1 w_2 \dots w_n \#}\right]$ ② $\forall \delta(q,a) = (r, b, R): \left[\dfrac{qa}{br}\right]$ ③: $\forall \delta(q,a) = (r, b, L), \forall c \in \Gamma: \left[\dfrac{cqa}{rcb}\right]$

④ $\forall a \in \Gamma: \left[\dfrac{a}{a}\right]$ ⑤: $\left[\dfrac{\#}{\#}\right]$ ⑥ $\forall a \in \Gamma: \left[\dfrac{aq_{accept}}{q_{accept}}\right], \left[\dfrac{q_{accept}a}{q_{accept}}\right], \left[\dfrac{q_{accept}\#\#}{\#}\right]$

چند مسئله تصمیم ناپذیر معروف:

① مسئله تناظر Post

② برای دو ماتریس داده شده A و B، $A, B \in M_{15}(\mathbb{Z})$ (درایه های $A_{15\times15}$ و $B_{15\times15}$ صحیح است) آیا دنباله متناهی C_1، C_2، ... و C_k وجود دارد که

$\forall i, C_i \in \{A, B\}$ و $C_1 \times C_2 \times \dots \times C_k = O_{15\times15}$

③ بیان مسئله ② با سه ماتریس صحیح ۶×۶

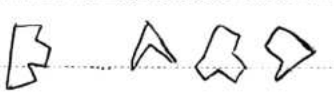
④ مولد گروه ؟ ...

PaPCO

Subject
Date

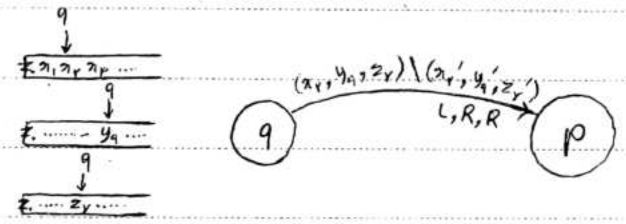
⑤ آیا با شروع از هر n طبیعی و استفاده مکرر از تابع f روبرو، می‌توان به ۱ رسید؟

$$f: \mathbb{N} \to \mathbb{N} \qquad n \mapsto \begin{cases} \frac{n}{2} & \text{زوج } n \\ 3n+1 & \text{فرد } n \end{cases}$$

⑥ کاشی‌کاری: آیا با کاشی‌های داده شده می‌توان صفحه‌ی نامتناهی را پوشاند؟ 

⑦ یک معادله سیاله چندجمله‌ای ریشه صحیح دارد؟

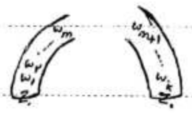
تلاش مذبوحانه برای قوی کردن ماشین تورینگ:

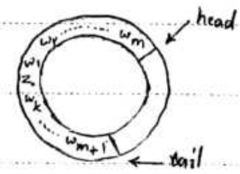


• چندنواره کردن ماشین تورینگ $\delta: Q \times \Gamma^3 \to Q \times \Gamma^3 \times \{L, R\}^3$

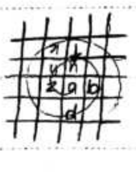
تبدیل به ماشین تورینگ عادی: Ƶ $x_1\ y_1\ z_1\ x_2\ y_2\ z_2$ .....

مثلا یک عملیات بالا در ماشین تورینگ عادی به این صورت است که ابتدا x_2 را به x_2' تبدیل می‌کند سپس x_1 را به x_1^{*p} تبدیل می‌کند (تا بداند در مرحله بعدی باید سراغ x_1 برود و به state p برود) سپس y_9 را به y_9'، y_{10} را به y_{10}^*، z_2 را به z_2' و z_8 را به z_8^* تبدیل کند. سپس به سراغ حروف ستاره دار برود و با توجه به δ، عملیات‌های بعدی را انجام دهد.

• استفاده از دو پشته: Ƶ $w_1 w_2 \ldots w_m \ldots w_k$ BB.... $\leftrightarrow$ 

• استفاده از صف: Ƶ $w_1 w_2 \ldots w_m \ldots w_k$ BB... $\leftrightarrow$ 

• Non-deterministic کردن ماشین تورینگ: استفاده از چند نوار برای ذخیره همه مسیرهای ممکن

• دوبعدی کردن نوار: نمایش نوار دوبعدی به صورت مارپیچ  $\leftrightarrow$ Ƶ a z y n B b B d ...

• ماشین ۴ حافظه‌ای طبیعی با ۳ دستور (توضیح در انتهای جزوه)؛ حتی ۲ حافظه طبیعی یا ۱ حافظه گویا هم معادل تورینگ است.

Subject
Date

پیچیدگی زمانی: $P \subseteq NP$ $P \overset{?}{\underset{open}{=}} NP$

P = مجموعه زبان‌هایی مثل L که برای آن یک چندجمله‌ای مثل Q(.) و یک ماشین تورینگ قطعی T وجود دارد که T به ازای هر ورودی w در زمان Q(|w|) پاسخ می‌دهد. (منظور از زمان، تعداد گام‌های تابع انتقال δ است.)
polynomial

- اگر ماشینی در زمان Q(|w|)، رشته‌های پذیرفته شده را بتواند تشخیص دهد، رشته‌های رد شده را هم می‌تواند تشخیص دهد؛ چون هر رشته‌ای که در این زمان پذیرفته نشود، قطعاً رد شده است!

مثال: $\{0^p \mid p \text{ عدد اول}\} \in P$ ← این مسئله سال‌ها open بود و در سال ۲۰۰۴ دو دانشجوی کارشناسی هندی آن را اثبات کردند!

NP = مجموعه زبان‌هایی مثل L که برای آن یک چندجمله‌ای Q(.) و یک ماشین تورینگ غیرقطعی T وجود دارد که T هر ورودی w را حداکثر در زمان Q(|w|) پاسخ می‌دهد.
non-deterministic polynomial

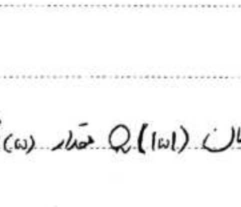
≡ مجموعه زبان‌هایی مثل L که برای آن‌ها ماشین تورینگ غیرقطعی T' و چندجمله‌ای Q'(.) وجود دارد که برای هر $w \in L$، $u \in \Sigma^*$ وجود دارد که زبان $\{(w,u) \mid w \in L\}$ توسط T' در زمان Q'(|w|) پاسخ داده می‌شود.

- در ماشین غیرقطعی T که زبان NP را در زمان چندجمله‌ای می‌پذیرد، در واقع u همان مسیری را مشخص می‌کند که ماشین را رسیدن به state پذیرش طی می‌کند. بنابراین اگر u را داشته باشیم می‌توان ماشین قطعی T' را ارائه دهیم که در زمان چندجمله‌ای زبان را تصمیم بگیرد.

تابع $f: \Sigma^* \to \Sigma^*$ محاسبه‌پذیر در زمان چندجمله‌ای است؛ یعنی ماشین تورینگ T وجود دارد که برای هر ورودی w در زمان Q(|w|) مقدار f(w) را حساب کند.

$L \le L'$ ★

اگر تابع f محاسبه‌پذیر در زمان چندجمله‌ای باشد، آنگاه داریم: $(w \in L \iff f(w) \in L')$ ⟸ (L' چندجمله‌ای ⟸ L چندجمله‌ای)



NP-hard: زبان L' را NP-hard می‌گوییم هرگاه: $\forall L \in NP,\ L \le L'$ (سخت‌تر (زبان تبدیل))

$NP \cap NP\text{-hard} = NP\text{-complete}$

SAT: تشخیص این که گزاره منطقی حالت "درست بودن" برای یک وجود دارد یا نه. مسئله SAT، NP-complete است.

Subject
Date

چند مسئله:

(ابهامی ←)

• زبان گرامر زیر را بیابید: $P(G)=\{S\to AB|\varepsilon,\ A\to aASb|a,\ B\to bS\}$

لم ①: در هر $\alpha\in(V+T)^*$ که $S\overset{*}{\vdash}\alpha$، تعداد A و a های هر پیشوند α بزرگتر یا مساوی تعداد B و b هاست و: $N_{a+A}(\alpha)=N_{b+B}(\alpha)$

اثبات با استقرا روی k که $S\overset{(k)}{\vdash}\alpha$ و دقت به سه قاعده گرامر: $\underbrace{S\overset{(k-1)}{\vdash}\beta y\gamma\vdash\alpha,\ \exists y\in\{S,A,B\},\ \exists\beta,\gamma\in(V+T)^* \ldots}_{\text{گزاره ناوردایی}}$

بنابراین زبان این گرامر، زیرمجموعه زبان روبروست: $L=\{\omega\,|\,N_a(\alpha)\geq N_b(\alpha):$ برای هر پیشوند α از ω $,\ N_a(\omega)=N_b(\omega)\}$

• زبان دهید گرامری با قواعد روبرو مبهم است و گرامر معادلی ارائه دهید که مبهم نباشد: $S\to ABA\quad A\to aA|\varepsilon\quad B\to bB|\varepsilon$

ـ گرامر مبهم گرامری است که در آن رشته‌ای وجود دارد که از دو طریق به دست می‌آید (که فرم اشتقاق از چپ این دو طریق یکسان نیستند)

• هر گرامر مستقل از متن دارای گرامر معادلی به فرم نرمال گریباخ است: $S\to\varepsilon,\ A\to \overset{\in T}{a}\,\overbrace{A_1\ldots A_n}^{\in V^*}$: فرم نرمال گریباخ

ثابت کنید اگر $\varepsilon\notin L(G)$، برای G گرامر (context free) معادلی به فرم روبرو وجود دارد: $A\to a,\ A\to aB,\ A\to aBC$

و ثابت کنید گرامر معادلی وجود دارد که در سمت راست، هیچ دو variable متوالی دیده نمی‌شود.

• زبان گرامر روبرو را توصیف کنید: $S\to aS|aSbS|\varepsilon$

جواب: $\{\omega\,|$ $\overbrace{\text{در هر پیشوند } \omega \text{، تعداد a ها بزرگتر یا برابر با تعداد b هاست}}^{\text{شبیه } Ab}$ $\}=X$، اثبات زیرمجموعه X بودن با گزاره ناوردایی: در هر عبارتی وسط مسیر شرط A برقرار است. ـ $\underbrace{\text{باید نشان دهیم همه اعضای X تولید می‌شود}}_{\text{قسمت سخت مسئله}}$

• گرامری برای زبان روبرو بیابید. $\Sigma=\{0,1,\#\}\quad L=\{\omega\#\omega^R\#\,|\,\omega\in\{0,1\}^+\}^*$

Subject
Date

• یک ماشین با ۴ حافظه در نظر بگیرید که در هر خانه یک عدد طبیعی یا صفر ذخیره می‌شود. ۳ نوع دستور داریم:

I_i $(1 \le i \le 4)$: یک واحد به خانه i بیفزا و به دستور بعد برو

D_i $(1 \le i \le 4)$: یک واحد از خانه i کم کن (اگر مثبت است) و از دستور بعدی اجتناب کن و به بعدی برو و اگر صفر بود دستور بعدی را اجرا کن.

T_j $(j \in \mathbb{Z})$: برو به jاُمین دستور بعد. (اگر j<0 بود، به |j|اُمین دستور قبل برو)

اجرای برنامه وقتی متوقف می‌شود که دستوری که نوبت آن است، وجود ندارد. وضعیت ۴ خانه در چنین لحظه‌ای خروجی برنامه است.

- مجموعه‌ای از دستورات بنویسید که به ازای ورودی $n\ 0\ 0\ 0$، خروجی $2n\ 0\ 0\ 0$ را بدهد.

$n000 \quad 02n00 \quad 2n000$

$D_1 T_4 I_2 I_2 T_{-4} D_2 T_3 I_1 T_{-3}$

- ورودی: $n\ 0\ 0\ 0$ خروجی: $a_n\ 0\ 0\ 0$ $\begin{cases} a_0 = a_1 = 1 \\ a_n = a_{n-1} + a_{n-2} \end{cases}$

$n\ a_0\ a_1\ 0$

$n-1\ a_2\ 0\ a_1$

$n-1\ 0\ a_2\ a_1$

$n-1\ a_1\ a_2\ 0$

⋮

- ورودی: $n\ 0\ 0\ 0$ خروجی: $a_n\ 0\ 0\ 0$ $\begin{cases} a_0 = a_1 = a_2 = 1 \\ a_n = a_{n-1} + a_{n-2} + a_{n-3} \end{cases}$

ساختن 2^n

① ساختن 2^n: $n\ 1\ 0\ 0$ $\xrightarrow{\times 2}$ $n-1\ 0\ 2\ 0$; $n-1\ 2\ 0\ 0$ $\xrightarrow{\times 2}$ ⋮

② فیبوناچی سه تایی: $0\ a_i\ a_{i+1}\ a_{i+2}$; $a_{i+2}\ \begin{smallmatrix} a_i + \\ a_{i+2} \end{smallmatrix}\ a_{i+1}\ 0$; $a_{i+2}\ a_{i+3}\ 0\ a_{i+1}$

مرتب کردن

$0\ a_{i+1}\ a_{i+2}\ a_{i+3}$

⋮

راه حل: $0\ 2^n a_1\ 2^n a_2\ 2^n a_3$

فیبوناچی سه تایی

تقسیم بر دو تا رسیدن به فرد

فیبوناچی سه تایی

تقسیم بر دو تا رسیدن به فرد

⋮

③ تقسیم بر دو کردن تا رسیدن به عدد فرد:

D_1 $2^n \times a\ 0\ 0\ 0$ $\xrightarrow{\div 2}$

D_1

T_3 $0\ 2^{n-1}\times a\ 0\ 0$

I_2

T_{-4} $2^{n-1}\times a\ 0\ 0\ 0$

PAPCO

Subject
Date

طبقه‌بندی چامسکی

نوع	ماشین‌ها	گرامرها
۰	تورینگ	غیرمحدود
۱	محدود خطی	وابسته به متن
۲	پشته‌ای ⟷	مستقل از متن
۳	متناهی ای‌ات ⟷	منظم

← گرامرها و ماشین‌هایی دربین سطح ۲ و ۳ وجود دارند.
مثل ماشین Non-deterministic PDA ....

← گرامرها و ماشین‌هایی ضعیف‌تر از سطح ۳ وجود دارد.
مثل گرامری که از +، . و c (جتم) ساخته شده ....