جزوه نظریه زبانها

استاد: سرکار خانم سمیه فرخ زاد

تهیه و تنظیم : سرکار خانم پیر بازاری

ab-rafiee.com publish

http://www.ab-rafiee.com

Subject
YEAR: MONTH: DATE:

مجموعه‌ای متناهی : Σ رشته : ω زبان : L

هر زبان به الفبا نیاز دارد. $L:\{\omega\in\Sigma^*, a^n b^n \mid n\geq 0\}$

الحاق :

مثال : $x='010' \;,\; z='111' \Longrightarrow x \overset{\text{الحاق}}{\circ} z='010111'$

Concat جابه‌جایی‌پذیر نیست.

$\{a,c\}\cdot\{d,e\}=\{a.d, a.e, c.d, c.e\}$

$L^2 \longrightarrow L.L=\{x.y \mid x\in L, y\in L\}$

$L^3=L^2.L$

$L^n=L^{n-1}.L$

ستاره $\bigcup_{n\geq 0}^{\infty} L^n=L^*$

$\bigcup_{n>0}^{\infty} L^n=L^+$ ستاره مثبت

نکته : رشته با طول صفر = λ

$\lambda=L^0=\omega^0$

هر چیزی به توان صفر شود λ : $\lambda.x=x \Longleftarrow$ λ عضو خنثی در الحاق است.

در L^+ ، رشته با طول صفر نداریم.

$$L_1 . L_2 = \{x.y \mid x \in L_1 , y \in L_2\}$$

ترتیب در الحاق مهم است.

$$L_1 \cup L_2 = \{w \in L_1 \text{ or } w \in L_2\}$$

$$L_1 \cap L_2 = \{w \in L_1 \text{ and } w \in L_2\}$$

$$L_1 - L_2 = \{w \in L_1 \text{ and } w \notin L_2\}$$

$$L_1 \vdash\dashv L_2 = (L_1 - L_2) \cup (L_2 - L_1)$$

تفاضل متقارن

$$(L_1 \cup L_2) - (L_1 \cap L_2)$$

تمرین: کدام یک از گزینه‌های زیر نادرست است؟

1) $\Sigma^* = \Sigma^+ \cup \{\lambda\}$ ✓

4) $L^+ = L^* - \{\lambda\}$ ✗

اگر خود زبان λ ندارد، درست است.

2) $\Sigma^+ = \Sigma^* - \{\lambda\}$ ✓

L^*: مجموعه‌ای از تمام الحاق‌های رشته‌های زبان L با طول از صفر تا بی‌نهایت

L^+: با طول از صفر به بالا

3) $L^* = L^+ \cup \{\lambda\}$

$L = \{001, 110, \lambda\}$ $\quad\quad$ $L^{+} = \bigcup_{n>0}^{\infty} L^{n}$

λ جزء الفبا نمی باشد.

ریورس یا معکوس : (R) :

مثال :

$w = x = '001'$

$\Downarrow$

$w^{R} = x^{R} = '100'$

$(x.y)^{R} = y^{R}.x^{R}$

$L^{R} = \{w \mid w^{R} \in L\}$

مثال :

$L = \left\{ \begin{matrix} '110', & '001', & '011' \\ 011 & 100 & 110 \end{matrix} \right\}$ *

$L^{R} = ?$ $\quad\quad$ $L^{R} = \{011, 110\}$

نکته : رشته هایی که معکوس آن در زبان باشد یعنی در مجموعه زبان باشد ، مدنظر است.

مکمل :

$\overline{L} = \{w \in \Sigma^{*} \text{ and } \notin L\}$

یعنی : $\Sigma^{*} - L$

Subject
YEAR: MONTH: DATE:

تمرین: مکمل زبان زیر را به دست آورید.

$L = \{a^n b^n \mid n \geq 0\}$ = رشته های به ترتیب ab دار به تعداد برابر است

۱) ترتیب ۲) رشته هایی که با a شروع و به b ختم میشوند

۳) تعداد ۴) $n \geq 0$

$\overline{L} = \{a^n b^m \mid n \neq m\} \cup \{w \mid$ حداقل رشته هایی که شامل ba باشند $w\}$

ترتیب را به هم بزنیم

aaaabbbb $\in L$

aabaabbbb $\in \overline{L}$

تمرین: حاصل عبارات زیر را بنویسید.

$\emptyset . \lambda = \emptyset$

$L . \{\lambda\} = L$

$\emptyset \cup \{\lambda\} = \{\lambda\}$

$L . \emptyset = \emptyset$?

$\emptyset^* = \{\lambda\}$

$\emptyset^+ = \emptyset$

Subject
YEAR: MONTH: DATE:

نکته : $\lambda \neq \emptyset$ ← مجموعه‌ی تهی

رشته با طول صفر

گرامرها :

گرامر مجموعه قوانینی برای تولید رشته های یک زبان با استفاده از الفبا است

$G = \{V, T, S, P\}$

V : متغیر، گره‌های میانی : با حروف بزرگ الفبای نمایش داده می‌شوند و به تولید رشته کمک می‌کنند و جزو الفبا نیستند.

T : ترمینال‌ها همان الفبا هستند با حروف کوچک نمایش داده می‌شوند و غیرقابل تجزیه‌اند.

S : نقطه‌ی شروع که یکی از متغیرها می‌باشد. $S \in V$

P : قوانین تولید

تمرین : زبان حاصل چیست؟

$G_1 = \{V_1, T_1, S_1, P_1\}$

$V_1 = \{S, A, B\}$ $T = \{a, b\}$

$S_1 = S$ $P = \{S \rightarrow AB, A \rightarrow aAb/\lambda, B \rightarrow Bb/b\}$

VAHDAT

پنج قانون داریم.

1) $S \xrightarrow{S \to AB} AB \xrightarrow{A \to aAb} aAbB \xrightarrow{A \to \lambda}$

$abB \xrightarrow{B \to b} abb$

2) $S \Longrightarrow AB \xrightarrow{B \to Bb} ABb \xrightarrow{B \to Bb} ABBb \xrightarrow{B \to b}$

$Abbb \xrightarrow{A \to aAb} aAbbbb \xrightarrow{A \to \lambda} abbbb$

$L = \{ w \in T^*, \; S \Longrightarrow w \mid a^n b^m \mid n < m \}$

نکته: مراحل تولید رشته با استفاده از قوانین تولید را اشتقاق گویند.

اشتقاق: تولید رشته های یک زبان به وسیله ی قوانین.

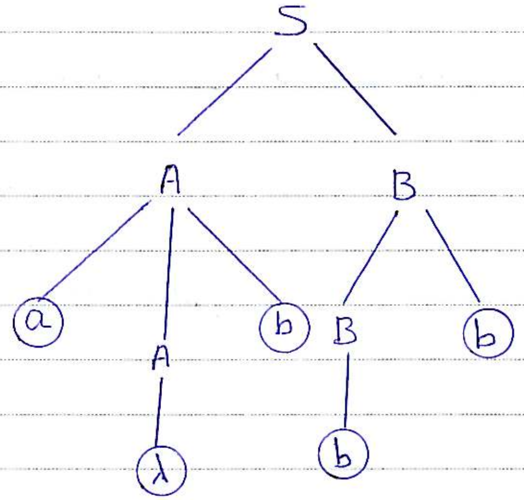
انواع اشتقاق:

۱- راست: در هر مرحله، راست ترین متغیر را جایگزین می کند

۲- چپ: در هر مرحله، چپ ترین متغیر را جایگزین می کند.

درخت اشتقاق:

گره ها: متغیر　　برگ: الفبا　　ریشه: S



$abbb = ab^3$

انواع گرامرها:

1- گرامر نوع سوم یا منتظم   regular

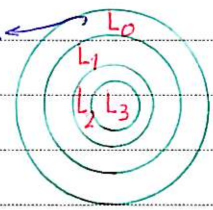
2- گرامر مستقل از متن   Context Free

3- نوع اول حساس به متن   Context Sesitive

4- نوع صفر   Unrestricted

قوی ترین

خانواده زبان های نوع صفر



Subject
YEAR: MONTH: DATE:

تفاوت گرامرها در نوشتن P است.

نکته: اگر الگوریتمی راه حل داشت پس یک گرامر نوع صفر وجود دارد.

و اگر گرامر نوع صفر وجود نداشته باشد آن الگوریتم قابل حل نیست.

گرامر منظم زبان منظم تولید می کند.

گرامر منظم:

$L = \{a^n \mid n \geqslant 3\}$ n مضارب 3 می باشد.

ریشه های کوتاه $S \rightarrow aaa \quad S/\lambda$

$S \rightarrow aaa / aaaA$

$A \rightarrow aA / a$

گرامر منظم، هم از راست و هم از چپ منظم است.

خطی از راست $\begin{cases} A \rightarrow a \\ A \rightarrow aA \end{cases}$ $a \in T, A \in V$ $\begin{cases} A \rightarrow \alpha \\ A \rightarrow \alpha A \end{cases}$ $\alpha \in T^*$

خطی از چپ $\begin{cases} A \rightarrow a \\ A \rightarrow Aa \end{cases}$ $a \in T, A \in V$ $\begin{cases} A \rightarrow \alpha \\ A \rightarrow A\alpha \end{cases}$ $\alpha \in T^*$

در خطی از راست و چپ، λ هم مورد قبول است.

VAHDAT

گرامر مستقل از متن:

$L=\{a^nb^n \mid n\geq 0\}$

از وسط a و b اضافه می‌شود.

گرامر $S \to aSb \,/\, \lambda$

مستقل تأکید می‌کند سمت چپ حتماً متغیر باشد و حتماً یک متغیر.

برترکیبی ⟵ $A \to \alpha$ تنها و تنها یک متغیر

$A \in V$

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

بیشترین کاربرد گرامر، مستقل از متن می‌باشد.

تمرین:

$L=\{w \mid w\in T^*,\ n_a(w)=n_b(w)\}$

تعداد a در رشته = تعداد b در رشته

$S \to aSb / bSa / \lambda$

در اینجا ترتیب ندارد

گرامر حساس به متن:

$L=\{a^nb^nc^n \mid n\geq 1\}$

$S \to aSbSc$

وقتی تعداد الفبا بیش از دو و نیز منظم باشد.

حساس به متن تأکید می‌کند کاهش طول نداشته باشیم.

حساس به متن λ را نمی‌پذیرد.

برترکیبی برترکیبی

$\alpha \longrightarrow \beta$

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

$\beta \in (V \cup T)^+$

راه حل های تولید λ در گرامرهای حساس به متن :

1) اگر گرامر داده شده، تمام رشته های زبان مدنظر را می پذیرفت و تنها λ باقی مانده بود، به طور ضمنی می پذیریم که گرامر و زبان مطابقت دارند.

2) یک S' در نظر می گیریم. (یک شروع مجازی با S')

$$S' \to \lambda / S$$

و در سایر قوانین هیچ گاه به S' بر نمی گردیم

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

1) $S \to abc$

2) $S \to aXbc$ X و Y متغیر نو.

3) $Xb \to bX$ X به جلو می رود

4) $Xc \to Ybcc$ برای اضافه کردن a

5) $bY \to Yb$ Y به عقب می رود

6) $aY \to aaX$

7) $aY \to aa$

Subject
YEAR: MONTH: DATE:

تمرین: با توجه به قوانین فوق، مراحل اشتقاق $a^3b^3c^3$ را بنویسید.

$S \Longrightarrow axbc \Longrightarrow abxc \Longrightarrow abybcc \Longrightarrow$

$aybbcc \Longrightarrow aaxbbcc \Longrightarrow aabxbcc \Longrightarrow$

$aabbxcc \Longrightarrow aabbybccc \Longrightarrow aabybbccc \Longrightarrow$

$aaybbbccc \Longrightarrow a^3b^3c^3$

تمرین: گرامر زبان زیر را بنویسید.

$$L = \left\{ w \mid w \in \{a, b, c\}^*, n_a(w) = n_b(w) = n_c(w) \right\}$$

Subject
YEAR: MONTH: DATE:

گرامر بی‌تحدید و شرط:

$\alpha \rightarrow \beta$

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

$\alpha \in (V \cup T)^+$

مجموعه‌های منتظم:

ویژگی خاص مجموعه‌ها: ۱- ترتیب مهم نیست. 2- عنصر تکراری ندارد.

قوانین مجموعه‌های منتظم:

۱) هر عنصر الفبا یک مجموعه‌ی منتظم است به‌تنهایی.

2) اگر X و Y هر دو مجموعه‌ی منتظم باشند، آنگاه $X \cup Y$ و $X.Y$ و X^* و Y^+ هم منتظمند.

3) هر مجموعه‌ای که بتوان با ترکیب . و $\cup$ و + و * ایجاد کرد، منتظم می‌شود.

گرامر: برای تولید رشته‌های زبان.

عبارات: برای نمایش رشته‌های زبان

ماشین: برای تشخیص رشته‌های زبان

VAHDAT

Subject
YEAR: MONTH: DATE:

خودبازگشتی میانی $A \rightarrow aAb$

خودبازگشتی از راست $A \rightarrow aA$

خودبازگشتی از چپ $A \rightarrow Aa$

* نکته: اگر گرامری دارای فرمت گرامرهای منظم باشد، ولی متناهی نباشد، منظم نیست.

زبانی که برای آن حداقل یک گرامر منظم داشته باشیم ← زبان منظم

چهار شرط که باعث گرامر مبهم می‌شود:

۱) گرامر داده شده روی یکی از متغیرهای خود، هم خودبازگشتی راست و هم چپ داشته باشد.

$A \rightarrow \alpha A$ و $A \rightarrow A\alpha$

2) اگر طی اشتقاق از یک متغیر به خود آن متغیر برسیم.

$A \rightarrow \ldots A$

3) شامل یک خودبازگشتی میانی و یک خودبازگشتی چپ یا راست.

$A \rightarrow aAb$, $A \rightarrow aA / Aa$

۴) اگر برای یک رشته‌ی w بتوان حداقل دو درخت اشتقاق رسم کرد.

اگر گرامر مبهم باشد، لزوماً زبان مبهم نیست. اگر هیچ گرامر نامبهمی نتوان برای زبان نوشت مبهم است.

انواع گرامر

1. نوع 3: منظم: همه خطی از چپ و راست ، یا همه جا انتها یا همه جا ابتدا

2. نوع 2: مستقل از متن: فقط سمت چپ یک متغیر

3. نوع 1: حساس به متن: کاهش طول نداشته باشیم.

4. نوع 0: بی قید و شرط:

گرامرهای مستقل از متن:

$A \rightarrow \alpha$

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

مثال: گرامر زبان مقابل را بنویسید. $L = \{ a^n b^n c^n \mid n \geq 1 \}$

1) کوچک ترین رشته ی زبان را تولید می کنیم. abc

2) به رشته های بعدی به طور هم زمان یکی به a ، یکی به b و یکی به c اضافه می شود.

$a \ldots a b \ldots b c c$

می تواند c به ابتدای c اضافه شود ~~...~~ یک bc به طور هم زمان بین b و c وارد می کنیم

1) $S \rightarrow abc$

2) $S \rightarrow aXbc$

3) $Xb \rightarrow bX$

4) $Xc \rightarrow Ybcc$

5) $bY \rightarrow Yb$

6) aY

7) aa / aaX

Subject
YEAR: MONTH: DATE:

تمرین: مراحل اشتقاق رشته ی $w = a^3 b^3 c^3$ را طبق گرامر مثال قبل بنویسید

$S \rightarrow a\underline{xb}c \Longrightarrow ab\underline{xc} \Longrightarrow a\underline{by}bcc \Longrightarrow$

$\underline{ay}bbcc \Longrightarrow aa\underline{xb}bcc \Longrightarrow aab\underline{xb}cc \Longrightarrow$

$aabbxcc \Longrightarrow aab\underline{by}bccc \Longrightarrow aa\underline{by}bbccc$

$\Longrightarrow a\underline{ay}bbbccc \Longrightarrow aaabbbccc$

گرامر فوق، مستقل از متن نیست چون باید سمت چپ فقط یک متغیر داشته باشیم.

3) حساس به متن:

$\alpha \rightarrow \beta$

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

$\alpha \in (V \cup T)^+$

$|\alpha| \leq |\beta|$: یعنی کاهش طول نداشته باشیم.

گرامر مثال قبل، حساس به متن است.

$L = \{a^n b^n c^n \mid n \geq 0\}$ $w_1 = \lambda$

Subject
YEAR: MONTH: DATE:

راه کارهای برخورد با λ در گرامرهای حساس به متن:

راه کار اول: اگر گرامر تمام رشته های زبان را به استثنای λ تولید کرد،

به طور ضمنی گرامر را برای آن زبان می پذیریم.

راه کار دوم: $S' \to \lambda / S$

۶- گرامرهای بی قید و شرط:

$\alpha \to \beta$

$\beta \in (V \cup T)^{+}$ محدودیت نیست بلکه قانون تولید است

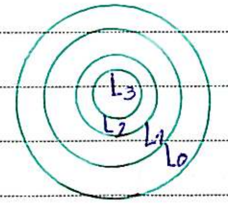
چون باید از یک جای شروع کنیم

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

اگر مسأله ای الگوریتم داشته باشد، حتماً یک گرامر بی قید و شرط دارد. پس اگر نتوانیم برای مسأله ای یک گرامر بی قید و شرط بنویسیم، آن مسئله غیر قابل حل است.

NP_com.

خانواده ی زبان ها



VAHDAT

خانواده‌ی زبان‌های غیر منتظم = مکمل خانواده‌های زبان‌های منتظم = $\overline{L_3}$

چون ما شامل مسائل لاینحل نمی‌شود. $\overline{L_3} \not\subset L_0$

مجموعه‌های منتظم:

۱) تمام نمادهای تکی الفبا یک مجموعه‌ی منتظمند.

اتحاد اجتماع

2) اگر X و Y دو مجموعه‌ی منتظم باشند، ترکیب‌های $X \cup Y$ ،

الحاق = اتصال = Concat ‏ مثبت بستار ‏ بستار

$X . Y$ و X^+ و Y^+ و X^* و Y^* هم منتظمند.

3) هر ترکیبی که بتوان با مجموعه عملیات بالا ساخت، نیز منتظم است.

عبارات منتظم:

۱) λ یک عبارت منتظم است.

2) تمامی اعضای Σ به تنهایی عبارت منتظمند.

3) اگر X و Y دو عبارت منتظم باشند، $X \cup Y$ و $X . Y$ و X^+ و X^* هم منتظمند.

۴) هر ترکیبی با اجرای عملیات فوق، منتظم است.

مثال: عبارت های زیر چه رشته هایی را نمایش می دهند؟

1) $a^+(a\cup b)^*$ هر عبارتی که حتماً با a شروع بشود.

$(a\cup b)^* = a^* \cup b^*$ a^+ یعنی می تواند a یا هر چیز غیر از a باشد.
$(a\cup b)^*$ شامل λ است.

2) $(a\cup b)^* c (a\cup b)^*$ رشته هایی که با a و b و c ساخته می شوند و حتماً و فقط یک c دارند.

3) $(a\cup b)^* c$ رشته هایی که فقط یک c دارند و به c ختم می شوند.

4) $(a\cup b)^+ = a^+ \cup b^+$ حتماً یکی از a و b باشد.

تمرین: برای زبان های داده شده، گرامر بنویسید و نوع گرامر را مشخص کنید.

1) $L = \{ w \mid w \in \{a,b\}^*,\ n_a(w) = n_b(w) \}$

2) $L = \{ w \mid w \in \{a,b,c\}^*,\ n_a(w) = n_b(w) = n_c(w) \}$

Subject
YEAR: MONTH: DATE:

?1) $S \to aSb \,/\, bSa \,/\, \lambda$

گرامر مستقل از متن است.

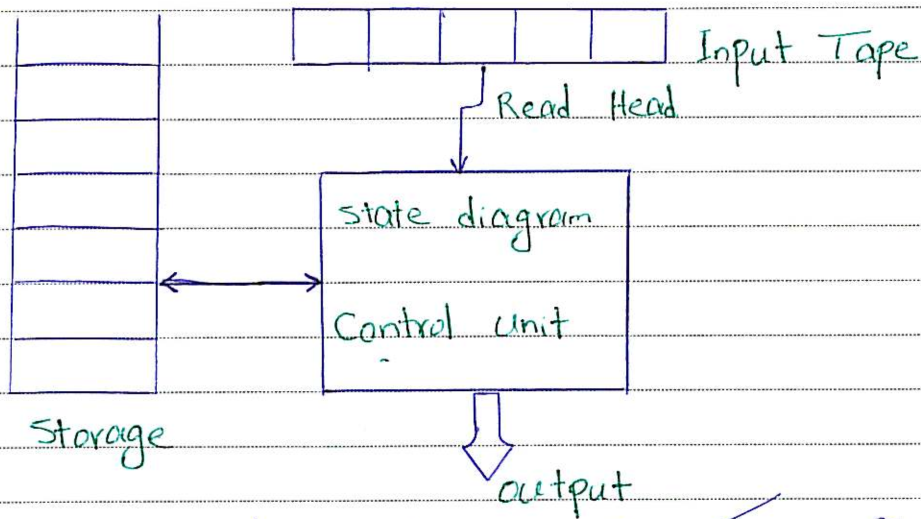
2) $S \to$

VAHDAT

Subject
YEAR: ۹۱ MONTH: ۸ DATE: ۸

ماشین (اتوماتا):

نمای یک ماشین کامل و استاندارد به صورت زیر است:



ماشین تشخیص می دهد رشته متعلق به زبان مورد نظر هست یا نه و در صورت لزوم تبدیل انجام می دهد.

انواع کلی ماشین

1- Acceptor (پذیرنده): اگر رشته‌ی داده شده به ماشین مطابق با زبان معادل ماشین باشد، yes تولید می کند و در غیر این صورت NO

2- Transducer (مبدل): بدون محدودیت روی Storage می نویسد.

ماشینی که شکل آن در بالا رسم شده، کامل است و معادل با زبان های بی قید و شرط است، این ماشین "ماشین تورینگ" نامیده می شود.
TM

Subject
YEAR: MONTH: DATE:

* با محدود کردن Storage به ماشین های کران دار خطی می رسیم که
Linear Bounded Automata

معادل گرامرهای حساس به متن است.

در این ماشین ها ابتدا و انتهای Storage به صورت خطی محدود می شود.

* با جایگزین کردن Stack با Storage، محدودیت زیادتر می شود و ماشین

پشته ای معادل گرامرهای مستقل از متن می سازیم.
stack push down

* با برداشتن Storage و تبدیل به ماشین متناهی، محدودیت بیشتر و معادل
Finite state A.
گرامرهای منظم می شود.

این نوع ماشین تعداد حالات متناهی دارد.

نکته: وقتی تعداد حالات متناهی باشد، زبان منظم است.

مثال:

$$\begin{cases} a^n b^m c^{n+m} d^n \\ 0<n , m<1000 \end{cases}$$

زبان منظم ⟸ متناهی

انواع مدل های ماشین:

1 - متناهی ⟵ گرامر منظم — Acc.

2 - پشته ای ⟵ گرامر مستقل از متن — Acc

3 - خطی کران دار ⟵ گرامر حساس به متن — Acc.

4 - تورینگ ⟵ گرامر بی قید و شرط — Trans.

Subject
YEAR: MONTH: DATE:

ماشین های متناهی:

$$M: (Q, \overset{\text{الفبا}}{\Sigma}, q_0, F, \delta)$$

Q: تعداد حالات

Σ: الفبا

q_0: نقطه ی شروع

F: نقطه ی پایانی

δ: تابع گذار — تابع انتقال — Transition Function

مثال:

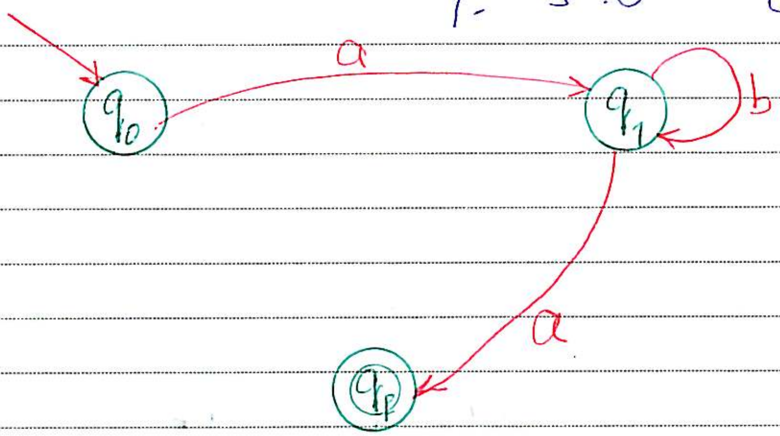
M_1:

$$Q = \{q_0, q_1, q_F\}$$

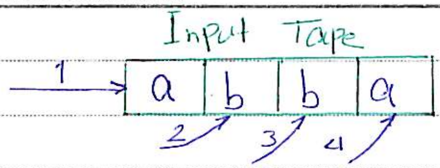
$$\Sigma = \{a, b\}$$

$$F = q_F$$

در q_0 به ازای a از q_0 به q_1 و در q_1 به ازای b انتقال به q_1 داریم و در q_1 به ازای a انتقال به q_F داریم.



می خواهیم ببینیم به ازای $w = abba$ ، yes تولید کند یا No؟



رشته در Input Tape قرار می‌گیرد و head از ابتدای رشته شروع به خواندن می‌کند و برگشت به عقب ندارد.

وقتی هد روی ابتدای رشته است، ماشین روی q_0 است.

ترتیب حالات ماشین: $q_0 \longrightarrow q_1 \longrightarrow q_1 \longrightarrow q_F$

وقتی رشته تمام می‌شود، در حالت پایانی هستیم پس پاسخ ماشین = yes

$w_1 = abbab$

$q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_1 \xrightarrow{b} q_1 \xrightarrow{a} q_F \xrightarrow{b} ?$

رشته تمام نشده ولی ماشین متوقف (Halt) شده، پس پاسخ = NO

ماشین‌های متناهی (FSA):

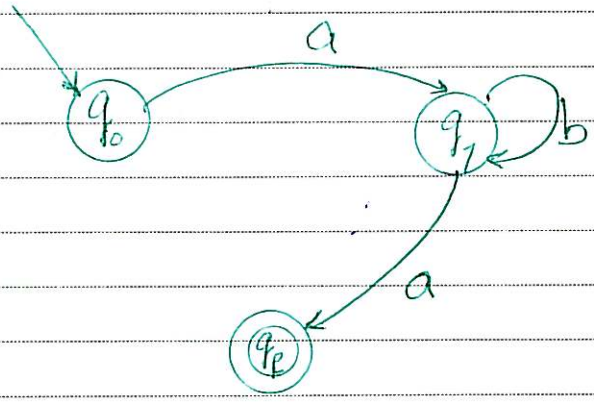
1_ قطعی Deterministic (DFA): در هر یک از حالت‌های ماشین به ازای هر کدام از الفبا حداکثر یک حرکت وجود داشته باشد.

2_ غیرقطعی Non Deterministic (NDFA): حداقل در یکی از حالت‌های ماشین، به ازای حداقل یکی از الفبا بیش از یک حرکت وجود داشته باشد.

Subject
YEAR: MONTH: DATE:

تابع گذار ماشین قطعی :

مثال :



ماشین قطعی است.

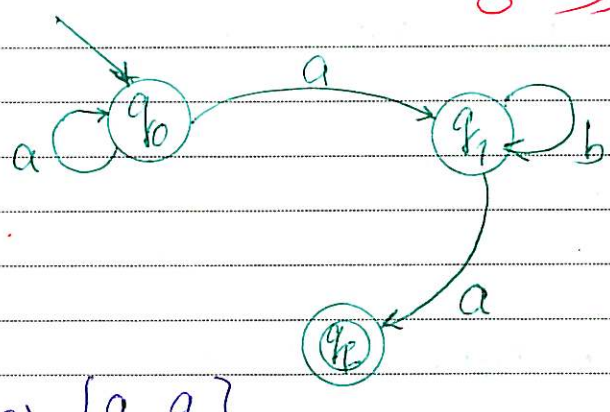
$\delta(q_0, a) = q_1$

$\delta(q_1, b) = q_1$

$\delta(q_1, a) = q_F$

تابع گذار ماشین غیر قطعی :

مثال :



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

$\delta(q_1, a) = q_F$

$\delta(q_1, b) = q_1$

VAHDAT

Subject
YEAR: MONTH: DATE:

پذیرش توسط DFA:

اگر با شروع از نقطه‌ی q_0، پس از اتمام رشته در یک موقعیت پایانی باشیم، رشته توسط این ماشین پذیرفته می‌شود و خروجی Yes تولید می‌کند.

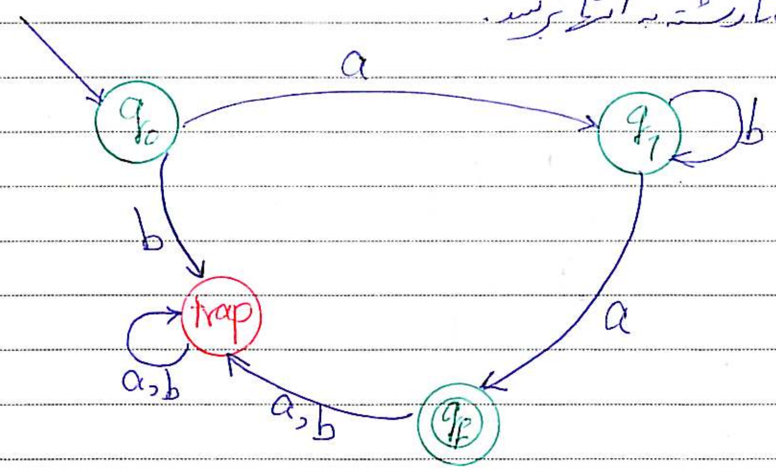
نکته: نقطه‌ی شروع ماشین همواره یکی است ولی حالات پایانی می‌توانند چندتا باشند.

عدم پذیرش توسط DFA:

اگر قبل از اتمام رشته، ماشین متوقف شود و یا پس از اتمام رشته در یک حالت غیرپایانی باشد، رشته توسط ماشین پذیرفته نمی‌شود و خروجی NO تولید می‌کند.

حالت Trap (دام): یک حالت غیرپایانی که وظیفه‌ی آن، مصرف ادامه‌ی رشته است تا رشته به انتها برسد.

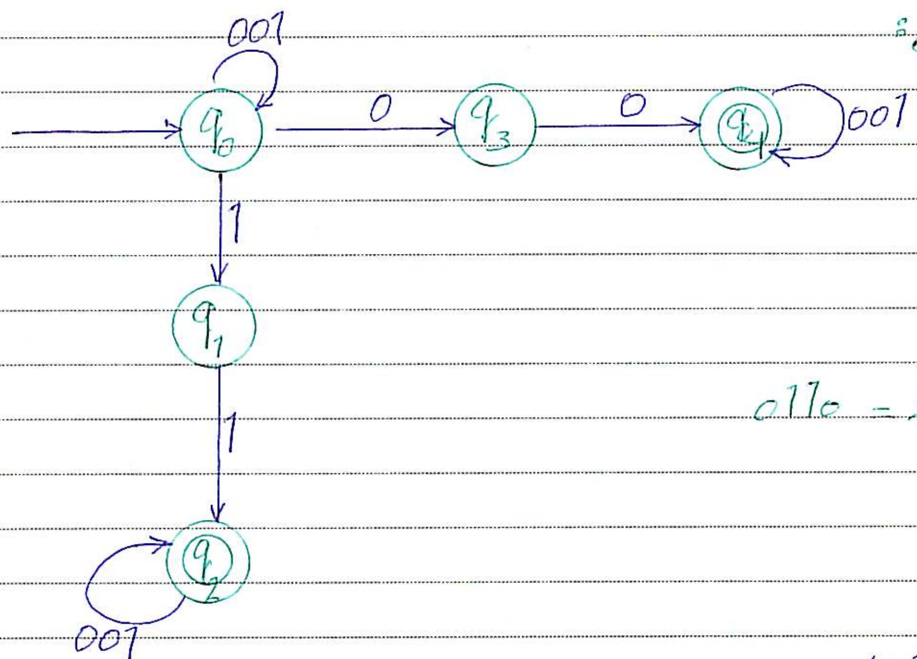
مثال:



- trap برای پیاده‌سازی سخت‌افزاری لازم است
- trap خروجی به حالات دیگر ندارد.
- trap همیشه یک loop به ازای کل الفبا به خودش دارد.

پذیرش و عدم پذیرش در ماشین متناهی غیرقطعی:

مثال:



رشته = 0110

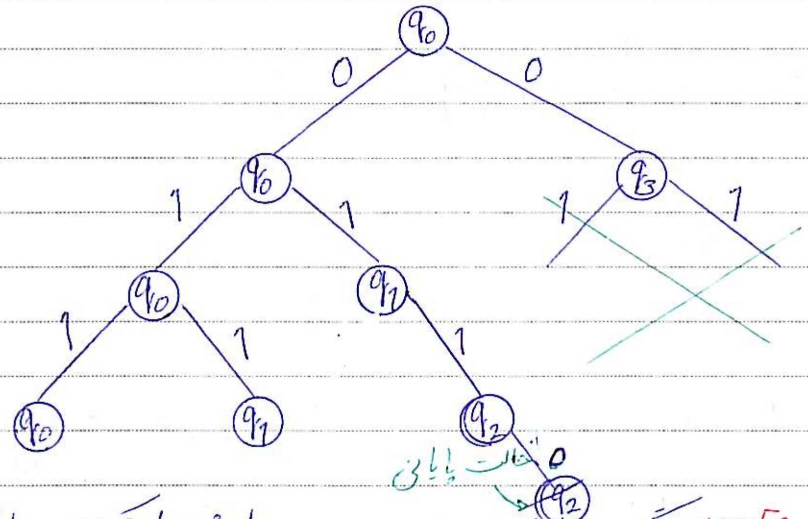
توابع گذار:

$$\delta(q_0, 0) = \{q_0, q_3\}$$

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

$\vdots$

ماشین یک درخت به صورت زیر رسم می کند:



پذیرش در NDFA: اگر حداقل یک مسیر وجود داشته باشد که پس از اتمام رشته در یک حالت پایانی باشد، خروجی Yes تولید می کند.

عدم پذیرش در NDFA: اگر هیچ مسیری وجود نداشته باشد که در پایان رشته، وضعیت ماشین پایانی باشد، خروجی NO تولید می کند.

زبان رسمی ماشین DFA:

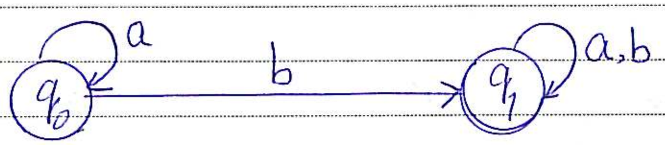
$$L(m)_{DFA} = \left\{ w \mid w \in \Sigma^* ,\ \underbrace{\delta^*}_{\text{(Extended) تابع گذار توسعه یافته}}(q_0, w) \in F \right\}$$

Subject
YEAR: MONTH: DATE:

تابع گذار توسعه یافته:

$$\begin{cases} \delta^*(q_0, a) = \delta(q_0, a) \\ \delta^*(q, wa) = \delta(\delta^*(q, w), a) \end{cases}$$

مثال:



$$\delta^*(q, \overbrace{abba}^{w}) = ?\ q_1$$

بصورت بازگشتی حل می کنیم:
یک a از انتها برمی داریم

$$\delta(\overbrace{\delta^*(q_0, abb)}^{q_1}, a) = ?$$

$$\delta^*(q_0, abb) = \delta(\overbrace{\delta^*(q_0, ab)}^{③\ q_1}, b) = ?$$

$$\delta^*(q_0, ab) = \delta(\overbrace{\underbrace{\delta^*(q_0, a)}_{①\ q_0}}^{②\ q_1}, b)$$

زبان پذیرش ماشین NDFA:

$$L_{NDFA}(m) : \{ w \mid w \in \Sigma^*, \delta^*(q_0, w) \cap F \neq \emptyset \}$$

VAHDAT

Subject
YEAR: MONTH: DATE:

مکمل زبان ماشین DFA:

$$\overline{L(m)}_{DFA} : \{\omega \mid \omega \in \Sigma^*, \delta^*(q_0, \omega) \notin F\}$$

مکمل زبان ماشین NDFA:

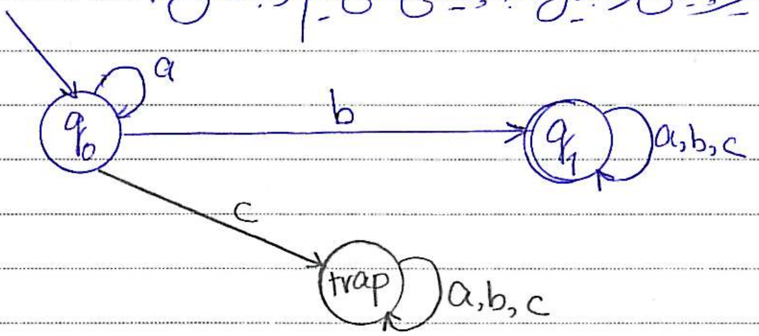
$$\overline{L(m)}_{NDFA} : \{\omega \mid \omega \in \Sigma^*, \delta^*(q_0, \omega) \cap F = \emptyset\}$$

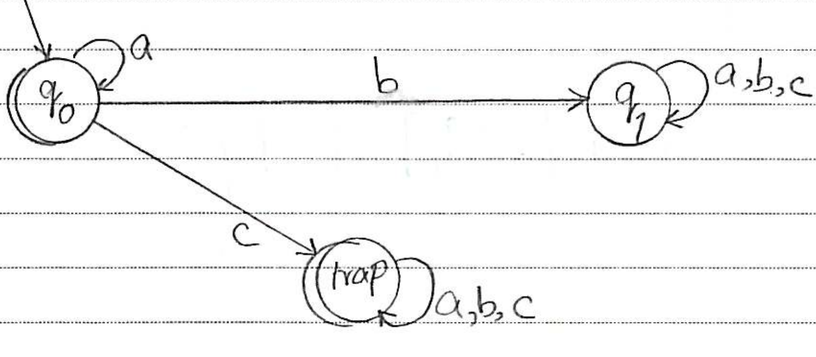
راه مکمل کردن یک DFA:

1) ابتدا trap را رسم می کنیم.

2) تمام حالات غیرپایانی را تبدیل به پایانی می کنیم و بالعکس.

مثال:





راه حل کردن یک NDFA :

1) NDFA را به DFA تبدیل می کنیم.

2) حل DFA تولید شده را به دست می آوریم.

تبدیل NDFA به DFA :

$$\underset{NDFA}{(Q, \Sigma, q_0, F, \delta)} \longrightarrow \underset{DFA}{(Q', \Sigma, q_0, F', \delta')}$$

$$\delta : Q \times (\Sigma \cup \{\lambda\}) \longrightarrow \underbrace{2^{Q}}_{\text{زیر مجموعه های } Q}$$

مجموعه توانی

$$\delta' : Q' \times \Sigma \longrightarrow Q'$$

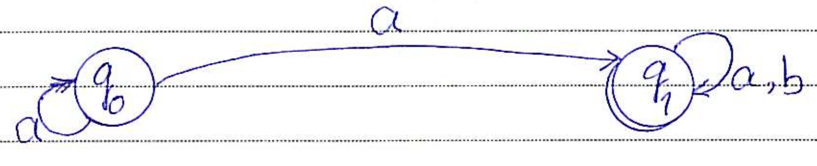
$\Downarrow$

از یک موقعیت با یک سری الفبا + λ می رویم به مجموعه ای از موقعیت ها

$\Downarrow$

از یک موقعیت با یک الفبا می رویم به یک موقعیت

مثال :



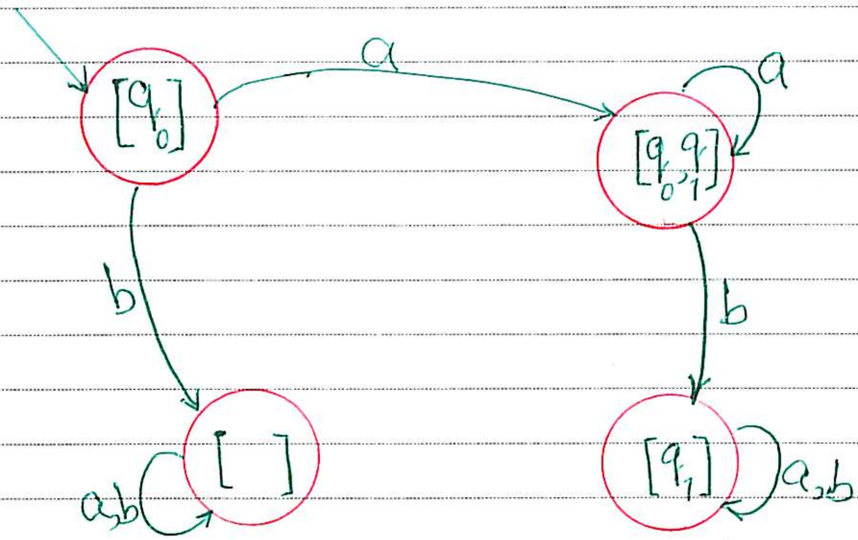
$$Q = \{q_0, q_1\}$$

$$Q' = \{[q_0], [q_0, q_1], [q_1], [\ ]\}$$

Subject

YEAR: MONTH: DATE:

δ'	a	b
$[q_0]$	$[q_0, q_1]$	$[\ ]$
$[q_0, q_1]$	$[q_0, q_1]$	$[q_1]$
$[q_1]$	$[q_1]$	$[q_1]$
$[\ ]$	$[\ ]$	$[\ ]$



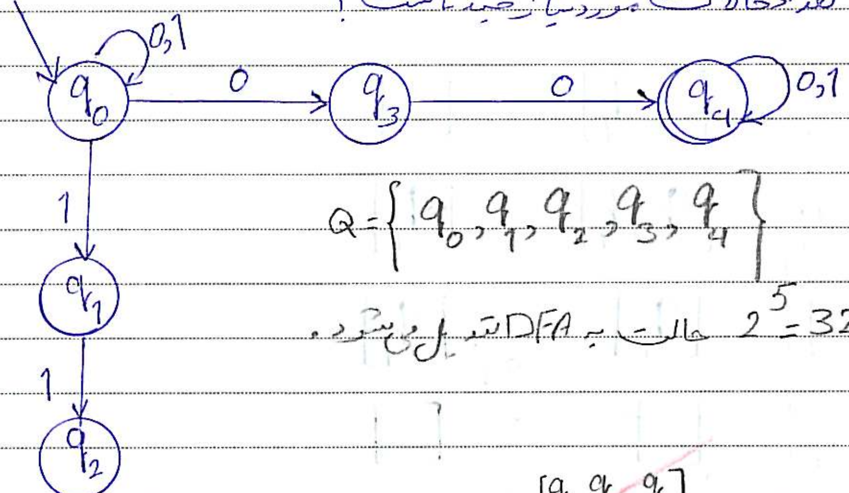
نکته1: اگر در نمودار تهی([ ]) داشتیم و کمانی غیر از loop داشت، همان trap است.

نکته 2: اگر حالتی در نمودار باشد که هیچ کمانی به آن وصل نشود، آن را حذف می کنیم.

نکته 3: اگر حالتی منحصراً یک خروجی داشته باشد (ورودی نداشته باشد)، حذف می‌شود

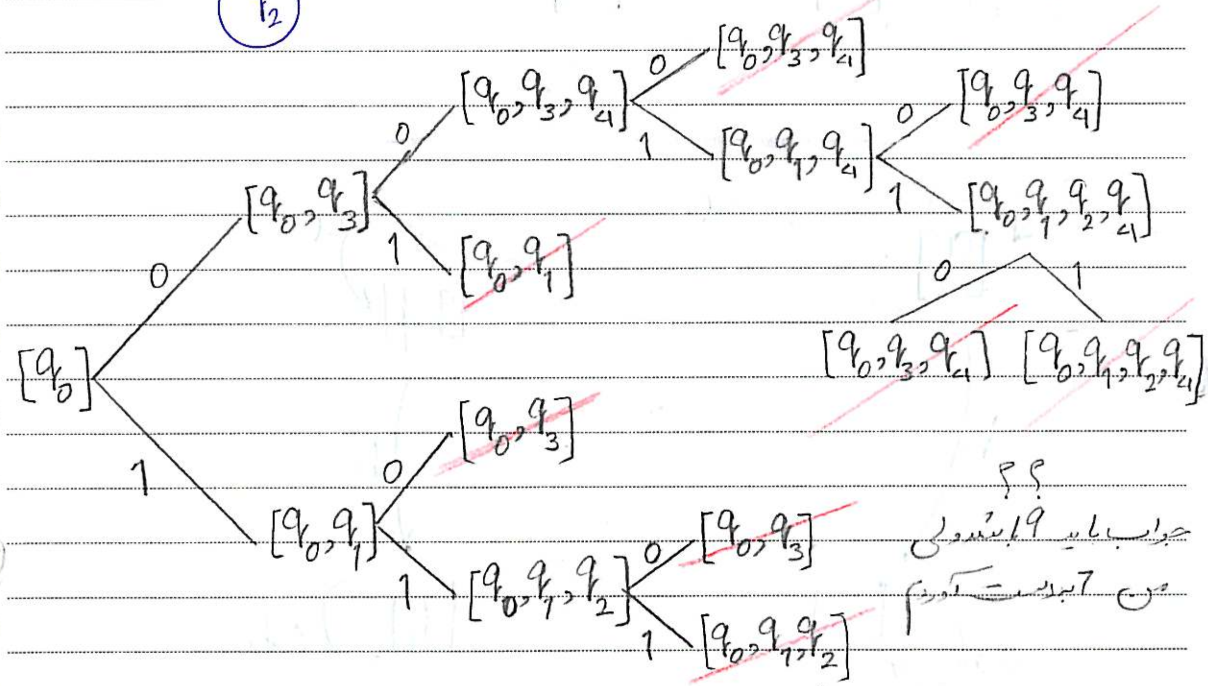
Subject
YEAR: MONTH: DATE:

تمرین ۴: دیاگرام داده شده با حداکثر چند حالت به DFA تبدیل می‌شود؟
تعداد حالات مورد نیاز چند تا است؟



$Q = \{q_0, q_1, q_2, q_3, q_4\}$

با حداکثر $2^5 = 32$ حالت به DFA تبدیل می‌شود.



؟؟
جواب باید 9 باشد ولی
من 7 بدست آوردم

نکته ۱: اگر ذیل مرحله سطحی ایجاد شود که در مراحل قبل ایجاد شده، حذف می‌شود و
ادامه می‌دهیم تا جایی همگی حذف شوند.

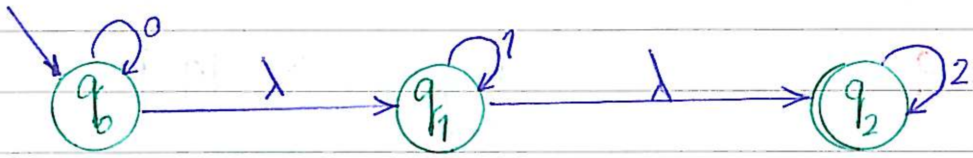
نکته 2: تعداد غیر تکراری‌ها، تعداد حالات مورد نظر است.

VAHDAT

۹۱/۸/۱۵

λ_NDFA:

$$\delta : Q \times (\Sigma \cup \{\lambda\}) \longrightarrow 2^{Q}$$



$$\Sigma = \{0, 1, 2\}$$

ابتدا همه ی مسیرهای λ را برای حالات مختلف می نویسیم:

λ_closure:

مسیر λ یا λ_closure مجموعه ای از حالاتی است که از حالت فعلی بدون صرف انرژی و تنها با حرکت λ می توانیم به آن حالات گذر کنیم.

$$\lambda\text{\_closure}(q_0) = \{q_0, q_1, q_2\}$$

$$\lambda\text{\_closure}(q_1) = \{q_1, q_2\}$$

$$\lambda\text{\_closure}(q_2) = \{q_2\}$$

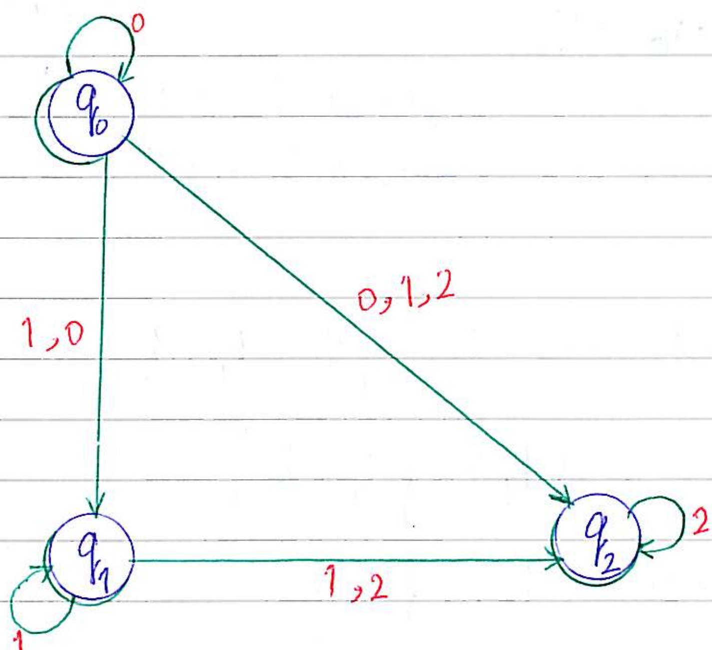
سپس باید λ_NDFA را به NDFA تبدیل کنیم:

از q_0 با صفر q_0 داریم $\Longrightarrow \lambda\_closure(q_0)$

تابع گذار:

δ	0	1	2
q_0	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
q_1	$\{\ \}$	$\{q_1, q_2\}$	$\{q_2\}$
q_2	$\{\ \}$	$\{\ \}$	$\{q_2\}$

از q_0 با یک، هیچ جا
از q_1 با یک: q_1
از q_2 با یک: هیچ جا
$\Longrightarrow$ بستار λ، q_1



F': $F \cup \{q_i \mid \lambda\_closure(q_i) \cap F \neq \emptyset\}$

یعنی چون در همه ی بستارهای λ، حالت پایانی (q_2) وجود دارد، همه ی حالات پایانی خواهند بود.

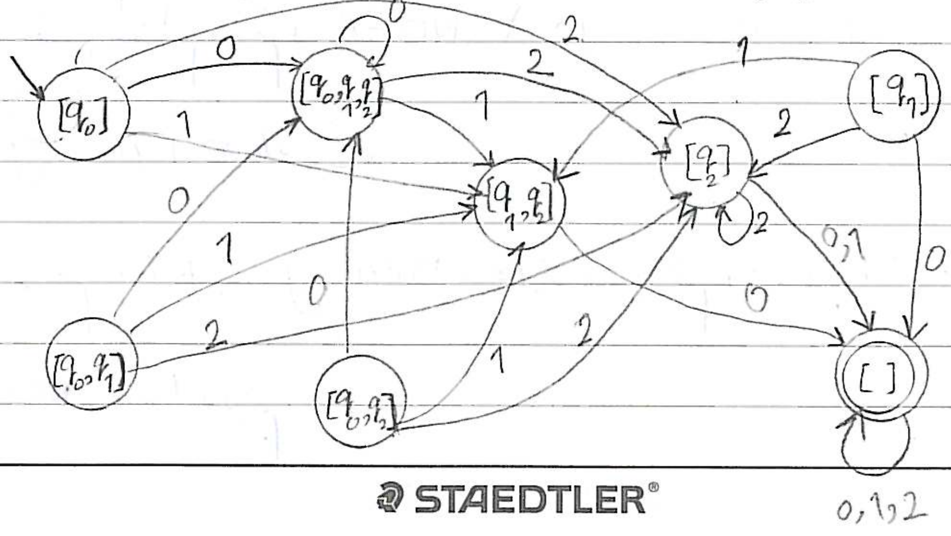
STAEDTLER®

در این ماشین، اگر q_1 را به حالت غیرپایانی تبدیل کنیم ، در پذیرفتن رشته ها
تفاوتی حاصل نمی شود. چون با 0،1،2 از q_0 به q_2 هم می توانیم برویم که حالت
پایانی است.

? تمرین: با ماشین مثال صفحه ی قبل را به DFA تبدیل کنید.

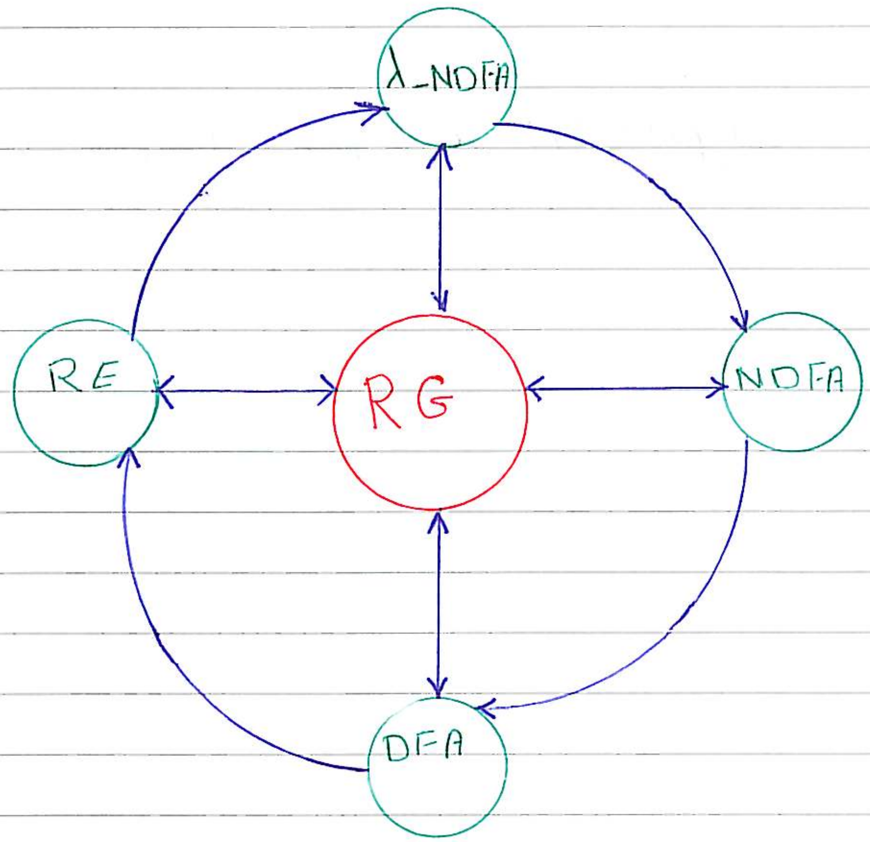
$Q = \{q_0, q_1, q_2\}$ $\quad Q' = \{[q_0], [q_0, q_1], [q_0, q_2], [q_1], [q_1, q_2], [q_2], [q_0, q_1, q_2], [\ ]\}$

	0	1	2
$[q_0]$	$[q_0, q_1, q_2]$	$[q_1, q_2]$	$[q_2]$
$[q_0, q_1]$	$[q_0, q_1, q_2]$	$[q_1, q_2]$	$[q_2]$
$[q_0, q_2]$	$[q_0, q_1, q_2]$	$[q_1, q_2]$	$[q_2]$
$[q_1]$	$[\ ]$	$[q_1, q_2]$	$[q_2]$
$[q_1, q_2]$	$[\ ]$	$[q_1, q_2]$	$[q_2]$
$[q_2]$	$[\ ]$	$[\ ]$	$[q_2]$
$[q_0, q_1, q_2]$	$[q_0, q_1, q_2]$	$[q_1, q_2]$	$[q_2]$
$[\ ]$	$[\ ]$	$[\ ]$	$[\ ]$



RG : گرامر منظم

RE : عبارت منظم

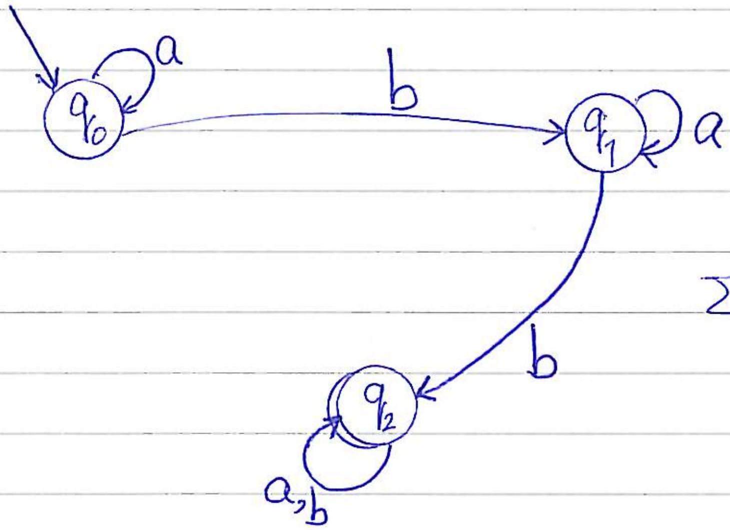


در گرامرهای منظم دیدیم که λ_NDFA قابل تبدیل به NDFA و NDFA قابل تبدیل به DFA است.

حالا می‌خواهیم مراحل تبدیل DFA به عبارات منظم و تبدیل عبارات منظم به λ_NDFA را ببینیم.

تبدیل DFA به RE :

مثال :



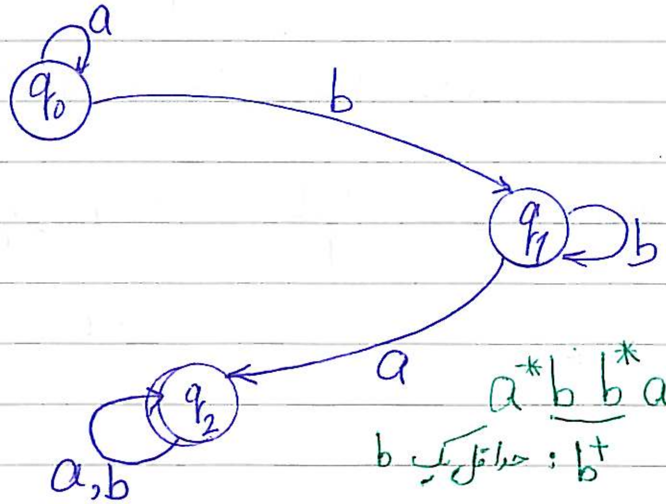
$\Sigma = \{a, b\}$

روش : از شروع به حالت پایان می‌رسیم و سعی می‌کنیم در حالت پایان بمانیم

همه‌ی loop ها ، * می‌گیرند . (a^* یعنی تعدادی a)

$$a^* b a^* b \underbrace{a^* b^*}_{(a+b)^*} \Longrightarrow a^* b a^* b (a+b)^*$$

مثال :



$$a^* \underline{b\, b^*} a (a+b)^*$$

b^+ : حداقل یک b

تمرین: برای عبارت مثال قبل، گرامر بنویسید.

$a^* b b^* a(a+b)^*$

$S \rightarrow ABaAC$

$A \rightarrow aA / \lambda$

$B \rightarrow b / bB$

$C \rightarrow bC / \lambda$

تبدیل عبارات منظم به $\lambda NDFA$:

خصوصیات عبارات منظم:

۱) $\emptyset$ عبارت منظم است.

2) λ عبارت منظم است.

3) هر $a \in \Sigma$ یک عبارت منظم است.

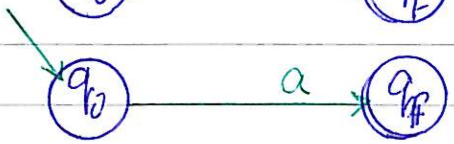
۴) اگر r_1 و r_2 عبارت منظم باشند $\Leftarrow$ $r_1 + r_2$ ، $r_1 . r_2$ ، r^* ، r^+ و (r)

عبارات منظمند.

۱) $\emptyset$: ماشینی که هیچ مسیری از شروع به پایان ندارد یا اصلاً حالت پایانی ندارد، $\emptyset$ را می‌پذیرد.

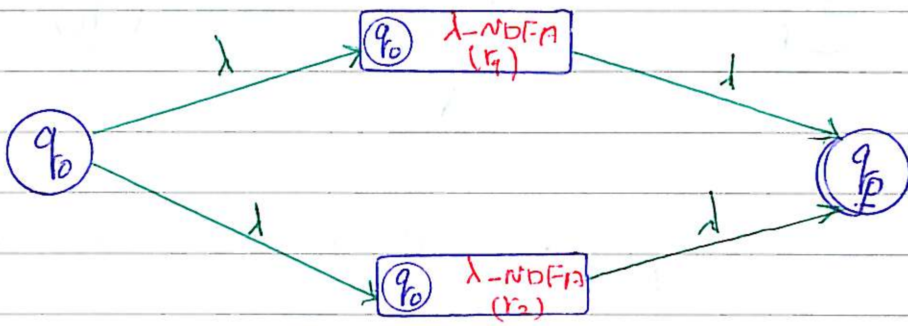
۲) λ :

۳) $a \in \Sigma$:

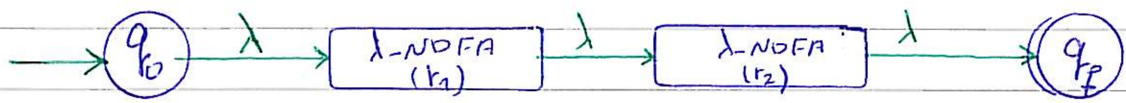


۴) r_1 و r_2 :

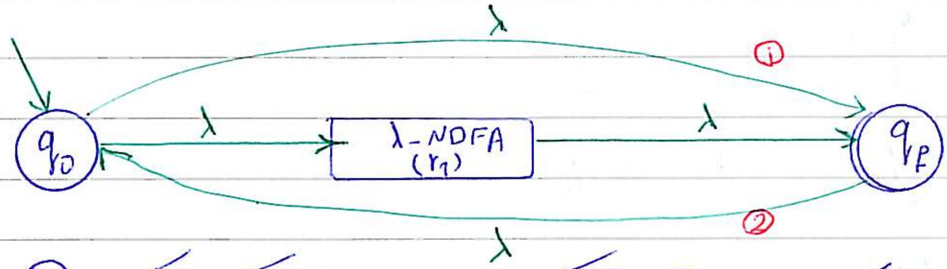
$$L(r_1 + r_2) = L(r_1) \cup L(r_2)$$



$$L(r_1 . r_2) = L(r_1) . L(r_2)$$

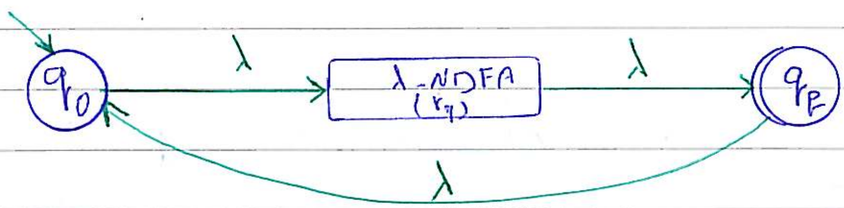


$L(r_1^*) = (L(r_1))^*$



تذکر: بدون کمان های ① و ② یک بار r_1 تولید می کند. با کمان ② چند بار تولید می کند و با کمان ① یا λ از q_0 به q_f می رود، چون استار است.

$L(r_1^+) = (L(r_1))^+$



تمرین: برای عبارات داده شده یک λ-NDFA رسم کنید.

1) $(a+b)^* aa (a+b)^*$

2) $(aa)^* (bb)^* b$

تمرین: برای عبارات تمرین قبل گرامر بنویسید.

تمرین: آیا دو عبارت داده شده، معادلند؟ اثبات کنید

1) $(1+01)^* (0+\lambda)$ 2) $(1^* 011^*)(0+\lambda)+1^*(0+\lambda)$

مثال: برای زبان های داده شده، گرامر بنویسید.

$L_1 = \{ a^n b^m \mid n < m \}$

$S_1 \longrightarrow aSb \,/\, Sb \,/\, b$ تعداد b ها یکی بیشتر از a یا خیلی بیشتر از a

$L_2 = \{ a^n b^m \mid n > m \}$

$S_2 \longrightarrow aSb \,/\, aS \,/\, a$

$L_3 = \{ a^{n_1} b^{m_1} a^{n_2} b^{m_2} a^{n_3} b^{m_3} \mid n_i > m_i \;\; i = 1, 2, 3 \}$

$A \longrightarrow aAb \,/\, aA \,/\, a$

$S_3 \longrightarrow AAA$

$L_4 = \{ a^n b^m \mid n \neq m \}$

$n < m \Rightarrow A_1 \longrightarrow aAb \,/\, Ab \,/\, b$

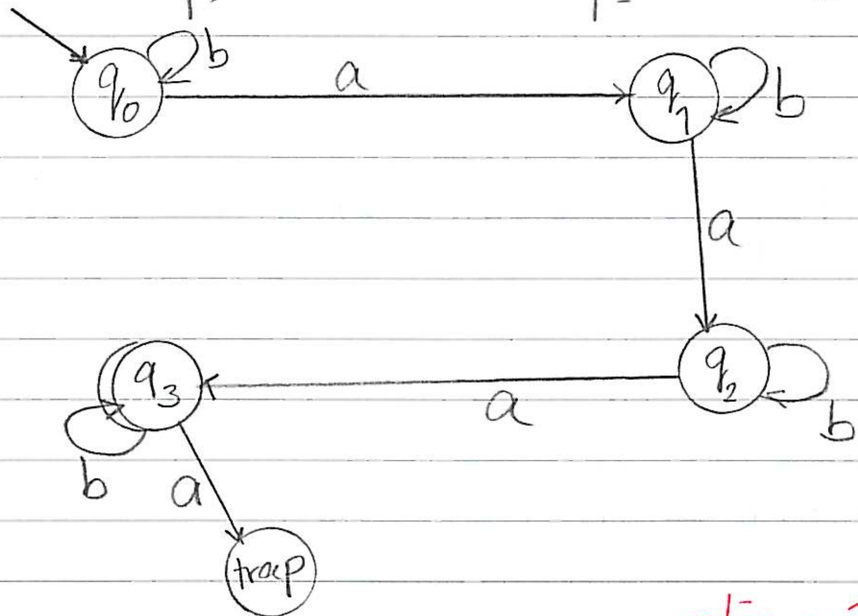
$n > m \Rightarrow A_2 \longrightarrow aAb \,/\, aA \,/\, a$

$S \longrightarrow A_1 / A_2$

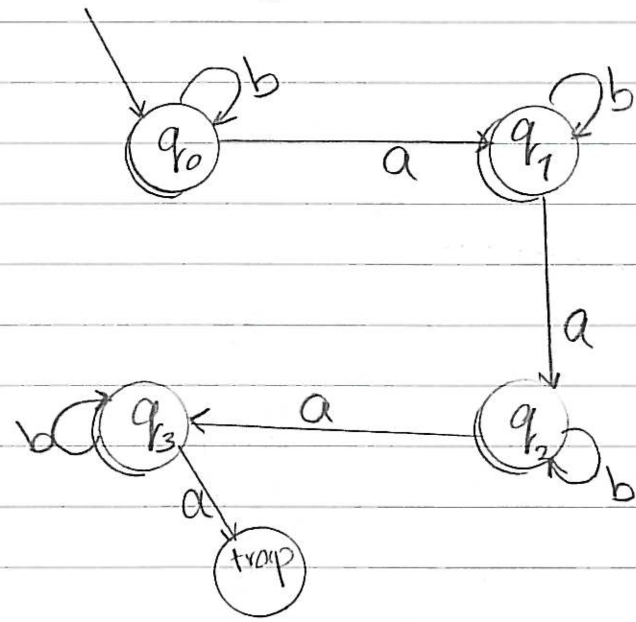
مثال: برای زبان داده شده یک DFA طراحی کنید:

زبانی شامل رشته های متشکل از a و b که دقیقاً 3 تا a داشته باشد.

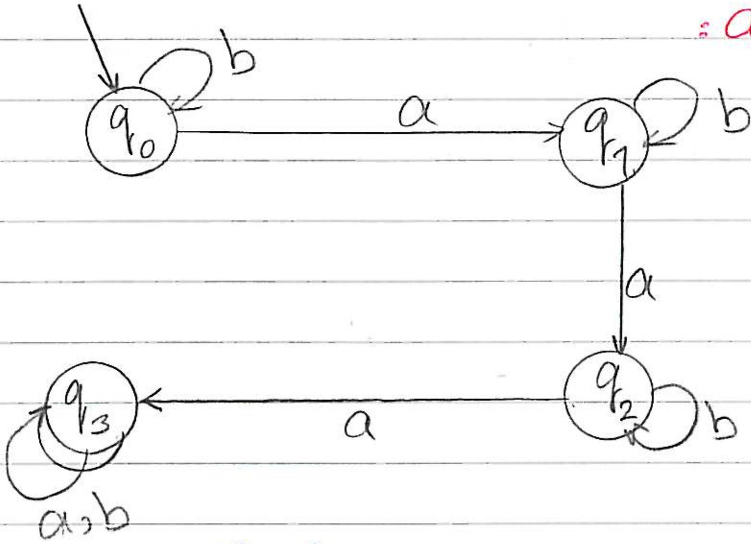
$\Sigma = \{a,b\}$ در DFA حافظه نداریم ⇐ شمارنده نداریم ⇐ باید با هر a تغییر حالت دهیم تا حساب تعداد a ها را داشته باشیم.



مثال قبل با حداکثر 3 تا a:



مثال قبل با حداقل 3 تا a :



?

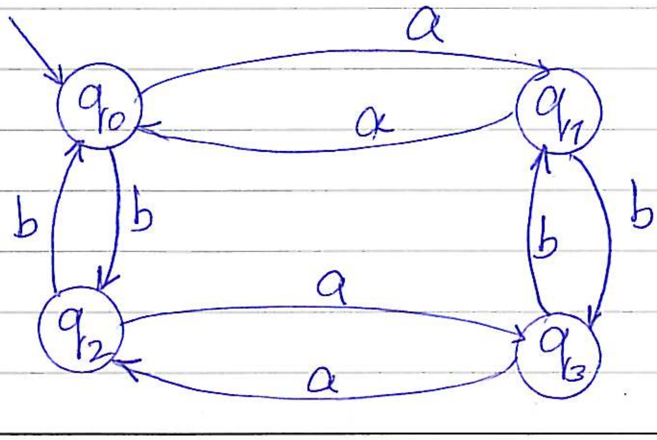
مثال : ماشین مثال قبل را به صورت عبارت منظم نشان دهید .

a دقیقاً 3 تا : $b^* a b^* a b^* a b^*$

a حداکثر 3 تا : $b^* + (b^* a b^*) + (b^* a b^* a b^*) + (b^* a b^* a b^* a b^*)$

a حداقل 3 تا : $b^* a b^* a b^* a (a+b)^*$

مثال : ماشین داده شده را تشریح کنید (در هر یک از حالات خواسته شده)



1) پایان ندارد: فقط تهی می پذیرد.

2) q_0، حالت پایانی است:

رشته هایی با تعداد a و b زوج را می پذیرد.

λ

a a

a b a b

b a a a b a

3) q_1، حالت پایانی است:

تعداد b زوج و تعداد a فرد را می پذیرد.

a

a b b

b a b

4) q_2، حالت پایانی است:

تعداد a زوج و تعداد b فرد را می پذیرد.

b

b b b

a b b

5) q_3، حالت پایانی است:

تعداد b و a فرد را می پذیرد.

a b

b a

تمرین: برای زبان داده شده، عبارت منظم و گرامر بنویسید.

1) $L = \{ a^n b^m \mid n, m \geq 3 \}$

2) $L = \{ a^n b^{n-3} \mid n \geq 3 \}$

حل :

۹۱/۸/۲۲

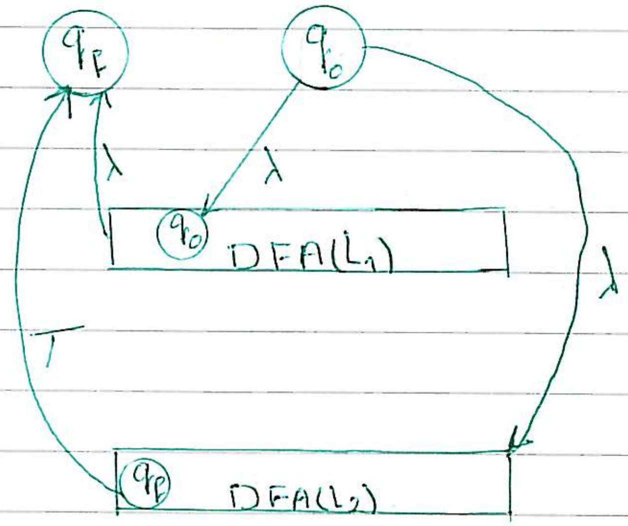
خصوصیات بستاری زبان های منظم:

قضیه ۱: اگر زبان های L_1 و L_2 منظم باشند، آنگاه $L_1 \cup L_2$ ، $L_1 \cap L_2$ ، $L_1 . L_2$ ، $\overline{L_1}$ ، L_1^* نیز منظمند. (یعنی خانواده ی زبان های منظم تحت عملگرهای $\cup$ ، $\cap$ ، . ، مکمل و استار بسته اند)

$\cup$:

اثبات:

$L_1 \cup L_2 \Longrightarrow$



از طریق گرامر:

$S_1 \rightarrow \dots$

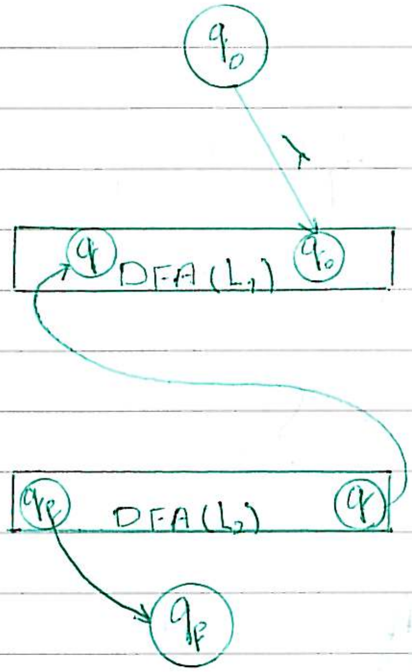
$S_2 \rightarrow \dots$

$S \rightarrow S_1 / S_2$

از طریق عبارت:

$r_1 + r_2$

Concat:



L_1

$L_2 \Longrightarrow$

ازطریق گرامر:

$S_1 \rightarrow \ldots \lambda \quad S_2 / aS_2$

منظور از $\bar{L}$ است $S_2 \rightarrow \ldots$

چپ $S_1 \rightarrow \ldots$

$S_2 \rightarrow \ldots \quad S_1 \lambda / S_2 a$

$r_1 \cdot r_2$

ازطریق عبارت:

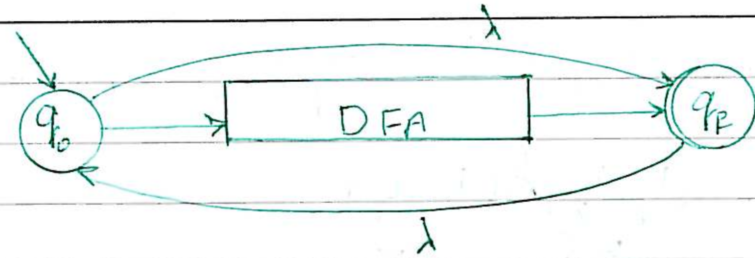
$\bar{L}$:

DFA

$\downarrow$ مکمل می‌شود با تبدیل نقاط پایانی به غیرپایانی و بالعکس

DFA $\Longrightarrow$ $\bar{L}$ منظم است.

L^*:



λ_NDFA قابل تبدیل به NDFA و NDFA قابل تبدیل به DFA است.

$L_1 \cap L_2$:

$$\underbrace{\overline{(\overline{L_1} \cup \overline{L_2})}}_{\text{منظم}} = \underbrace{L_1 \cap L_2}_{\text{منظم}}$$

منظم ← منظم

قضیه‌ی 2: خانواده‌ی زبان‌های منظم تحت عملگر Riverse بسته است.

L^R: L: DFA

طراحی $DFA(L^R)$ از روی $DFA(L)$:

1) DFA را طوری طراحی می‌کنیم که فقط یک شروع و پایان داشته باشد.

2) جای پایان و آغاز را عوض می‌کنیم.

3) همه‌ی یال‌ها را برعکس می‌کنیم.

قضیه‌ی 3: خانواده‌ی زبان‌های منظم تحت عملگر تفاضل بسته است.

$$\underbrace{L_1 - L_2}_{\text{منظم}} = \underbrace{L_1 \cap \overline{L_2}}_{\text{منظم}}$$

منظم ← منظم

قضیه‌ی ۴: خانواده‌ی زبان‌های منظم تحت عملگر تفاضل متقارن، بسته است.

$$L_1 \mapsto L_2 = \underbrace{(L_1 - L_2)}_{\text{منظم}} \overset{\cup}{\cap} \underbrace{(L_2 - L_1)}_{\text{منظم}}$$

منظم $\Leftarrow$ منظم

قضیه‌ی ۵: خانواده‌ی زبان‌های منظم تحت عملگر زیرمجموعه، بسته نیست.

(یعنی همه‌ی زیرمجموعه‌های زبان منظم، منظم نیستند)

مثال: $L_1 : (a+b)^*$ همه‌ی عباراتی که با هر تعداد a و b ساخته می‌شوند.

$L_2 : \{ww^R \mid w \in \{a,b\}^*\}$ همه‌ی رشته‌های متقارن که با a و b ساخته شوند.

$w = 001 \Rightarrow w^R = 100 \Rightarrow ww^R = 001|100$

$L_2 \subseteq L_1$ $\qquad$ L_2 نامنظم است. (اثبات در صفحه‌ی بعد)

روش تشخیص زبان‌های منظم:

۱) تمام زبان‌های متناهی حتماً منظمند. (تعداد رشته‌های متناهی)

۲) اگر برای زبان نیاز به شمارش نامتناهی بود و یا نیاز به ذخیره‌سازی بود، زبان نامنظم است.

۳) اگر برای رسم ماشین متناهی زبان، تعداد حالات نامعین باشد، زبان نامنظم است.

می‌خواهیم ثابت کنیم ww^R نامنظم است:

1) $q_0 \xrightarrow{\lambda} q_f$

2) $q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_f$

3) $\xrightarrow{b}$ $\xrightarrow{b}$

4) a b b a

5) b a a b

پس تعداد حالات بی‌شماری است.

قضیه‌ی 6: خانواده‌ی زبان‌های منظم تحت هومومورفیسم بسته است.

هومومورفیسم: تابعی است از Σ به Γ^* (الفبا) یعنی هر یک از الفبا را رشته‌ای جایگزین می‌کند.

مثال: $\Sigma = \{a, b\}$

$$f: \begin{cases} a \to 111 \\ b \to 000 \end{cases}$$

$w = aba \to$ حاصل زبان L

$w = 111\,000\,111 \to$ حاصل هومومورفیسم L

روی DFA یعنی اگر از حالتی با a به حالت دیگری می‌رویم حالا با 111 برویم. مثل این که چند حالت میانی اضافه کرده‌ایم.

قضیه ی 7: خانواده ی زبان های منظم تحت عملگر خارج قسمت از راست بسته است. (یعنی اگر L_1 منظم و L_2 منظم باشد ← L_1/L_2 منظم است)

$$L_1/L_2 = \{x \mid xy \in L_1 \text{ for some } y \in L_2\}$$

مثال:

$$\begin{cases} L_1 = \{abb, bba, aaa\} \\ L_2 = \{bb, a\} \end{cases}$$

$$L_1/L_2 = \{a, bb, aa\}$$

bb از abb جدا شده — a از bba جدا شده — a از aaa جدا شده

در بزرگ ترین حالت، L_2 با λ برابر است، پس $L_1/L_2 = L_1$ که منظم است.

مثال:

$$L_1 = \{a^n b^m \mid n \geq 1, m \geq 0\} \cup \{ba\}$$

$$L_2 = \{b^m \mid m \geq 0\}$$

$$L_1/L_2 = ?$$

کوچک ترین حالت L_2 برابر با λ است. بنابراین اگر λ را از آخر هر رشته برداریم:

$$L_1/L_2 = \{a^n b^m \mid n \geq 1, m \geq 0\} \cup \{ba\}$$

مثال:

$$L_1 = \{a^n b^m \mid n \geq 1, m \geq 0\} \cup \{ba\}$$

$$L_2 = \{a^n b^m \mid n \geq 1, m \geq 0\}$$

$$L_1 / L_2 = ?$$

کوچک‌ترین حالت L_2 برابر با a است:

$$L_1 / L_2 = \{a^k \mid k \geq 0\} \cup \{b\}$$

L_1 : aaabbbb

L_2 : aabb $\Rightarrow$ امکان‌پذیر نیست

L_2 : aabbbb $\Rightarrow L_1/L_2 = a$

L_2 = aaabbbb $\Rightarrow L_1/L_2 = \lambda$

قضیه ۸: خانواده‌ی زبان‌های منظم تحت عملگر head ~~منتظم~~ بسته است.

head : همه‌ی پیشوندهای یک زبان را برمی‌گرداند.

$$head(L) = \{x \mid xy \in L, y \in \Sigma^*\} = L / \Sigma^*$$

مثال:

$$L = \{001, 100, 111\} \quad , \quad \Sigma = \{0, 1\}$$

$$head(L) = \{\underbrace{001, 100, 111}_{\text{از انتها برداشته شده } \lambda}, 10, 00, 11, \underbrace{\lambda}_{\text{کل رشته برداشته شده}}, \ldots\}$$

نکته:

$$\left.\begin{array}{l} L \subset head(L) \\ if\ (L = \Sigma^* \text{ یا } L = \{\lambda\}) \Longrightarrow L = head(L) \end{array}\right\} \Rightarrow L \subseteq head(L)$$

قضیه ۹: الگوریتمی وجود دارد که می‌تواند تعیین کند آیا رشته‌ی $w \in \Sigma^*$ عضو زبان L هست یا خیر. (L به صورت فرم استاندارد نمایش داده شده است)

فرم استاندارد: DFA برای زبان رسم شده است.

الگوریتم: پیمایش DFA با رشته‌ی داده شده.

قضیه ۱۰: الگوریتمی وجود دارد که می‌تواند تعیین کند زبان L که به فرم استاندارد نمایش داده شده است، متناهی یا نامتناهی یا تهی است.

تشخیص تهی بودن: ۱) از شروع به پایان مسیری وجود نداشته باشد. یا
۲) DFA، پایان نداشته باشد.

تشخیص متناهی یا نامتناهی بودن: اگر از q_0 تا q_F حلقه‌ای وجود داشت (با واسط بین چند حالت یا بی واسطه روی یک حالت)، زبان نامتناهی است. درغیر این صورت، زبان متناهی است.

قضیه‌ی 11: الگوریتمی وجود دارد که می‌تواند تعیین کند آیا دو زبان منظم L_1 و L_2 مساوی هستند یا خیر.

الگوریتم: $if\ (L_1 \ominus L_2 = \varnothing) \Rightarrow L_1 = L_2$

قضیه‌ی 12: الگوریتمی ارائه دهید که تعیین کند آیا زبان منظم L، پالیندرام (Palindrome) هست یا نه؟

Polindrome: رشته‌هایی که با معکوسشان برابرند یعنی متقارنند. $w = w^R$

مثال: 100¦001

زبان پالیندرام: زبانی که هر رشته‌ی معکوس آن هم در زبان باشد.

الگوریتم: از روی DFA(L)، معکوس DFA(L) را رسم می‌کنیم. اگر برابر بودند، زبان پالیندرام است.

قانون Pumping: (Pumping Lema)

این قانون برای اثبات نامنظم بودن زبان به کار می‌رود. اگر به نتیجه‌ی نامنظم بودن نرسیدیم، هیچ تصمیمی نمی‌گیریم.

متن قانون: اگر زبان منظم و نامتناهی L وجود داشته باشد، آنگاه عدد صحیح و مثبت m وجود خواهد داشت به گونه‌ای که به ازای هر رشته‌ای که عضو L باشد، طول w بزرگتر یا مساوی m باشد. ($|w| \geq m$)، در آن صورت می‌توان رشته‌ی w را به سه زیر رشته‌ی x و y و z تقسیم کرد به طوری که $|xy| \leq m$ و $|y| \geq 1$ (یعنی هیچ کدام λ نباشند).

آنگاه رشته‌ی pump شده‌ی w_i عضو L است. $(i = 0, 1, 2, \dots)$

$$w_i = x\,(y)^i\,z$$

اثبات: زبان منظم است ⇐ DFA دارد.

زبان نامتناهی است ⇐ Loop دارد.

اگر $i=0$ ⇐ در دور نیفتاده‌ایم. $x(y^i)z \in L$

روش کار: چون $p \Rightarrow q$ پس از $\sim q \Rightarrow \sim p$ استفاده می‌کنیم.

مثال: با استفاده از قانون pumping ثابت کنید زبان $L=\{a^n b^n \mid n \geq 0\}$ نامنتظم است.

حریف	شما
① m	② $w=a^m b^m$
③ $xy=a^m$, $x=a^{m-k}$; $y=a^k$ $1 \leq k \leq m$; $z=b^m$	④ $w_i=a^{m-k}(a^k)^i b^m$

$\Downarrow$

اگر $i=0 \Leftarrow w=a^{m-k}b^m$

$\Leftarrow$ $w \notin L$ $\Leftarrow$ زبان L نامنتظم است.

مثال: با استفاده از قانون pumping ثابت کنید زبان $L=\{ww^R \mid w=\{a,b\}\}$ نامنتظم است.

حریف	شما
① m	② $w=\underbrace{a^m}_{xy}\underbrace{a^m}_{z}$
③ $x=a^{m-1}$; $y=a^1$; $z=a^m$	④ $w_i=a^{m-1}(a)^i a^m \notin L$

$i=0 \Rightarrow w_0=a^{m-1}a^m \Rightarrow$

زبان منتظم نیست

حل: با در نظر گرفتن رشته ی دلخواه:

حریف	شما
① m	② $w = a^m b^m b^m a^m$
③ $\begin{cases} x = a^{m-k} \\ y = a^k \\ z = b^m b^m a^m \end{cases}$	④ $w_i = a^{m-k} (a^k)^i b^m b^m a^m$

$i = 0 \Rightarrow w_0 = a^{m-k} b^m b^m a^m \notin L \Rightarrow$

زبان نامنظم است.

۹۱/۹/۶

فصل 5: گرامرهای مستقل از متن:

$A \to \alpha$

$\alpha \in (V \cup T)^*$ سمت چپ تمام قواعد گرامر تک متغیر باشد.

انواع اشتقاق:

1) LMD: همیشه سمت چپ ترین متغیر را باز می کنیم.

2) RMD: همیشه سمت راست ترین متغیر را باز می کنیم.

3) D_tree: درخت اشتقاق یا پارس

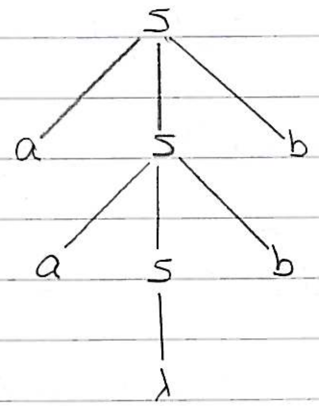
بالا به پایین

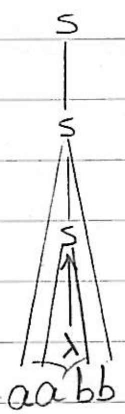
پایین به بالا

مثال: گرامر زبان $L=\{a^n b^n \mid n \geq 0\}$ را بنویسید و رشته a^2b^2 را تولید کنید و از aabb به S برسید.

$S \to aSb / \lambda$

تولید رشته a^2b^2





مثال: زبان گرامر مقابل را بنویسید. سپس مراحل اشتقاق a^2b^3 را بنویسید.

$$L = \{a^n b^m \mid m>n\}$$

$S \rightarrow AB$
$A \rightarrow aAb/\lambda$ } تا این قسمت $a^n b^n$ تولید می‌شود
$B \rightarrow bB/b$ → $\left\{ \begin{array}{l} a^n b^m \\ m>n \end{array} \right.$ را تولید می‌کند یعنی $a^n b^n b$

از چپ:

$$S \Rightarrow AB \Rightarrow aAbB \Rightarrow aaAbbB \Rightarrow a^2b^2B \Rightarrow a^2b^2b \Rightarrow a^2b^3$$

از راست:

$$S \Rightarrow AB \Rightarrow Ab \Rightarrow aAbb \Rightarrow aaAbbb \Rightarrow a^2b^3$$

قضیه: اگر G یک گرامر مستقل از متن و فاقد قواعد $A \rightarrow \lambda$ و $A \rightarrow B$ باشد، در آن صورت حتماً یک روش جستجوی کلی وجود دارد که به ازای هر $w \in \Sigma^*$ یا w را تجزیه می‌کند و یا بیان می‌کند که این رشته توسط این الگوریتم قابل تجزیه یا تولید نیست.

الگوریتم جستجوی کلی:

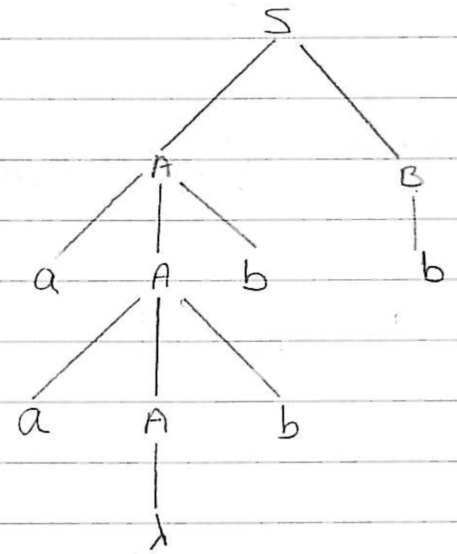
از start شروع می‌کنیم و درخت های موازی را تا مرحله $K+1$ رسم می‌کنیم.

$(|w|=k)$ سپس به مرحله ۲ می‌بریم. اگر در مرحله k ام رشته ساخته شده بود

رشته توسط این گرامر قابل تجزیه و تولید هست، در غیر این صورت نیست.

مثال: $L=\{a^n b^m \mid m>n\}$ ، $w=a^2b^3$

$|w|=5 \Longrightarrow k+1=6$



قبل از مرحله ششم به جواب رسیدیم چون $A \to \lambda$ داریم.

قضیه: در هر گرامر مستقل از متن، الگوریتمی وجود دارد که رشته‌ی $w \in L(G)$

را در تعداد مراحلی متناسب با طول رشته به توان 3 $(|w|^3)$، تجزیه یا پارس کند.

نکته: همیشه ابتدا با فرم‌های نرمال بعد با حذف $A \to B$ و $A \to \lambda$ و بعد با این

قضیه بررسی می‌کنیم.

Simple Grammer:

به گرامر مستقل از متن که شرایط زیر را داشته باشد Simple Grammer می گویند:

1) $\begin{cases} A \rightarrow aX \\ a \in T \\ X \in V^* \end{cases}$

یعنی تمام قواعد باید با یک پایانه شروع شوند و حداکثر یک پایانه داشته باشند و بقیه غیر پایانه باشند.

2) سمت راست هیچ دو قاعده‌ای A (مانند $A \rightarrow ab/aA$) با ترمینال یکسان شروع نشود.

مثال: گرامر $S \rightarrow aSb/ab$ را ساده کنید.

$a(SB/B)$

$$\begin{cases} S \rightarrow aSB/aB \\ B \rightarrow b \end{cases} \Rightarrow \begin{cases} S \rightarrow aZ \\ Z \rightarrow SB/B \\ B \rightarrow b \end{cases} \Rightarrow \begin{cases} S \rightarrow aZ \\ Z \rightarrow aZB/b \\ B \rightarrow b \end{cases}$$

فاکتورگیری　　　　جایگذاری

مثال: گرامر داده شده را به فرم simple تبدیل کنید.

$$\begin{cases} S \rightarrow AB \\ A \rightarrow aAb \\ B \rightarrow bB/b \end{cases}$$

$S \rightarrow aAbB/B$

bZ

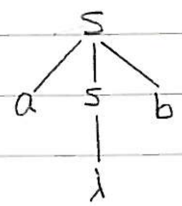
$$\begin{cases} S \rightarrow aAX_bB/bB/b \\ B \rightarrow bB/b \\ X_b \rightarrow b \end{cases} \rightarrow \begin{cases} B \rightarrow bZ \\ Z \rightarrow B/\lambda \\ Z \rightarrow bZ/\lambda \end{cases}$$

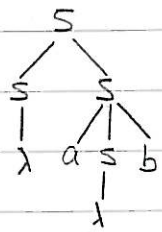
STAEDTLER

گرامرهای مبهم: اگر در یک گرامر مستقل از متن به ازای حداقل یک رشته که $w \in L(G)$ ، حداقل دو درخت اشتقاق یا دو اشتقاق از راست یا دو اشتقاق از چپ متفاوت وجود داشته باشد، گرامر مبهم است.

مثال:

$$\begin{cases} S \to SS / aSb / \lambda \\ w = ab \end{cases}$$





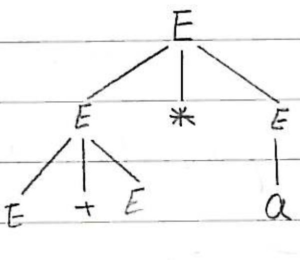
دو درخت متفاوتند $\Rightarrow$ گرامر مبهم است

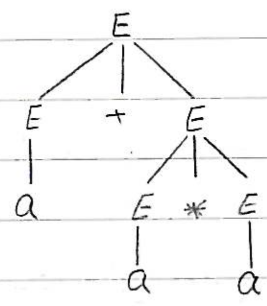
مثال:

$$\begin{cases} E \to E + E \\ E \to E * E \\ E \to (E) \\ E \to a \end{cases}$$

$w = a + a * a$

نکته: در گرامرهای مانند این گرامر، همه‌ی عملگرها (+ ، * ، (،)) جزء الفبا هستند. (ترمینالند)





نکته: عملیاتی که در درخت، پایین‌تر قرار بگیرند، اولویت بالاتری دارند.

Subject: Date:

رفع ابهام:

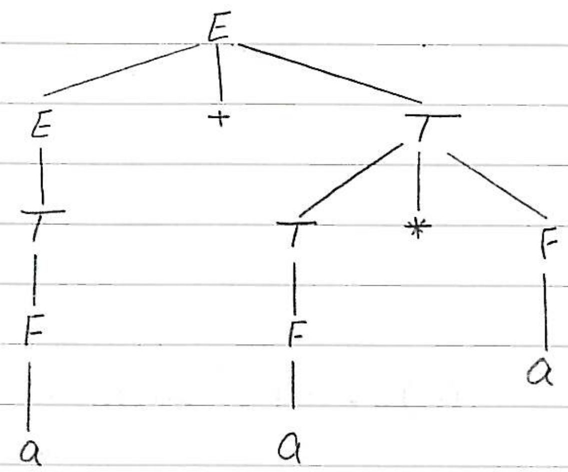
مثال قبل:

$$\begin{cases} E \rightarrow E + E \\ E \rightarrow E * E \\ E \rightarrow (E) \\ E \rightarrow a \end{cases}$$

$E \rightarrow E + T \,/\, T$

$T \rightarrow T * F \,/\, F$

$F \rightarrow (E) \,/\, a$



نکته ی 1: برای تشخیص ابهام در گرامرهای مستقل از متن و حتی برای رفع ابهام، هیچ الگوریتم مشخصی وجود ندارد.

نکته ی 2: اگر در گرامری هم راست گردی (بازگشتی از راست) و هم چپ گردی به ازای یک متغیر وجود داشت، آن گرامرها مبهم است. ولی ممکن است گرامری فاقد راست گردی و چپ گردی باشد ولی باز هم مبهم باشد.

Subject: Date:

نکته‌ی 3: زبان‌های ذاتاً مبهم: اگر برای یک زبان مستقل از متن، هیچ گرامر مستقل از متن غیر مبهمی وجود نداشت، آن زبان ذاتاً مبهم است.

مثال: گرامر زبان زیر را بنویسید و تعیین کنید زبان مبهم است یا نه؟

$$L = \underbrace{\{a^n b^m c^m\}}_{S1} \cup \underbrace{\{a^n b^n c^m\}}_{S2}$$

$S \to S1 / S2$

$S1 \to aS1 / G$

$G \to bGc / \lambda$

$S_2 \to AB$

$A \to aAb / \lambda$

$B \to cB / \lambda$

رشته‌ی $w = a^n b^n c^m$ را در نظر می‌گیریم. می‌توانیم هم از طریق S1 و هم از طریق S2 آن را تولید کنیم، پس مبهم است.

Subject: Date:

فصل 6: ساده سازی و فرم های نرمال گرامرهای مستقل از متن:

ساده سازی:

1) حذف قواعد λ:

قاعده ی λ: هر قاعده به صورت $A \to \lambda$ قاعده ی λ است و هر متغیری که در صفر یک یا بیشتر مرحله به λ برسد ($A \overset{*}{\Rightarrow} \lambda$)، عنصر میرا (پوچ یا nullable) است.

2) حذف قواعد واحد (unit): $A \to B$

3) حذف قواعد بی فایده (useless): متغیری که حداقل به ازای یکی از رشته های زبان در مسیر اشتقاق ظاهر شده باشد، مفید خوانده می شود. در غیر این صورت غیر مفید است. یعنی هیچ وقت به آن نمی رسیم یا اگر برسیم نمی توانیم آن را از بین ببریم.

$S \to aSb$

$B \to b$ ← useless

نکته: باید دقت کنیم ساده سازی را با Simple Grammer اشتباه نکنیم.

Subject: Date:

مثال: گرامر داده شده را ساده کنید.

$$\begin{cases} S \to ABaC \\ A \to BC \\ B \to b / \lambda \\ C \to D/\lambda \\ D \to d \\ F \to ab \end{cases}$$

۱) حذف λ :

$$\left.\begin{array}{l} B \to \lambda \\ C \to \lambda \\ A \to BC \end{array}\right] \Rightarrow A \to \lambda \Rightarrow$$ nullable مجموعه عناصر $= \{A, B, C\}$

$$\begin{cases} S \to ABaC / BaC / AaC / ABa / aC / Aa / Ba / a \\ A \to BC / C / B \\ C \to D \\ B \to b \end{cases}$$

۲) حذف Unit :

$A \to BC / C / B$

$A \to BC / d / b$

$C \to d$

$D \to d$

$F \to ab$

۳) حذف Useless :

حذف $\begin{cases} D \to d \\ F \to ab \end{cases}$

فرم های نرمال:

۱) چامسکی

۲) گریباخ

Subject: Date:

فرم نرمال چامسکی: قواعد به شکل زیر باشد:

$$\begin{cases} A \to BC & \text{فقط به دو متغیر} \\ A \to a & \text{یا فقط به یک ترمینال} \\ a \in T \\ A, B, C \in V \end{cases}$$

مثال: زبان گرامر داده شده را بنویسید. سپس آن را به فرم نرمال چامسکی درآورید.

$S \to AB$

$A \to aaA / aa$

$B \to aBb / ab$

چون دوتا a اضافه دارد.

$$L = \{ a^{2m} a^n b^n \mid m \geq 1 \}$$

$S \to AB$

$A \to X_a X_a A / X_a X_a$

$X_a \to a$

$Z \to X_a A$

$B \to X_a B X_b / X_a X_b$

$X \to B X_b$

$X_b \to b$

نکته ی 1: برای تمام زبان های مستقل از متن که شامل λ نباشد، می توان فرم نرمال چامسکی نوشت.

نکته ی 2: قبل از تبدیل به فرم نرمال چامسکی، ابتدا ساده سازی را انجام دهید. (حداقل مراحل 1 و 2 ساده سازی انجام شود)

نکته ی 3: در فرم نرمال چامسکی، رشته ای با طول K، در $2K-1$ مرحله تولید می شود.

Subject: .... Date: ....

2) فرم نرمال گریباخ: در فرم نرمال گریباخ، باید تمام قواعد گرامر مستقل از متن،

شرایط زیر را داشته باشد:

$$\begin{cases} A \rightarrow aX \\ a \in T \\ X \in V^* \end{cases}$$

حتماً با یک پایانه شروع شود و بقیه غیرپایانه باشند.

مثال: گرامر زیر را به فرم گریباخ بنویسید.

$$\begin{cases} S \rightarrow AB \\ A \rightarrow aaA / aa \\ B \rightarrow aBb / ab \end{cases}$$

$$S \rightarrow aaAB / aaB \Longrightarrow \begin{cases} S \rightarrow aX_a AB / aX_a B \\ X_a \rightarrow a \\ A \rightarrow aX_a A / aX_a \\ B \rightarrow aBX_b / aX_b \\ X_b \rightarrow b \end{cases}$$

Subject:

Date:

نکته‌ی ۱: اگر زبان مستقل از متن شامل λ نباشد، حتماً فرم نرمال گریباخ دارد.

نکته‌ی 2: در تبدیل به فرم نرمال گریباخ، اگر بازگشتی از چپ (با یا بدون واسطه) داشتیم، ابتدا بازگشتی را حذف می‌کنیم. مثل $A \to Aa$ بی‌واسطه یا $(A \to Ba, B \to Aa)$ با واسطه

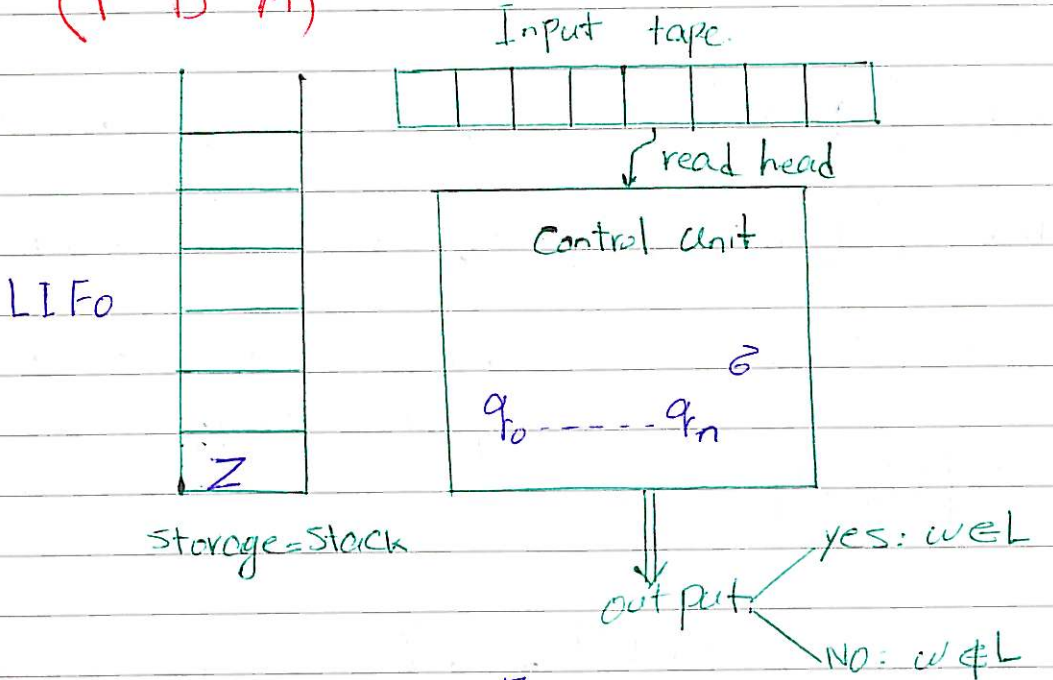
نکته‌ی 3: در فرم نرمال گریباخ، رشته‌ای به طول k در k مرحله تولید می‌شود.

Subject: .... Date: ۹۱/۹/۱۳

فصل هفتم:

ماشین های پشته ای:

Push Down Automata:
(P D A)



$M(Q, \Sigma, \delta, q, F, \Gamma, Z)$

$q, F \in Q$ $\quad$ $Z \in \Gamma$ $\quad$ Z آخر پشته را نشان می دهد

اگر Z نباشد، ماشین کار نمی کند.

* وقتی چیزی را از پشته POP می کنیم، از پشته برداشته می شود.

* ماشین های پشته ای Accepter هستند.

$$\delta : Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \longrightarrow \underbrace{\{(Q \times \Gamma^*)\}}_{\text{Finite subset of } (Q \times \Gamma^*)}$$

Subject: .... Date: ....

مثال: $\delta(q_0, \underset{\text{ورودی}}{a}, \underset{\text{پشته}}{c}) = \{(q_1, xc)\}$ از q_0 به q_1 می رود. c در جای خود در پشته قرار می گیرد. x به بالای پشته اضافه می شود پس طول پشته یک واحد اضافه می گردد.

$\delta(q_1, b, x) = \{(q_2, x)\}$ x در جای خود قرار می گیرد و طول پشته تغییر نمی کند

$\delta(q_1, b, x) = \{(q_2, B)\}$ B به جای x قرار می گیرد و طول پشته تغییر نمی کند

$\delta(q_2, b, B) = \{(q_3, \lambda)\}$ B از پشته برداشته می شود و λ یعنی هیچی قرار می گیرد پس طول پشته یک واحد کم می شود.

$\delta(q_3, \lambda, C) = \{(q_4, \lambda)\}$ یعنی با λ به حالت بعدی می رود. زمانی رخ می دهد که یا نخواهیم هیچ حرکت کنیم یا رشته به آخر رسیده باشد.

مثال: برای $L = \{a^n b^n \mid n \geq 0\}$ که یک زبان مستقل از متن است یک Push down Automata رسم کنید.

λ : $\delta(q_0, \lambda, z) = \{(q_f, z)\}$ آلوی هم جزء آن زبان است.

ab

a^2b^2 پس برای λ تابع جدا می نویسیم.

a^3b^3

الگوریتم کنترل تعداد a و b: همه ی a ها را در پشته قرار می دهیم به ازای هر b، یک a از پشته بر می داریم. اگر رشته به آخر رسید و b هم اضافه نیامد، رشته پذیرفته می شود.

aabb را در نظر می گیریم:

$\delta(q_0, a, z) = \{(q_0, az)\}$ $\quad a$

$\delta(q_0, a, a) = \{(q_0, aa)\}$ $\quad aa$

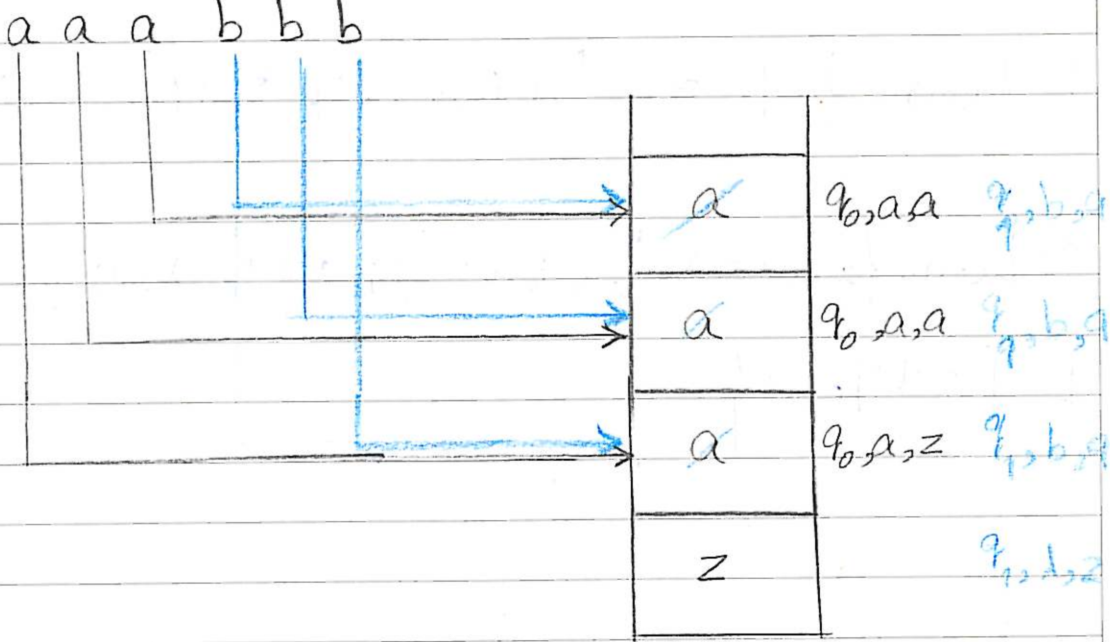
Subject: Date:

$\delta(q_0, b, a) = \{(q_1, \lambda)\}$ aab

$\delta(q_1, b, a) = \{(q_1, \lambda)\}$ aabb

$\delta(q_1, \lambda, Z) = \{(q_f, Z)\}$ رشته تمام شده و پشته به آخر رسیده

aaabbb را با توجه به قوانین بالا در نظر بگیریم:



تعریف پذیرش رشته توسط PDA:

1) رشته تمام شده و پشته خالی است.

2) رشته تمام شده و در حالت نهایی (q_f) هستیم و به پشته کاری نداریم.

$(q_r, \alpha, \beta) \overset{*}{\vdash} (q_f, \lambda, z)$

satisfy می‌کند صفر یا بیشتر مرحله

مثال:
$(q_0, abba, xcbaz) \vdash (q_1, bba, BCbaz)$

$\overset{*}{\vdash} (q_f, \lambda, z)$

با توجه به تعریف فوق شروط پذیرش رشته را می‌توان به صورت زیر نوشت:

پشته خالی رشته تمام

1) $L(m) = \{ w \mid w \in \Sigma^*, (q_0, w, z) \overset{*}{\vdash} (q', \lambda, z'), q' \in Q \}$

2) $L(m) = \{ w \mid w \in \Sigma^*, (q_0, w, z) \overset{*}{\vdash} (q_f, \lambda, u), u \in \Gamma^* \}$

مثال:
$L = \{ a^n b^n \mid n \geq 1 \}$

ab

a^2b^2

a^3b^3

$\vdots$

$\delta(q_0, a, z) = \{ (q_0, az) \}$

$\delta(q_0, a, a) = \{ (q_0, aa) \}$

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

$\delta(q_1, b, a) = \{ (q_1, \lambda) \}$

$\delta(q_1, \lambda, z) = \{ (q_f, z) \}$

Subject: Date:

$L = \{a^n b^{n+1} \mid n \geq 0\}$ مثال ۳:

b : $\delta(q_0, b, z) = \{(q_f, z)\}$ 1

ab^2

a^2b^3

a^3b^4

$\vdots$

الگوریتم کنترلی برای این که b همیشه یکی از a بیشتر باشد:

اولین بار به جای یک a، دو a به پشته اضافه می‌کنیم.

$\delta(q_0, a, z) = \{(q_0, aaz)\}$ 2

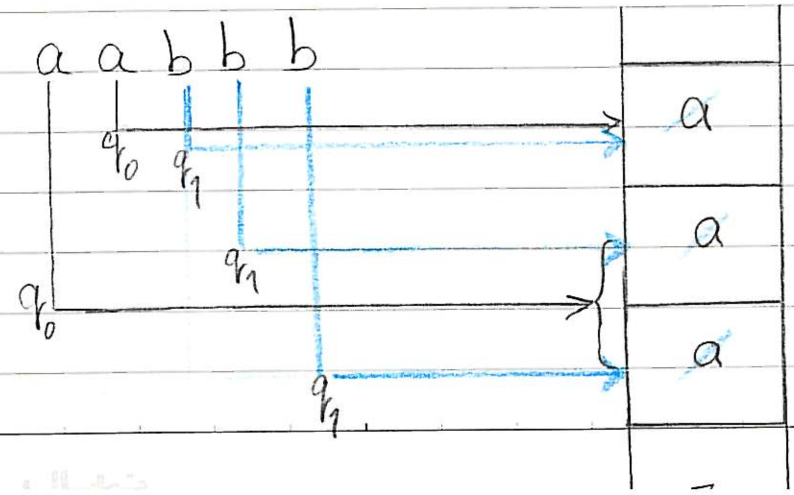
$\delta(q_0, a, a) = \{(q_0, aa)\}$ 3

$\delta(q_0, b, a) = \{(q_1, \lambda)\}$ 4

$\delta(q_1, b, a) = \{(q_1, \lambda)\}$ 5

$\delta(q_1, \lambda, z) = \{(q_f, z)\}$ 6

رشته‌ی aabbb را بررسی می‌کنیم:



اگر رشته aabbbb باشد در نهایت به (q_1, b, z) می‌رسد و چون قانون آن ایجاد نمی‌کند، متوقف می‌شود و رشته را نمی‌پذیرد.

Subject: .... Date: ....

$$L = \{ a^4 b^n c^n \mid n \geq 0 \}$$

۳ ن:

ته: جایی که باید تعداد متناهی کنترل شود، فقط حالت عوض می‌کنیم و به پشته کار نداریم.

$$a^4 \begin{cases} \delta(q_0, a, z) = \{(q_1, z)\} \\ \delta(q_1, a, z) = \{(q_2, z)\} \\ \delta(q_2, a, z) = \{(q_3, z)\} \\ \delta(q_3, a, z) = \{(q_4, z)\} \end{cases}$$

Subject: Date:

مثال: مهم $a^n c^4 b^n$: همه ی a ها را در پشته قرار می دهیم به ازای c ها حالت عوض می کنیم

مثال: $L = \{ww^R \mid w \in \{a,b\}^*\}$

λ : $\delta(q_0, \lambda, z) = \{(q_f, z)\}$

abba
baab
abbaabba

الگوریتم: تا مرز تقارن در پشته بگذارد، بعد از مرز، به ازای a، a و به ازای b، b از پشته بردارد.

$$\delta(q_0, a, z) = \{(q_0, az)\}$$

$$\delta(q_0, b, z) = \{(q_0, bz)\}$$

$$\delta(q_0, a, b) = \{(q_0, ab)\}$$

$$\delta(q_0, b, a) = \{(q_0, ba)\}$$

اگر مرز تقارن باشد. اگر مرز تقارن نباشد.

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

$$\delta(q_0, b, b) = \{(q_1, \lambda), (q_0, bb)\}$$

$$\delta(q_1, b, b) = \{(q_1, \lambda)\}$$

$$\delta(q_1, a, a) = \{(q_1, \lambda)\}$$

$$\delta(q_1, \lambda, z) = \{(q_f, z)\}$$

این ماشین، پشته ای غیرقطعی است.

شرایط قطعی بودن ماشین پشته‌ای :

1) تابع δ به ازای یک حالت با یک الفبا و با یک پشته حداکثر یک عضو داشته باشد.

حداکثر یک عضو ; $\forall q \in Q, \forall a \in \Sigma, \forall x \in \Gamma \quad \delta(q, a, x)$

2) اگر از یک q، بدون توجه به ورودی و با یک b داخل پشته راهی برای رفتن بود، با همان b و با توجه به یک ورودی راهی میسر نباشد.

$$\text{if } \big(\underbrace{\delta(q, \lambda, b) \text{ is not NULL}}\big) \Longrightarrow \underbrace{\delta(q, a, b) \text{ is NULL}}$$

با b و ورودی جای نرو ⟸ اگر با b به جای رفتن بدون ورودی

تعریف زبان مستقل از متن قطعی :

اگر برای زبان مستقل از متن بتوانیم ماشین پشته‌ای قطعی ارائه دهیم، زبان ماشین، مستقل از متن قطعی است.

نکته : ماشین پشته‌ای قطعی و غیرقطعی معادل هم نیستند.

$$\begin{tikzcd} \text{CFG} \arrow[rr, bend left, "\text{فرم نرمال گریباخ}"] & & \text{PDA} \arrow[ll, bend left] \end{tikzcd}$$

Subject: Date:

مثال: ماشین معادل گرامر زیر را رسم کنید.

$$\begin{cases} S \to aA & (1) \\ A \to aABC \ (2) \ / \ bB \ (3) \ / \ a \ (4) \\ B \to b \ (5) \\ C \to c \ (6) \end{cases}$$

نکته ی 1: همیشه برای تبدیل گرامر به ماشین پشته ای در گرامر ابتدا λ را اضافه می شود.

نکته ی 2: متغیرها در پشته قرار می گیرند. ($V \in \Gamma$)

رسم گراف:

آغازین: $\delta(q_0, \lambda, Z) = \{(q_0, SZ)\}$

(1): $\delta(q_0, a, S) = \{(q_0, A)\}$

(2): $\delta(q_0, a, A) = \{(q_0, ABC)\}$

(3): $\delta(q_0, b, A) = \{(q_0, B)\}$

(4): $\delta(q_0, a, A) = \{(q_0, \lambda)\}$

(5): $\delta(q_0, b, B) = \{(q_0, \lambda)\}$

(6): $\delta(q_0, c, C) = \{(q_0, \lambda)\}$

پایانی: $\delta(q_0, \lambda, Z) = \{(q_F, Z)\}$

نکته: اگر گرامری به فرم simple باشد، ماشین پشته ای معادل آن حتماً قطعی است.

ل 8: قضیه ی Pumping در مستقل از متن ها (اصل لانه کبوتری، تزریق):

فرض کنید زبان L مستقل از متن نامتناهی باشد، آنگاه عدد صحیح مثبت m وجود خواهد داشت به گونه ای که $|w| \geq m$، آنگاه می توان w را به صورت زیر تجزیه کرد:

$$w = u\underbrace{vxy}z$$

$$|vxy| \leq m$$

$$|vy| \geq 1$$

$$w_i = uv^i x y^i z \in L \qquad i = 0, 1, \ldots$$

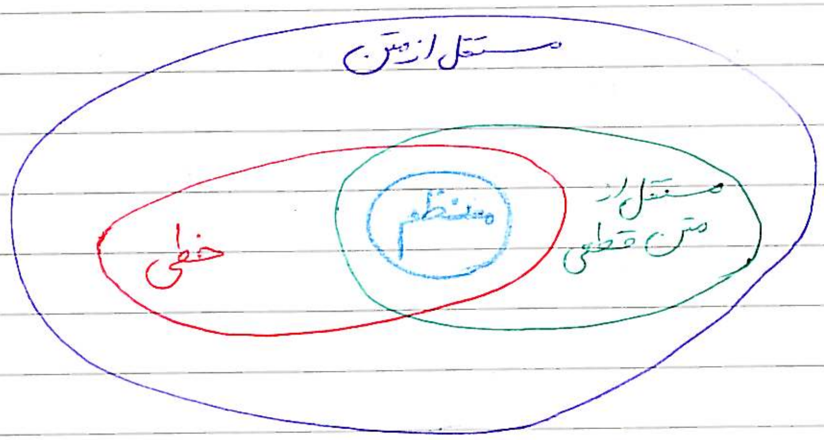
نکته: از قضیه ی بالا فقط می توان برای اثبات مستقل از متن نبودن استفاده کرد.

مثال: $L = \{a^n b^n \mid n \geq 0\}$ می دانیم مستقل از متن است ولی با Pumping نمی توانیم تناقضی در این رابطه بدهیم.

مثال: $L = \{a^n b^n c^n \mid n \geq 0\}$

حریف	شما
1) m	2) $w = a^m b^m c^m = a^{\frac{m}{2}} a^{\frac{m}{2}} b^{\frac{m}{2}} b^{\frac{m}{2}} c^m$
3) $u = a^{\frac{m}{2}}$ $v = a^{\frac{m}{2}-1}$ $x = a$ $y = b^{\frac{m}{2}}$ $z = b^{\frac{m}{2}} c^m$	4) $w = a^{\frac{m}{2}} (a^{\frac{m}{2}-1})^i a (b^{\frac{m}{2}})^i b^{\frac{m}{2}} c^m$ $i=0 \Rightarrow w_0 = a^{\frac{m}{2}+1} b^{\frac{m}{2}} c^m \notin L \Rightarrow$ زبان، مستقل از متن نیست.

قضیه 2 : Pumping در زبان های خطی :



فرض کنید که یک زبان مستقل از متن خطی نامتناهی باشد. آنگاه عدد صحیح و مثبت m وجود خواهد داشت به طوری که $\forall w \in L ; |w| \geq m$ و رشته ی w را می توان به صورت زیر تجزیه کرد:

$$w = uvxyz$$

$$|uvyz| \leq m$$

$$|vy| \geq 1$$

$$w_i = uv^ixy^iz \in L$$

نکته 1: الگوریتم تشخیص تهی بودن زبان مستقل از متن: اگر useless ها را حذف کردیم و S هم حذف شد، تهی است.

Subject: Date:

نکته ی 2: الگوریتم تشخیص نامتناهی بودن زبان مستقل از متن: اگر سه مرحله‌ی ساده سازی (λ، Unit، Useless) را انجام دادیم، ولی متغیر تکرار شونده (بازگشتی با یا بدون واسط) داشتیم، زبان مستقل از متن، نامتناهی است.

Subject: .... Date: ۹۱/۴/۲۷

فرم کلی گرامرهای خطی و اثبات Pumping خطی:

$$\begin{cases} S \rightarrow uAz \\ A \rightarrow vAy / x \\ u, z, v, y, x \in T \end{cases}$$

$$S \Rightarrow uAz \Rightarrow uvAyz \xRightarrow[\text{بار } i-1]{A \text{ را}} \underline{uv^{i-1} A y^{i-1} z} \Rightarrow \underbrace{uv^i x y^i z}_{\text{Pumping}}$$

قضیه: خانواده‌ی زبان‌های مستقل از متن تحت عملگرهای $\cup$ و $\cdot$ و $*$ بسته است.

L_1: مستقل از متن : S_1

L_2: مستقل از متن : S_2

$L_1 \cup L_2 \Rightarrow S \rightarrow S_1 / S_2$

$L_1 \cdot L_2 \Rightarrow S \rightarrow S_1 S_2$

نکته: برزدن یعنی الحاق های پشت سرهم مثل $L_1 \cdot L_2 \cdot L_1 \cdot L_2$ و اگر مجموعه‌ای تحت عمل الحاق بسته باشد، تحت برزدن نیز بسته است.

$L_1^* \Rightarrow S \rightarrow S_1 S / \lambda$

$L_1^+ \Rightarrow S \rightarrow S_1 S / S$

...ده‌ی زبان‌های مستقل از متن تحت عملگرهای $\cap$ و مکمل

...د.

$$\overline{(\overline{L_1} \cup \overline{L_2})} = L_1 \cap L_2$$

...مجموعه‌ای، اگر نسبت به اجتماع و مکمل بسته باشد، نسبت به اشتراک

...ل نیز بسته است. و بالعکس.

...ت : اگر مجموعه‌ای تحت اجتماع و اشتراک بسته باشد، ممکن است تحت

...مل بسته باشد یا نباشد.

...ثال: $L_1 = \{a^n b^m c^m \mid n, m \geq 0\}$ مستقل از متن

$L_2 = \{a^m b^m c^n \mid n, m \geq 0\}$ مستقل از متن

$L_1 \cap L_2 = \{a^n b^n c^n \mid n \geq 0\}$ حساس به متن

نکته : اگر زبان به storage نیاز نداشته باشد، منظم است.
اگر زبان با پشته قابل حل باشد، مستقل از متن است.

مسائل تصمیم پذیر در گرامرهای مستقل از متن :

تصمیم پذیر یعنی الگوریتم برای آن داریم.

Subject: Date:

قضیه ی اول: الگوریتمی وجود دارد که می تواند تصمیم بگیرد که آیا یک زبان مستقل از متن، تهی است یا خیر؟

1) حذف useless ها

- از S به آن ها راهی نیست
- یا
- از آن ها به پایان راهی نیست

2) اگر S حذف شد، زبان تهی است.

قضیه ی دوم: الگوریتمی وجود دارد که می تواند تصمیم بگیرد که آیا یک زبان یا گرامر مستقل از متن، نامتناهی است یا متناهی؟

1) حذف λ

2) حذف unit ها

3) حذف useless ها

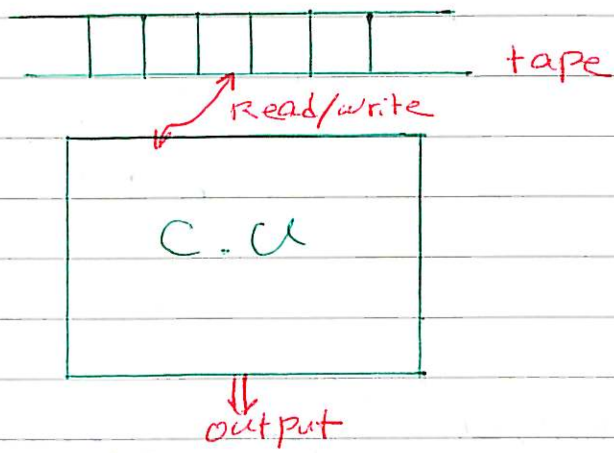
4) اگر متغیر تکرار شونده (بازگشتی) داشتیم، زبان نامتناهی است.

ماشین های تورینگ:

ماشین های تورینگ معادل گرامر بی قید و شرط هستند.

Subject: Date:

ماشین تورینگ استاندارد:



ویژگی ها:

1) هیچ فایل ورودی و خروجی ندارد.

2) تورینگ استاندارد یک نوار مجزا است که از دو سمت نامتناهی می باشد.

3) نوار هم شامل ورودی است و هم می توان در آن نوشت یعنی یک حافظه‌ی نامتناهی است.

4) تورینگ استاندارد، یک هد مجزا است که به منظور خواندن و نوشتن، قادر است به تعداد به سمت چپ یا راست حرکت کند.

ماشین:

الفبای نوار

در خانه های خالی نوار قرار می گیرد

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

blank یا B یا ⊁

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

به چپ یا راست

از یک حالت با خواندن الفبای نوار

به حالت دیگر با نوشتن روی ...

Subject: Date:

مثال:

a	b	a	b

پیکربندی

$\delta(q_0, a) = (q_1, x, R)$

از q_0 با خواندن a برو به q_1 و x را به جای a بگذار و به راست حرکت کن.

ماشین تورینگ قطعی: اگر برای هر پیکربندی (سمت راست تابع گذار) حداکثر یک حرکت مجاز باشد، ماشین تورینگ قطعی است.

مثال: ماشین تورینگی طراحی کنید که در رشته های متشکل از a و b، همه ی a ها را با b جایگزین کند.

$w \in \{a,b\}^*$ $\quad$ w: ab

bbab

aabbb

$\delta(q_0, a) = (q_0, b, R)$

$\delta(q_0, b) = (q_0, b, R)$

تغییر حالت نمی دهیم تا برای a و b های بعدی از همین قانون استفاده کنیم.

$\delta(q_0, \square) = (q_f, \square, L)$ → هد به عقب برگردد تا روی آخر رشته باشد.

نکته: همیشه در پایان کار، هد باید روی پایان رشته قرار بگیرد.

Subject: Date: پایان رشته

$L(TM) = \{w \mid w \in \Sigma^+ \mid q_0 w \vdash^* x_1 q_f x_2 , x_1, x_2 \in \ldots , q_f \in \ldots$

نهایت، هد ابتدای رشته قرار می‌گیرد. چون اگر رشته جزء زبان نباشد، ماشین تا ابد به دنبال

λ یعنی blank می‌گردد و داخل loop می‌افتد.

مثال: ماشین تورینگ زیر چه زبانی را می‌پذیرد؟

$\delta(q_0, a) = (q_1, b, R)$

$\delta(q_0, b) = (q_1, a, R)$

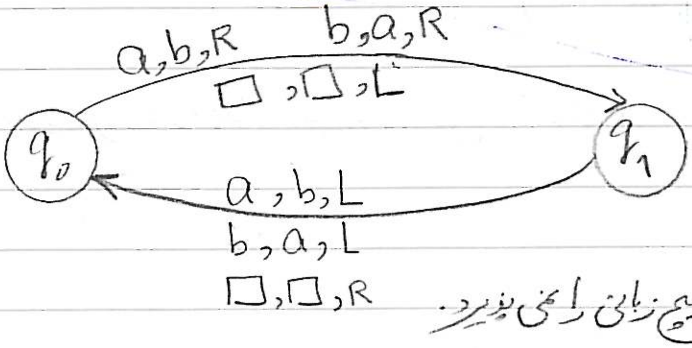
$\delta(q_1, a) = (q_0, b, L)$

$\delta(q_1, b) = (q_0, a, L)$

$\delta(q_1, \square) = (q_0, \square, R)$

$\delta(q_0, \square) = (q_1, \square, L)$

ابتدا گراف را رسم می‌کنیم:



در حلقه می‌افتد یعنی هیچ زبانی را نمی‌پذیرد.

مثال: ماشین تورینگی طراحی کنید که تمام الفبای موجود در رشته‌های متشکل از a، b و c را به x تبدیل کند.

$\delta(q_0, a) = (q_0, x, R)$

$\delta(q_0, b) = (q_0, x, R)$

$\delta(q_0, c) = (q_0, x, R)$

$\delta(q_0, \square) = (q_F, \square, L)$

تمرین: ماشین تورینگی طراحی کنید که اگر در رشته، زیر رشته‌ی ab وجود داشت، پیغام yes و در غیر این صورت No بدهد. (جستجوی $w=ab$ در تمام رشته‌های $\{a,b\}$)

NO q_{F_0} yes q_{F_1}

راهنمایی:
- state تغییر $\Leftarrow$ a
- yes $\Leftarrow$ b

اگر پس از a، a بود دوباره و اگر فقط b دیدیم تغییر حالت نداریم.

$\delta(q_0, a) = (q_1, a, R)$

$\delta(q_1, b) = (q_{F_1}, b, R)$

$\delta(q_1, a) = (q_1, a, R)$

$\delta(q_1, \square) = (q_{F_0}, \square, L)$

$\delta(q_{F_1}, b) = (q_{F_1}, b, R)$

$\delta(q_{F_1}, a) = (q_{F_1}, a, R)$

$\delta(q_{F_1}, \square) = (q_{F_1}, \square, L)$

$\delta(q_0, b) = (q_0, b, R)$

STAEDTLER

مثال:

$L=\{a^n b^n \mid n \geq 1\}$

ab

aabb

aaabbb

هر a که خواندیم تبدیل x و هر b بازای آن که خواندیم، y به جای b می‌گذاریم.

$\delta(q_0, a) = (q_1, x, R)$

$\delta(q_1, b) = (q_2, y, L)$

$\delta(q_1, a) = (q_1, a, R)$

$\delta(q_1, y) = (q_1, y, R)$

$\delta(q_2, a) = (q_2, a, L)$

$\delta(q_2, x) = (q_0, x, R)$

$\delta(q_2, y) = (q_2, y, L)$

$\delta(q_0, y) = (q_3, y, R)$ — همه a ها به x تبدیل شده

$\delta(q_3, y) = (q_3, y, R)$

$\delta(q_3, \square) = (q_f, \square, L)$

STAEDTLER

تمرین: برای زبان L ماشین تورینگ طراحی کنید $L = \{a^n b^n c^n \mid n \geq 1\}$

$\delta(q_0, a) = (q_1, x, R)$

$\delta(q_1, a) = (q_1, a, R)$

$\delta(q_1, b) = (q_2, y, R)$

$\delta(q_2, b) = (q_2, b, R)$

$\delta(q_2, c) = (q_3, z, L)$

$\delta(q_1, y) = (q_1, y, R)$

$\delta(q_2, z) = (q_2, z, R)$

$\delta(q_3, a) = (q_3, a, L)$

$\delta(q_3, b) = (q_3, b, L)$

$\delta(q_3, y) = (q_3, y, L)$

$\delta(q_3, z) = (q_3, z, L)$

$\delta(q_3, x) = (q_0, x, R)$

$\delta(q_0, y) = \delta(q_4, y, R)$

$\delta(q_4, z) = (q_4, z, R)$

$\delta(q_4, \square) = (q_F, \square, L)$

قضیه: هر مسئله‌ای که توسط قلم قابل حل باشد یا برای آن الگوریتم وجود داشته باشد، یا به صورت مکانیکی قابل حل باشد، حتماً برای آن ماشین تورینگ وجود دارد.

قضیه: تابع f روی دامنه D قابل حل توسط ماشین تورینگ است. اگر:

$$\forall w \in D: \; q_0 w \overset{*}{\longmapsto} q_f f(w)$$

$$q_f \in F$$

$$q_0 \in Q$$

مثال: تابع ADD :

1	1	0	1	1	1

$$\underbrace{11}_{x}\,0\,\underbrace{111}_{y}$$

$$ADD \begin{cases} |x| \geq 1 , \; |y| \geq 1 \\ x + y \\ x , y \in \{1\}^* \end{cases}$$

قرار می‌دهیم لذا بین دو عدد یک صفر به عنوان جداکننده قرار دهیم.

1	1	1	1	1

$\delta(q_0 , 1) = (q_1 , 1 , R)$

$\delta(q_1 , 1) = (q_1 , 1 , R)$

$\delta(q_1 , 0) = (q_2 , 1 , R)$

$\delta(q_2 , 1) = (q_3 , 1 , R)$

$\delta(q_3 , 1) = (q_3 , 1 , R)$

$\delta(q_3 , \square) = (q_4 , \square , L)$

$\delta(q_4 , 1) = (q_f , \square , L)$

تمرین: برای تابع ضرب، مانند مثال قبل کپی کنید و مقایسه کنید.

1	1	∅	1	1	1

$|x| \geq 1$ $\qquad$ $|y| \geq 1$

نتیجه باید ۱۱۱۱۱۱ باشد.

۱۱۱ باید دوبار تکرار شود.

اعداد را می خوانیم تا به □ برسیم به جای آن = می گذاریم. دوباره به عقب برمی گردیم به ازای ۱ اول، یک بار عدد دوم را بعد از = می نویسیم و ۱ را x می کنیم.

$\delta(q_0, 1) = (q_0, 1, R)$

$\delta(q_0, \emptyset) = (q_0, \emptyset, R)$

$\delta(q_0, \square) = (q_1, =, L)$

$\delta(q_1, 1) = (q_1, 1, L)$

$\delta(q_1, \emptyset) = (q_2, \emptyset, L)$

$\delta(q_2, 1) = (q_3, x, R)$

$\delta(q_3, \emptyset) = (q_3, \emptyset, R)$

$\delta(q_3, 1) = (q_4, y, R)$

$\delta(q_4, 1) = (q_4, 1, R)$

$\delta(q_4, =) = (q_5, 1, L)$

$\delta(q_5, =) = (q_6, =, L)$

$\delta(q_6, 1) = (q_6, 1, L)$

$\delta(q_6, y) = (q_3, y, R)$

$\delta(q_3, =) = (q_1, =, L)$

$\delta(q_1, y) = (q_1, y, L)$

$\delta(q_1, x) = (q_1, x, L)$

$\delta(q_1, \square) = (q_1, \square, R)$

مقایسه کننده:

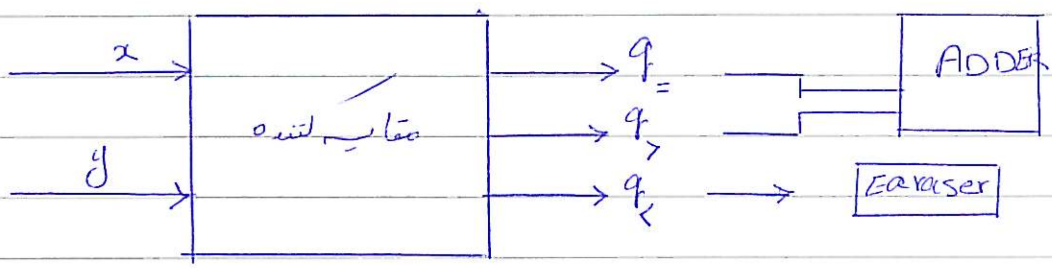
$$x \oslash y \begin{cases} x=y & q_= \\ x>y & q_> \\ x<y & q_< \end{cases}$$

پس سه q پایانی هستند.

طریقه ی تشخیص تساوی $a^n b^n$ است. اگر a اضافه آوردیم: x بزرگتر، اگر b اضافه آوردیم: y بزرگ تر و در غیر این صورت: $x=y$

مثال:

$$F(x,y)=\begin{cases} x+y & x>y \\ \emptyset & x<y \end{cases}$$



فصل ۱۵:

تعریف: دو ماشین تورینگ، در صورتی باهم برابر، معادل یا هم ارزند که هر دو زبان یکسانی را بپذیرند. فرض کنید C_1 و C_2 دو کلاس مختلف از ماشین های تورینگند:

if $\forall\, m_1 \in C_1\ \exists\, m_2 \in C_2,\ L(m_1) = L(m_2) \Rightarrow$ C_2 حداقل به قدرت C_1

and

if $\forall\, m_2 \in C_2\ \exists\, m_1 \in C_1,\ L(m_1) = L(m_2) \Rightarrow$ C_1 به قدرت C_2

پس C_1 و C_2 هم ارزند.

نکته: کلاس ماشین های DFA و NDFA باهم، هم ارزند.

نکته: ماشین پشته ای غیر قطعی حداقل به قدرت ماشین های پشته ای قطعی هستند، ولی معادل نیستند.

۹۱/۱۰/۴

انواع ماشین های تورینگ :

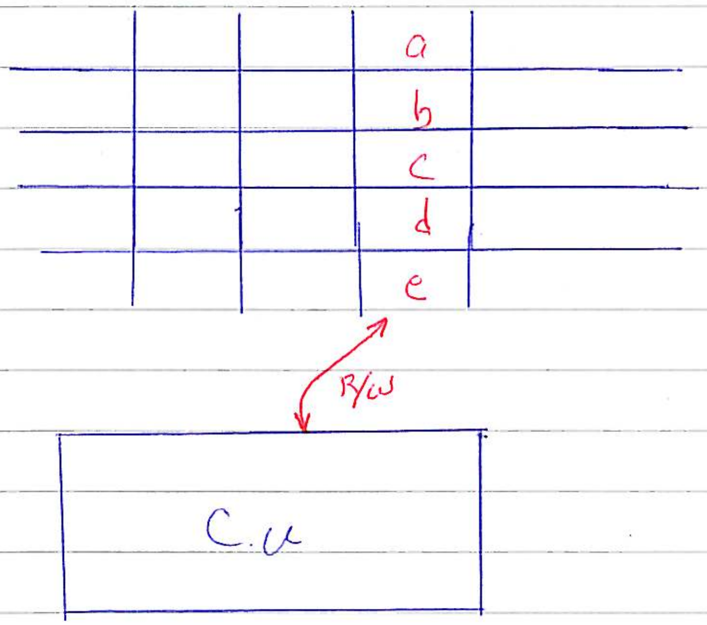
1) Turing Machine with an stay option:

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

$\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R, S\}$ با توقف

قدرت زیاد نشده چون در استاندارد هم با یک حرکت به چپ و یک حرکت به راست، کار توقف انجام می شود

2) Multiple track TM : (چند شیاره)



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

این تورینگ با تورینگ استاندارد هم ارز است. چون قابل تبدیل است به

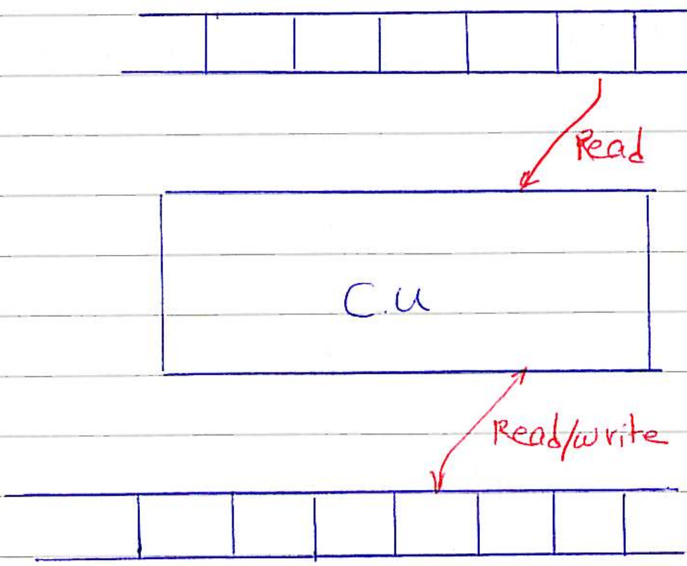
Subject: Date:

تورینگ استاندارد با یکی از روش های زیر :

1) هر شیار عمودی را به صورت افقی بنویسیم در یک نوار

2) هر ترتیب را کد کنیم (مثلاً $x_1 = abcde$ یا $x_2 = aabcd$)

3) Offline Turing:

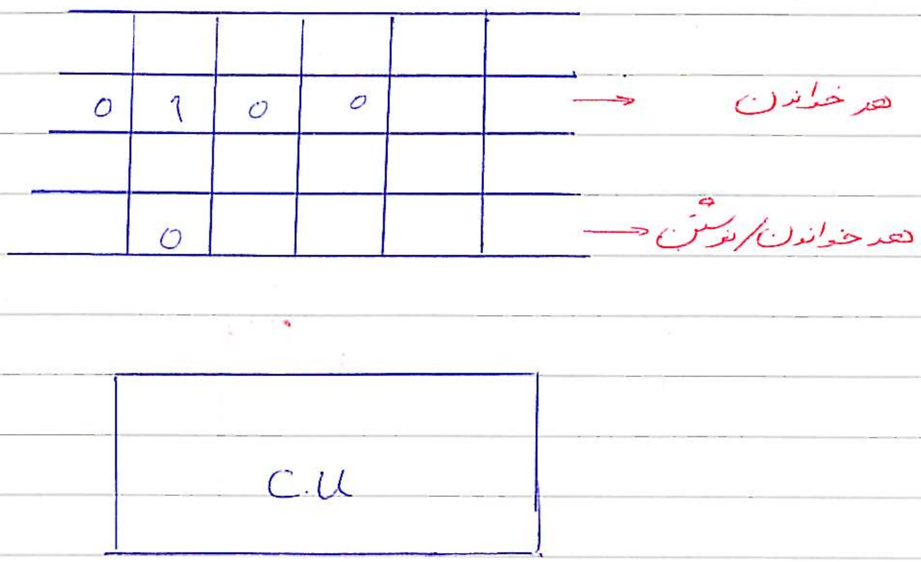


$$\delta : Q \times \Gamma_1 \times \Gamma_2 \longrightarrow Q \times \Gamma_2 \times \{L, R\}$$

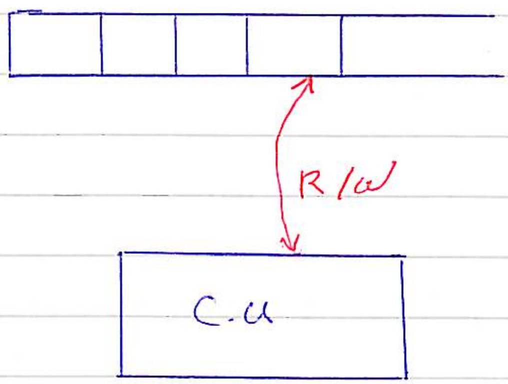
این تورینگ قابل تبدیل به تورینگ چند شیاره است به صورت زیر :

یک تورینگ ۴ شیاره تعریف می کنیم که شیار اول برای نوار offline ، شیار

دوم برای شبیه سازی هد فقط خواندنی، نوار سوم برای storage اصلی و نوار چهارم برای شبیه سازی هد خواندنی و نوشتنی است. پس تورینگ OFF-line هم با تورینگ استاندارد، هم ارز است.

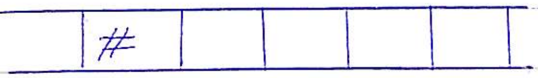


4) Turing Machine with semi-infinite tape:



Subject: Date:

این تورینگ مانند تورینگ استانداردی است که ابتدای نوار با یک حرف محدود شده باشد.



$\delta(q', \#) = (q'', \#, R)$

$\delta: Q \times \Gamma \longrightarrow Q \times \Gamma \times \{R, L\} + \delta(q', \#) = (q'', \#, R)$

5) Multiple tape TM: تورینگ چندنواره:

به تعداد نوار هم داریم:

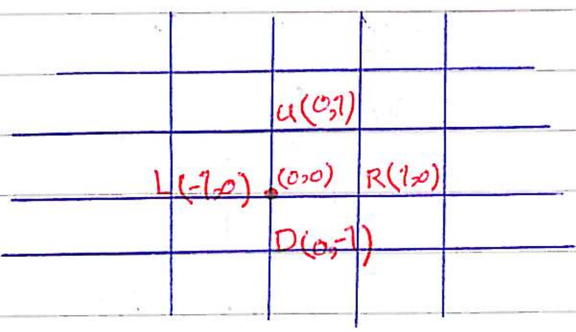
$Q \times \Gamma^n \longrightarrow Q \times \Gamma^n \times \{L, R\}^n$

این ماشین قابل تبدیل به تورینگ چندشیاره با 2n شیار است، پس

معادل استاندارد است.

6) Multi dimentional TM: تورینگ چندبعدی

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



Subject: .... Date: ....

با استفاده از مختصات آ دو شیاره تبدیل می کنیم :

یک شیار برای مختصات و یک شیار برای محتویات در نظر می گیریم و بین

مختصات از جداکننده و بین محتویات از کاراکتر خنثی استفاده می کنیم .

(مانند x یعنی وقتی به آدرس فقط جلو برود)

#	0	,	0	#	1	,	0	#
a	x	x	x	b	x	x	x	

7) Nondeterministic TM : تورینگ غیر قطعی :

$$Q\times\Gamma \longrightarrow 2^{Q\times\Gamma\times\{L,R\}}$$

این ماشین تورینگ قابل تبدیل به تورینگ چندشیاره است ، بنابراین

با تورینگ استاندارد هم ارز است .

مثال :

(q_0, a, L)
(q_1, b, R)

a	#	#	
#	q_0		#
#	q_1	b	
		¦	

اطراف تابع گذار را با یک علامت ، محصور می کنیم .

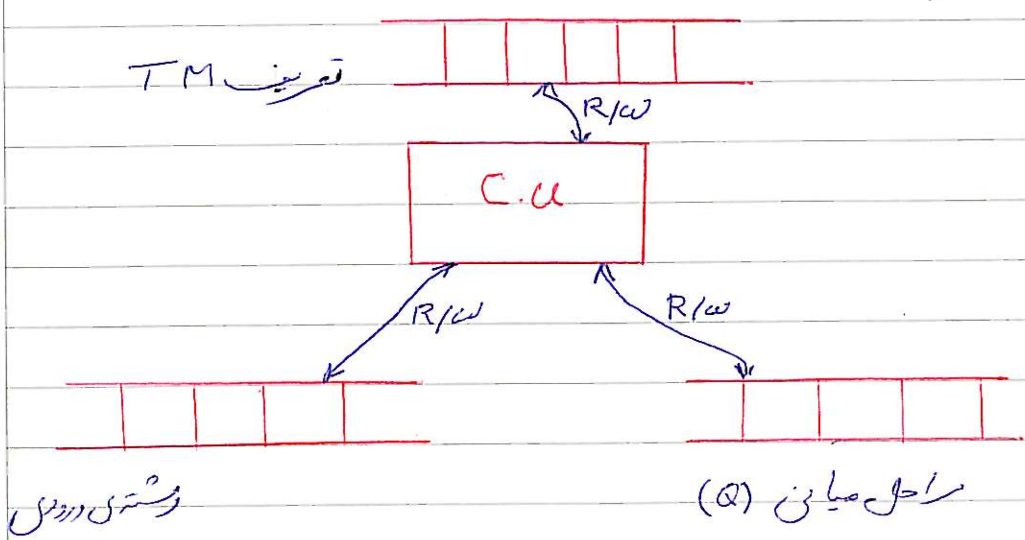
Subject: Date:

قضیه: کلاس ماشین های تورینگ قطعی و کلاس ماشین های تورینگ غیر قطعی با یکدیگر معادلند.

قضیه: ماشین تورینگ استاندارد، بیشترین قدرت را دارد؛ چون معادل الگوریتم است و با افزایش قابلیت ها، قدرت تغییر نمی کند.

ماشین تورینگ جهانی: Universal TM

این تورینگ هر ماشین تورینگی را حل می کند. این تورینگ مانند یک کامپایلر عمل می کند (کامپایلری که مثلاً کامپایلر C، زبان C و ورودی به آن می دهیم و خروجی می گیریم)



مجموعه های شمارا و ناشمارا:

به مجموعه‌ای شمارا گفته می‌شود که بتوان بین اعضای آن مجموعه و اعداد طبیعی یک تناظر یک به یک برقرار کرد. یعنی بتوان آن ها را شمرد.

مثال: مجموعه‌ی اعداد صحیح شمارا است.

الگوریتم شمارش: $0, 1, -1, 2, -2, 3, -3, 4, -4, \dots$

مثال: مجموعه‌ی اعداد گویا شمارا است.

$$\begin{array}{cccc}
\frac{1}{1} & \frac{1}{2} & \frac{1}{3} & \dots \\
\frac{2}{1} & \frac{2}{2} & \frac{2}{3} & \dots \\
\frac{3}{1} & \frac{3}{2} & \frac{3}{3} & \dots \\
\frac{4}{1} & & &
\end{array}$$

الگوریتم شمارش: روش قطری (اگر سطری یا ستونی بشماریم، انتها ندارد و معلوم نیست چه زمانی نوبت شمارش سطر بعدی یا ستون بعدی می‌شود)

Subject: Date:

مثال: مجموعه‌ی اعداد حقیقی ناشمارا است.

مجموعه‌ی ماشین های تورینگ:

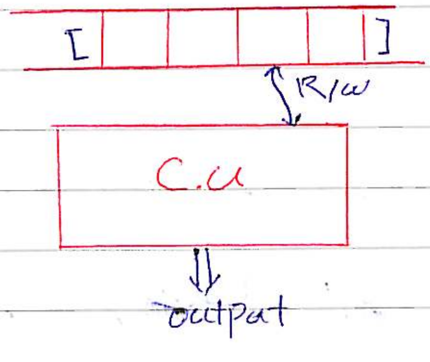
مجموعه‌ی ماشین های تورینگ $\equiv$ همه‌ی الگوریتم ها

تمام TM ها به کدهای متشکل از 0 و 1 تبدیل می شوند: ... ,11 ,10 ,01 ,00 ,1 ,0

پس شمارا است.

ماشین های کران دار خطی:

این ماشین ها با زبان های حساس به متن معادلند.



$$\delta: Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\} \quad + \delta(q',]) = (q'',], L)$$

$$+ \delta(q', [) = (q'', [, R)$$

شرط، یافتن $P(|w|)$ است که خطی باشد.

Subject: Date:

مثال:

$L = \{ a^{n!} \mid n \geq 1 \}$

می خواهیم a^9 را بررسی کنیم:

$$\begin{array}{r|l} 9 & 2 \\ \underline{8} & 4 \\ 1 & \end{array}$$

$\Rightarrow$ 9 فاکتوریل نیست

می خواهیم a^{24} را بررسی کنیم:

$$\begin{array}{r|l} 24 & 2 \\ \underline{24} & 12 \mid 3 \\ 0 & \underline{12} \quad 4 \mid 4 \\ & 0 \quad \underline{4} \quad 1 \\ & \quad 0 \end{array}$$

$\Rightarrow$ 24 فاکتوریل است.
$(2 * 3 * 4)$

$$F(|w|) = |w| + \frac{|w|}{2} + \frac{|w|}{2}$$

- $|w|$: میزان حافظه برای w
- $\frac{|w|}{2}$: میزان حافظه برای مقسوم علیه (حداکثر نصف $|w|$)
- $\frac{|w|}{2}$: میزان حافظه برای خارج قسمت (حداکثر نصف $|w|$)

برای تشخیص حساس به متن بودن زبان باید بتوانیم $F(|w|)$ خطی برای آن بیابیم. برای یافتن $F(|w|)$، میزان حافظه ی مورد نیاز برای محاسبات را براساس طول w بدست می آوریم. چون تابع خطی باید تابعی باشد برای تمام رشته های آن زبان. اگر $F(|w|)$ خطی یافتیم،

زبان حساس به متن و ماشین آن کران دار خطی است.

فصل ۹۶ :

سلسله مراتب ماشین ها و زبان های رسمی:

نتیجه گیری

Accept : if $w \in L \Rightarrow$ yes

در صورت عدم پذیرش، خروجی تولید نمی کند.

Decide : if $w \in L \Rightarrow$ yes

else No

تعریف: به زبان L شمارش پذیر بازگشتی (فهرست پذیر) یا RE گفته می شود اگر ماشین تورینگی وجود داشته باشد که این زبان را بپذیرد. (Accept)

و به زبان L روی Σ بازگشتی یا REC گفته می شود اگر ماشین تورینگی وجود داشته باشد که زبان L را بپذیرد. (Accept) و روی تمام رشته های $w \in \Sigma^*$ بتواند Halt کند یعنی متوقف شود. (Decide)

به عبارت دیگر زمانی بازگشتی است که اگر و تنها اگر برای آن یک الگوریتم عضویت وجود داشته باشد.

Subject: Date:

قضیه: فرض کنید S یک مجموعه‌ی شمارای نامتناهی باشد آنگاه مجموعه‌ی توانی S ناشمارا است.

(مهم)
قضیه: به ازای هر مجموعه‌ی ناتهی Σ زبان هایی وجود دارند که حتی RE هم نباشند. (زبان هایی که الگوریتم ندارند)

$$\underbrace{\Sigma}_{\text{شمارا متناهی}} \longrightarrow \underbrace{\Sigma^*}_{\text{شمارا نامتناهی}} \longrightarrow \underbrace{2^{\Sigma^*}}_{\text{ناشمارا، کل زبان ها}} \Longrightarrow$$ بعضی از رشته ها

الگوریتم ندارند ⟸ RE هم نیستند.

قضیه: می‌توان یک زبان RE پیدا کرد که مکمل آن RE نباشد.

یعنی خانواده‌ی زبان های RE تحت عمل مکمل بسته نیستند.

(اگر مکمل RE، RE باشد ⟸ REC است)

قضیه: RE هایی داریم که REC نیستند.

قضیه: اگر زبان L و مکمل آن هردو RE باشند، آنگاه هردو REC هم هستند. اگر زبان L، REC باشد، مکمل آن هم

Subject: Date:

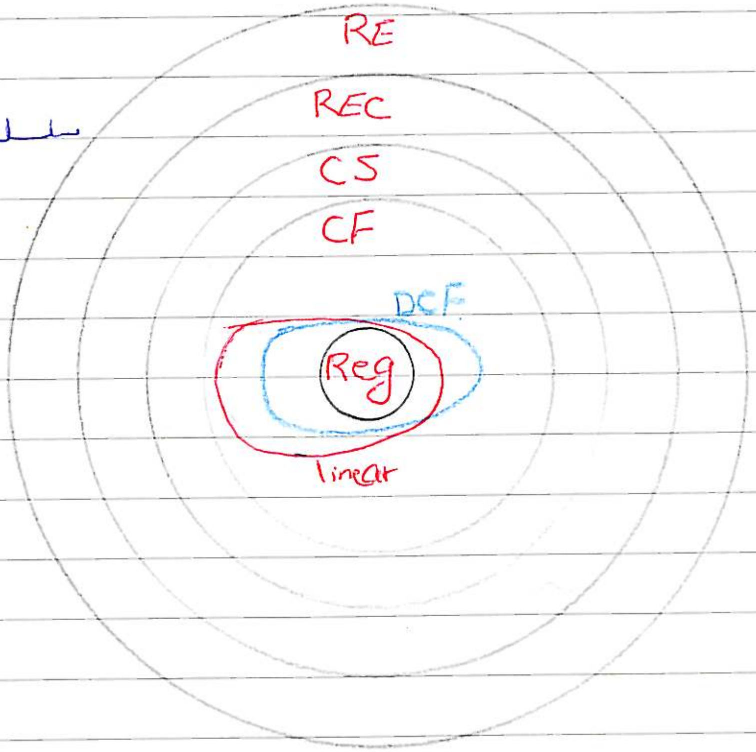
REC است.

قضیه: یک زبان RE وجود دارد که REC نیست. یعنی خانواده‌ی زبان های REC، زیرمجموعه‌ی محض خانواده‌ی زبان های RE است.

$L(REC) \subset L(RE)$

خانواده خانواده

سلسله مراتب چامسکی



نکته: هر زبان حساس به متن، REC یا بازگشتی است.

نکته: یک زبان بازگشتی وجود دارد که حساس به متن نباشد.

نکته: اگر یک زبان L توسط یک LBA پذیرفته شود، برای آن

Subject: Date:

یک گرامر حساس به متن وجود دارد.

نکته: خانواده‌ی زبان‌های RE تحت اجتماع و اشتراک بسته است.

نکته: هر زبان تولید شده توسط یک گرامر بی قید و شرط، RE است.
یا (هر زبانی که توسط یک گرامر نامحدود تولید شود، بازگشتی شمارش‌پذیر است و به ازای هر زبان RE یک گرامر نامحدود وجود دارد)