Subject:
Year. Month. Date. ( )

«تئوری زبان‌ها و ماشین‌ها»

الفبا: الفبا مجموعه‌ای کمرد از علامات است.

رشته: رشته دنباله‌ای از علامات است.

رشته تهی: λ رشته‌ای به طول صفر رشته تهی نامیده می‌شود.

عملگرهای رشته‌ای

concatenation: $x = 001$, $y = 111$ $\rightsquigarrow$ $xy = 001111$

$\lambda x = x\lambda = x$

$|xy| = |x| + |y|$

Reversal: $x = 001$ $\rightsquigarrow$ $x^R = 100$

$(x^R)^R = x$

$(xy)^R = y^R x^R$

Subject:

Year. Month. Date. ( )

رشته Palindrom:

$x = x^R$

$x = x^R \Rightarrow (xx)^R = xx$

زبان: زبان مجموعه‌ای محدود یا نامحدود از رشته‌هاست.

ستاره یک زبان Star closure:

$$L^* = \bigcup_{i=0}^{\infty} L^i = L^0 \cup L^1 \cup L^2 \cup \dots$$

هر عضوی که از داخل L^* انتخاب کنیم می‌توان آن را به تطاقی شامل L تقسیم کرد.

زبان positive closure:

$$L^+ = \bigcup_{i=1}^{\infty} L^i = L^1 \cup L^2 \cup L^3 \cup \dots$$

! زبان $\{\lambda\}$ با زبان $\emptyset$ متفاوت است.

0

Subject:

Year. Month. Date. ( )

عملگر سر

$$HEAD(L) = \{ x \mid xy \in L \text{ for some } y \in \Sigma^* \}$$

عملگر سر به صورت مجموعه روی زبان تعریف می‌شود
که L زبانی است که روی Σ تعریف شده است.

به بیانی دیگر تمام prefix های Σ^* می‌سازند. اگر رشته‌ای در HEAD را انتخاب کنیم می‌توانیم یک y در Σ^* پیدا کنیم که اگر به دنبال آن بیاید، رشته حاصل در L باشد.

$$L \subset HEAD(L)$$

مثال:

$$L = \{011, 10, 11, 0\}$$

$$HEAD(L) = \begin{Bmatrix} \lambda, & \cancel{x}, & \cancel{\lambda}, & \cancel{\lambda}, \\ 0, & 1, & \cancel{x}, & 0, \\ 01, & 10, & 11, & \\ 011, & & & \end{Bmatrix} \xrightarrow{\text{حذف عناصر تکراری}} \{\lambda, 0, 1, 01, 10, 11, 011\}$$

عملگر دم

$$TAIL(L) = \{ x \mid yx \in L \text{ for some } y \in \Sigma^* \}$$

عملگر دم نیز به صورت مجموعه روی زبان تعریف شد
که L زبان تعریف شده روی Σ است.

به بیانی دیگر تمام suffix های Σ^* می‌سازند.

Subject:

Year. Month. Date. ( )

مثال:

$L = \{011, 10, 11, 0\}$

$TAIL(L) = \begin{cases} \lambda, & \cancel{\lambda}, & \cancel{\lambda}, & \cancel{\lambda}, \\ 1, & 0, & \cancel{1}, & \cancel{0} \\ 11, & & & \\ 011, & 10, & \cancel{11}, & \end{cases}$ حذف تکراری‌ها $\rightarrow \{\lambda, 0, 1, 10, 11, 011\}$

تمرین: یک مانند زبان مانند L پیدا کنید بطوریکه:

1. $HEAD(TAIL(L)) = HEAD(L)$ L چه خصوصیاتی باید داشته باشد تا رابطه روبرو برقرار باشد؟
2. $TAIL(HEAD(L)) = HEAD(L)$ ؟ " " " " " " " " " " "
3. $TAIL(HEAD(L)) = HEAD(TAIL(L))$ ؟ " " " " " " " " " " "

گرامر Grammer

set of None-Terminal variables

مجموعه متغیرهای (حروف بزرگ)

گرامر بصورت یک چهارتایی تعریف می‌شود.

$G(V, \Sigma, P, S)$

starting symbol متغیر آغازین

set of Terminals

مجموعه نمادهای (الفبا)

set of production

set of Rules

set of rewriting rules

مجموعه قواعد

Subject:

Year. Month. Date. ( )

* قاعده:

$\alpha \to \beta$ ⟹ α می‌تواند جایش را به β بدهد / β می‌تواند جایگزین α شود / α به β می‌رود

P مجموعه‌ای از قواعد است.

* اشتقاق: Derivation

"in one step"

$\Rightarrow$ ⟹ اشتقاق در یک مرحله که می‌توان از یک فرم جمله‌ای به فرم جمله‌ای دیگر رسید.

$\overset{*}{\Rightarrow}$ ⟹ اشتقاق در صفر مرحله یا بیشتر. (در صورتی که مراحل طی شده اهمیت ندارد و جواب نهایی)

$S \Rightarrow AB$ ⟹ Sentutianal form "فرم جمله‌ای"

مثال:

$G = (V, T, P, S)$, $V = \{S, A, B\}$, $T = \{a, b\}$, $S = S$

$P = \{S \to AB,\ A \to aA,\ A \to a,\ B \to bB,\ B \to b\}$

می‌خواهیم از روی گرامر تولید زبان را بدست آوریم.

برای این منظور از علامت اولیه (Starting symbol) شروع می‌کنیم و با استفاده از قواعد جایگزینی P پیش می‌رویم

$S \Rightarrow AB$ ($S \to AB$)

$\Rightarrow aAB$ ($A \to aA$)

$\Rightarrow aaAB$ ($A \to aA$)

$\Rightarrow aaaB$ ($A \to a$)

$\Rightarrow aaabB$ ($B \to bB$)

$\Rightarrow aaabb$ ($B \to b$)

✓ هر رشته‌ای که از طریق حاصل می‌شود، عضوی است از زبانی که این گرامر آن زبان را تعریف شده است.

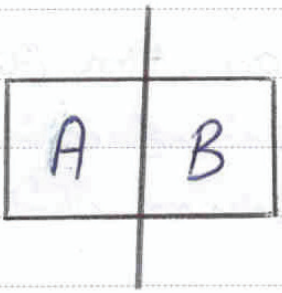
$S \overset{*}{\Rightarrow} aaabb$

Subject :

Year . Month . Date . ( )

اکنون می خواهیم ببینیم رشته های این زبان چه هستند و چه خصوصیاتی دارند؟

در واقع می خواهیم با استفاده از آن خصوصیات مشترک، فرم دیگری را برای زبان ارائه دهیم.



در مورد مثال قبل رشته های زبان را در دو دسته تشکیل شده بودند. گروه A و گروه B.

رشته های گروه A، در واقع رشته هایی هستند که اگر از A شروع کنیم به آنهایی میرسیم

رشته های گروه A بصورت بازگشتی تعریف شده اند.

رشته های گروه B نیز مشابه گروه A هستند و بصورت بازگشتی تعریف شده اند.

$A \to aA$

$A \to a$ ⟶ «Direct Recursion» (A مستقیماً برحسب خودش تعریف شده است)

اکنون فرم رشته های زبان را مشخص می نماییم.

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

❗ «یک گرامر به تنهایی یک زبان را مشخص می کند و بدون نیاز به چیز دیگری زبان را کاملاً مشخص می کند»

مثال :

$G = (V, T, P, S)$ $\quad V = \{S, A, B, C\}$, $T = \{a, b\}$

$S \in S$

$P = \{ S \to AB \ , \ A \to aC \ , \ C \to aA \ , \ A \to a \ , \ B \to bB \ , \ B \to b \}$

Subject:
Year. Month. Date. ( )

در اینجا هم رشته‌های زبان از دو قسمت تشکیل شده اند، گونه A و گونه B. (bها به دنبال aها می‌آیند)

گونه A شامل رشته‌هایی با تعداد فرد a می‌باشد.

$A \Rightarrow a$

$A \Rightarrow aC$
$\Rightarrow aaA$
$\Rightarrow aaa$

A از طریق یک واسطه مانند C موجب خودش تعریف شده است.

$A \rightarrow aC$
$C \rightarrow aA$
$A \rightarrow a$

→ «Indirect Recursion»

گونه B نیز که به مانند سابق است.

$L = \{ a^{2k+1} b^m \mid k \geq 0, m \geq 1 \}$

مثال: $G = (V, T, P, S)$, $V = \{S, A, B, C\}$, $T = \{a, b\}$, $S = S$

$P = \{ S \rightarrow AB, A \rightarrow aC, C \rightarrow aA, C \rightarrow a, B \rightarrow bB, B \rightarrow b \}$

در این مثال هم رشته‌های زبان L از دو قسمت گونه A و گونه B تشکیل شده‌اند.

قسمت A در این مثال شامل رشته‌هایی با تعداد زوج a است.

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

✓ حروف بزرگ متغیر، حروف کوچک علامات نهایی هستند. یعنی رشته‌های نهایی زبان باید فقط دارای علائم و حروف کوچک باشند.

Subject:

Year. Month. Date. ( )

✓ گرامری می‌تواند بصورت یک الگوریتم در نظر گرفته شود که در واقع رشته‌های زبان را تولید می‌کند. از حالت اولیه شروع می‌کنیم و با طی مراحلی یک رشته زبان را تولید می‌کنیم.

! برای اینکه ببینیم یک رشته داده شده در زبان $L(G)$ (زبانی که توسط گرامر G تعریف شده است) هست یا نه؟ باید ببینیم آیا می‌شود با شروع از حالت اولیه و با طی مراحلی که گرامر G مشخص می‌کند به رشته داده شده برسیم یا نه؟ در واقع باید نرم کلی رشته‌های $L(G)$ را بشناسیم.

💡 اگر زبانی دارای بی‌نهایت رشته باشد، تنها روشی برای اینکه بتوانیم گرامری را بنویسیم که آن زبان را تولید کند، استفاده از روابط Recursion است و برعکس اگر در تعریف گرامر بازگشت مستقیم یا غیر مستقیم نداشته باشیم، تعداد رشته‌های زبان محدود است.

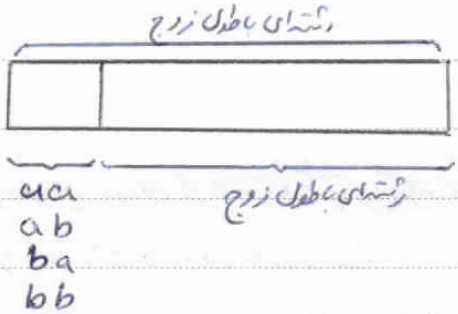
مثال: گرامری که زبان مورد را تولید می‌کند، بنویسید.

$$L = \{ w \mid \underbrace{w \in \{a,b\}^*}_{\text{تمام رشته‌هایی که از a و b تشکیل می‌شوند.}} , \underbrace{|w| \bmod 2 = 0}_{\text{رشته‌هایی که طول آنی مضرب 2 است}} \}$$

✓ برای توصیف یک زبان گرامرهای زیادی می‌تواند وجود داشته باشد.

✓ چون تعداد رشته‌های زبان بی‌نهایت است، پس باید بازگشتی فکر کنیم!

⚠ بعد از نوشتن گرامر باید مطمئن شویم که گرامر فقط و فقط رشته‌های زبان را تولید می‌کند، نه بیشتر و نه کمتر.



S ⟿ رشته‌ای که طول آن زوج است.

$S \to abS \mid S \to aaS \mid S \to bbS \mid S \to baS$

✓ قسمت برگشتی تعریف بازگشتی بالا برای رشته‌های با طول زوج است.

$S \to \lambda$ رشته‌ای با طول زوج

$S \to aaS \mid abS \mid baS \mid bbS$ تعریف بازگشتی است.

برای آنکه نشان دهیم گرامر بالا دقیقاً زبان L را تولید می‌کند (نه کمتر و نه بیشتر)، باید نشان دهیم:

$L = L(G)$

$L \subset L(G)$ و $L(G) \subset L$ برای نشان دادن تساوی فوق، باید نشان دهیم:

$\alpha \in L \Rightarrow \alpha \in L(G)$: هر رشته‌ای که در ... صدق می‌کند طول زوج است که توسط گرامر G تولید می‌شود.

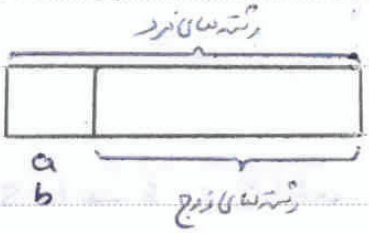
$\alpha \in L(G) \Rightarrow \alpha \in L$: به اینجا با روش استقرا که رابطه گرامر را بدست آوردیم، نشان می‌دهیم رشته‌های تولید شده توسط گرامر G برای طول زوج هستند و لذا در L هم وجود دارند.

✓ بنابراین اگر استقرایی که با آن گرامر را بدست آوردیم صحیح باشد، رابطه $L(G) \subset L$ برقرار است و کافیست تا فقط $L \subset L(G)$ را اثبات کنیم.

Subject:

Year. Month. Date. ( )

مثال: $L = \{w \mid w \in \{a,b\}^*, |w| \bmod 2 = 1\}$



باز هم باید به دنبال یک تعریف بازگشتی برای گرامر باشیم زیرا تعداد رشته‌های زبان نامحدود است.

S_o ⟵ رشته‌های با طول فرد

S_o ⟵ حالت اولیه

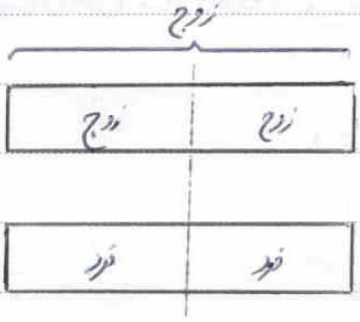
$S_o \longrightarrow aS_e \mid bS_e$

$S_e \longrightarrow aaS_e \mid abS_e \mid baS_e \mid bbS_e$

رشته‌ای با طول زوج (S_e) باید بازگشت را داشته باشد.

$S_e \longrightarrow \lambda$

مثال: $L = \{w \mid w \in \{a,b\}^*, |w| \bmod 2 = 0\}$ (روش حل دیگر)



هر رشته با طول زوج را می‌توانیم به دو قسمت با طول زوج تقسیم کنیم یا به دو قسمت با طول فرد (برای رشته‌های با طول ≥ 2) تقسیم کنیم.

$S_e \longrightarrow S_eS_e \mid S_oS_o$

$S_e \longrightarrow aaS_e \mid abS_e \mid baS_e \mid bbS_e$

$S_e \longrightarrow \lambda$

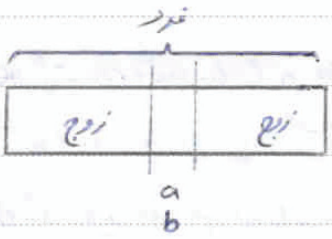
$S_o \longrightarrow aS_e \mid bS_e$

Subject:

Year. Month. Date. ( )

مثال: (روش حل سیگر)

$L = \{ w \mid w \in \{a,b\}^*, \ |w| \bmod 2 = 1 \}$



$S_0 \rightarrow SeaSe \mid SebSe$

$Se \rightarrow aaSe \mid abSe \mid baSe \mid bbSe$

$Se \rightarrow \lambda$

اکنون بی خواهیم رشته «abbababbb aa» را تولید کنیم.

abbababbbaa

b a

$S_0 \Rightarrow SebSe \Rightarrow abbababbbSe$

$\Rightarrow abSebSe \Rightarrow abbababbbaaSe$

$\Rightarrow abbaSebSe \Rightarrow abbababbbaa$

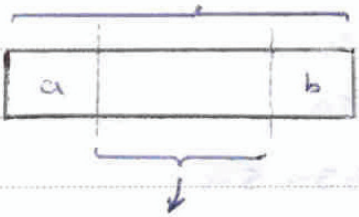
$\Rightarrow abbabaSebSe$

$\Rightarrow abbababSe$

مثال:

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

تعدادی a و تعدادی b که تعداد هر دو مساوی است.



رشته‌ای با همان خصوصیت

$S \rightarrow aSb$

$S \rightarrow \lambda$

بعد از نوشتن گرامر با بررسی گرامر

راهم اثبات کنیم.

$L = L(G)$ ⟸

$L \subset L(G)$ ⟺ $x \in L \Rightarrow x \in L(G)$ رشته‌ای که دارای خصوصیت L باشد، توسط گرامر G تولید می‌شود.

$L(G) \subset L$ ⟺ $x \in L(G) \Rightarrow x \in L$ رشته‌ای که توسط گرامر G تولید می‌شود، دارای خصوصیت L می‌باشد.

PAPCO

Subject:
Year. Month. Date. ( )

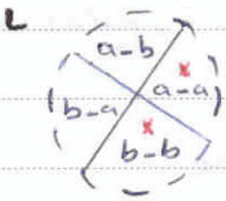
*مثال:

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

($n_a(w)$ → number of a's ، $n_b(w)$ → number of b's)

زبان بالا رشته‌هایی با تعداد مساوی a و b را تعریف می‌کند. باز هم به دلیل نامحدود بودن تعداد رشته‌های زبان، تنها مکانیزم تولید این رشته‌ها، استفاده از روابط بازگشتی می‌باشد.

برای حل مسئله باید سعی کنیم مسئله را به مسائل ساده‌تر تبدیل کنیم. مسائل ساده‌تر را حل کنیم سپس جوابها را ادغام کنیم.



رشته‌های زبان L چهار صورت کلی به صورت روبرو می‌توانند داشته باشند.

حالات a-a , b-b را می‌توان با استفاده بازگشت هم بدست آورد.

$S \rightarrow aSb \mid bSa \mid SS$

$S \rightarrow \lambda$

رابطه $S \rightarrow SS$ حالات a-b , b-a را شامل نمی‌شود، فقط مربوط به حالات a-a , b-b می‌باشد.

می‌خواهیم رشته روبرو بدست آوریم.

abba abab bb ab aa ab ba

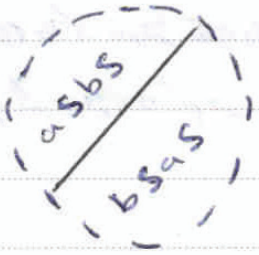
S ⇒ SS
⇒ SSS
⇒ aSbSS
⇒ abSS
⇒ abbSaS
⇒ abbaS
⇒ abbaSS
⇒ abbaaSbS
⇒ abbaabSabS
⇒ abbaababS
⇒ abbaabab bSa
⇒ abbaababbSSa
⇒ abbaababbbSaSa
⇒ abbaababbbaSbaSa
⇒ abbaababbbabaSa
⇒ abbaababbbabaaSba
⇒ abbaababbbabaaaSbba
⇒ abbaababbbabaaabba

Subject:

Year. Month. Date. ( )

مثال: (روش حل دیگر) $L = \{ w \mid w \in \{a,b\}^*, \ n_a(w) = n_b(w) \}$

L



$S \rightarrow aSbS \mid bSaS$

$S \rightarrow \lambda$

یا باز هم می‌توان گرامرهای دیگری را برای این زبان ارائه داد.

$S \rightarrow aSb \mid bSa \mid SS \mid SSS$

$S \rightarrow \lambda$

دو رابطه تعداد رشته‌ای در زبان فوق را می‌توان به صورت تعداد رشته با تعداد بسیاری a و b تقسیم نمود.

تعریف: دو گرامر G_1 و G_2 معادل (equivalent) هستند اگر و فقط اگر: $L(G_1) = L(G_2)$

(زبان‌هایی که دو گرامر G_1 و G_2 تولید می‌کنند باید معادل باشند)

مثال: آنگرز برای مثال بالا مثال‌های دوم دو گرامر فوق معادل هستند.

$S \rightarrow aSb \mid bSa \mid SS \mid \lambda \quad (G_1)$

$S \rightarrow aSb \mid bSa \mid SS \mid SSS \mid \lambda \quad (G_2)$

$L(G_1) = L(G_2)$

$L(G_1) \subset L(G_2) \rightsquigarrow x \in L(G_1) \Rightarrow x \in L(G_2)$

تعداد رشته‌ای که توسط G_1 تولید می‌شود، توسط G_2 هم تولید می‌شود.

تعداد تعدادهایی که در G_1 هست در G_2 هم هست لذا رابطه بالا درست می‌باشد.

$L(G_2) \subset L(G_1) \rightsquigarrow x \in L(G_2) \Rightarrow x \in L(G_1)$

تعداد رشته‌ای که توسط G_2 تولید می‌شود، توسط G_1 هم تولید می‌شود.

G_1: $S \Rightarrow SS \Rightarrow SSS$

تعداده $S \rightarrow SSS$ را G_1 تولید می‌کند می‌توان اثبات آن را ارائه کرد.

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

تعدادی b که بعد از تعدادی a می‌آیند. پس
آنچه که زبان بعضی از concatination دو گونه حاصل می‌شود.

$S \rightarrow AB$

$A \rightarrow aA \mid a$ ← a حداقل یک

$B \rightarrow bB \mid b$ ← b حداقل یک

مثال: $L = \{a^n b^{2m+1} \mid n, m \geq 0\}$

تعدادی زوج از b ها که پس از a ها می‌آیند.

$S \rightarrow AB$

$A \rightarrow aA \mid \lambda$

$B \rightarrow bC$

$C \rightarrow bB$

$B \rightarrow b$

قاعده $A \rightarrow \lambda$ یا بازگشت با حداقل صفر را ایجاد می‌کند.
تعدیف B از نوع تعدیف بازگشتی غیرمستقیم است.

Subject:

Year. Month. Date. ( )

* مثال: گرامر زبان روبرو را بدست آورید. $L = \{ w \mid w \in \{a,b\}^*, n_a(w) = n_b(w) \}$

✓ با نظر گرفتن یک رشته از زبان فوق، با جابجا کردن یک a و b می‌توان تمام رشته‌ها با آن تعداد a و b را بدست آورد. لذا برای بدست آوردن تمام رشته‌هایی با تعداد مساوی a و b، یک رشته با آن تعداد a و b انتخاب می‌کنیم و سپس با جابجا کردن a و b ها بقیه رشته‌ها را هم بدست آوریم.

از همین ایده برای نوشتن گرامر زبان بالا استفاده می‌کنیم.

$S \to ABS \mid \lambda$

$AB \to BA$

$BA \to AB$

$A \to a$

$B \to b$

$S \overset{*}{\Rightarrow} (AB)^n S \quad n \geq 0$

$\Rightarrow (AB)^n$ "

$\overset{*}{\Rightarrow} (ab)^n$ "

برای مثالی خواهیم رشته "abbbaa" را تولید کنیم.

$S \Rightarrow ABS$

$\Rightarrow ABABS$

$\Rightarrow ABABABS$

$\Rightarrow ABABAB$

$\Rightarrow ABBAAB$

$\Rightarrow ABBABA$

$\Rightarrow ABBBAA$

$\Rightarrow aBBBAA$

$\Rightarrow abBBAA$

$\Rightarrow abbBAA$

$\Rightarrow abbbAA$

$\Rightarrow abbbaA$

$\Rightarrow abbbaa$

Subject:
Year. Month. Date. ( )

مثال: رشته‌های زبان روبرو را با استفاده از گرامر آن بدست آورید.

$P = \{ S \to abc,\ S \to aXbc,\ Xb \to bX,\ bY \to Yb,\ Xc \to Ybcc,\ aY \to aaX \mid aa \}$

گرامر فوق در واقع رشته‌هایی را تولید می‌کند که تعداد a ، سپس تعدادی b و در نهایت تعدادی c می‌آیند که تعداد a ، b و c عدا با هم برابر است.

$S \Rightarrow aXbc$
$\Rightarrow abXc$
$\Rightarrow abYbcc$
$\Rightarrow aYbbcc$
$\Rightarrow aaXbbcc$
$\Rightarrow aabXbcc$
$\Rightarrow aabbXcc$
$\Rightarrow aabbYbccc$
$\Rightarrow aabYbbccc$
$\Rightarrow aaYbbbccc$
$\Rightarrow aaaXbbbccc$
$\Rightarrow aaabXbbccc$
$\Rightarrow aaabbXbccc$
$\Rightarrow aaabbbXccc$
$\Rightarrow aaabbbYbcccc$
$\Rightarrow aaabbbbYbccccc$
$\Rightarrow aaabbbbbYbcccccc$
$\Rightarrow aaaabbbbbccccc$

! گرامر فوق از نوع گرامرهای حساس به متن‌ای باشد.
مثلا در قاعده $Xc \to Ybcc$ در واقع X با Ybc جایگزین شده‌اما این جایگزینی فقط زمانی ممکن است که بعد از X یک c بیاید، یعنی جایگزینی X با Ybc بستگی به متن دارد.

✓ X مثل یک pointer می‌ماند که یکی یکی جلو می‌رود و a و b و c ها را تولید می‌کند.

! اثبات حالت کلی با استناد از استقرای امکان‌پذیر است.

Subject:

Year. Month. Date. ( )

✓ انواع گرامرها:

- گرامرهای نوع سوم (منظم)
- گرامرهای نوع دوم (مستقل از متن)
- گرامرهای نوع اول (حساس به متن)
- گرامرهای نوع صفر (بدون محدودیت)

• گرامر نوع سوم: یک گرامر از نوع سوم است اگر قواعد به فرم زیر باشند.

$$\begin{cases} A \to aB \\ A \to a \end{cases} \quad \text{or} \quad \begin{cases} A \to Ba \\ A \to a \end{cases} \qquad A,B \in V,\ a \in T$$

(منظم یکی از حالات زبان باز است) «خطی از سمت چپ» «خطی از سمت راست»

مثال:

$P_1 = \{ S \to aA,\ A \to aA,\ A \to b \}$ $\quad L_1 = \{ a^n b \mid n \geq 1 \}$

$P_2 = \{ S \to aS \mid bS,\ S \to a \mid b \}$ $\quad L_2 = \{a,b\}^+$

• گرامرهای نوع دوم: یک گرامر از نوع دوم است اگر قواعد به فرم زیر باشند.

$A \to \alpha$ $\quad A \in V,\ \alpha \in (V \cup T)^*$ (سمت چپ یک متغیر تکی است)

مثال:

$P_1 = \{ S \to aSb \mid bSa,\ S \to SS,\ S \to \lambda \}$

$P_2 = \{ S \to AB,\ A \to aA \mid b,\ B \to bB \mid b \}$

• **گرامر نوع اول:** یک گرامر از نوع اول (حساس به متن) است اگر قواعد به فرم زیر باشند.

$$\alpha \to \beta \qquad \alpha \in (V \cup T)^+ \ , \ \beta (V \cup T)^* \ , \ |\alpha| \leq |\beta|$$

مثال:

$$P = \{ S \to abc \ , \ S \to aXbc \ ,$$
$$Xb \to bX \ , \ bY \to Yb$$
$$Xc \to Ybcc \ , \ aY \to aaX \ ,$$
$$aY \to aa \}$$

• **گرامر نوع صفر:** به گرامر نوع صفر هیچ محدودیتی (مانند انواع قبل) وجود ندارد.

$$\alpha \to \beta \ , \ \alpha \in (V \cup T)^+ \ , \ \beta \in (V \cup T)^*$$

تمرین: برای گرامر زیر مولد زبان را اثبات کنید.

$$P = \{ S \to abc, \ S \to aXbc \ , \ Xb \to bX,$$
$$bY \to Yb \ , \ Xc \to Ybcc \ , \ aY \to aaX \ , \ aY \to aa \}$$

1) $S \overset{*}{\Rightarrow} a^n X b^n c^n \qquad (n \geq 1)$
2) $S \overset{*}{\Rightarrow} a^n b^n c^n \qquad (n \geq 1)$
3) $S \overset{*}{\Rightarrow} a^n b^n Y b c^{n+1} \qquad (n \geq 1)$

مثال: گرامر نوع صفر

$$P = \{ S \to ABaC,\ Ba \to aaB,\ BC \to DC|E,\ aD \to Da,\ AD \to AB,\ aE \to Ea,\ AE \to \lambda \}$$

تمرین: رشته‌های زبان گرامر بالا را پیدا کنید.

انواع زبان ها

- **زبان نوع سوم:** زبانی نوع سوم یا منظم است که بتوان برای آن یک گرامر نوع سوم یا منظم نوشت

$$L = \{ a^n b^m \mid n \geq 1, m \geq 2 \}, \quad P = \{ S \to aA,\ A \to aA,\ A \to bB,\ B \to bB|b \}$$

- **زبان نوع دوم:** زبانی از نوع دوم است که بتوان برای آن یک گرامر نوع دوم تعریف کرد ولی نتوان برای آن یک گرامر نوع سوم بدست آورد.

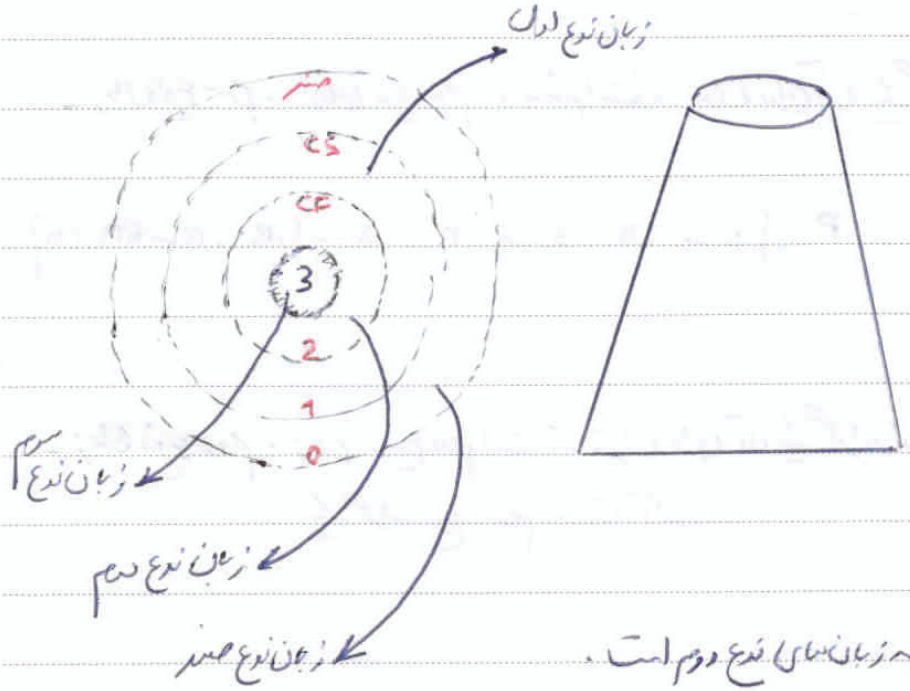
$$L = \{ a^n b^n \mid n \geq 0 \}, \quad P = \{ S \to aSb|\lambda \}$$

- زبان نوع اول: زبانی از نوع لدل است که بتوان برای آن یک گرامر نوع لدل طراحی کرد و نیز نتوان برای آن گرامر نوع سوم یا دوم طراحی کرد.

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

- زبان نوع صفر: زبانی که نتوان برای گرامرهای از نوع سوم، دوم یا لدل بدست آورد.

هرم زبان‌ها:



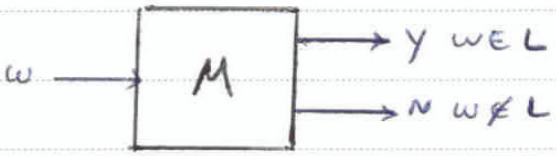
! زبان‌های نوع سوم در واقع زیر مجموعه زبان‌های نوع دوم است.

Subject:

Year. Month. Date. ( )

ماشین ها Automata

اگر یک ماشین برای زبان L طراحی شده باشد، یک رشته را گرفته و سپس اگر آن رشته در زبان L باشد، ماشین می گوید بله و اگر نباشد می گوید خیر!



acceptor , Recognizer

انواع ماشین ها

- پذیرنده برای زبان های نوع سوم — Finite State Machine
- پذیرنده برای زبان های نوع دوم — Push Down Automata «ماشین های پشته ای»
- پذیرنده برای زبان های نوع اول — Linear Bounded Automata
- پذیرنده برای زبان های نوع صفر — Turing Machine

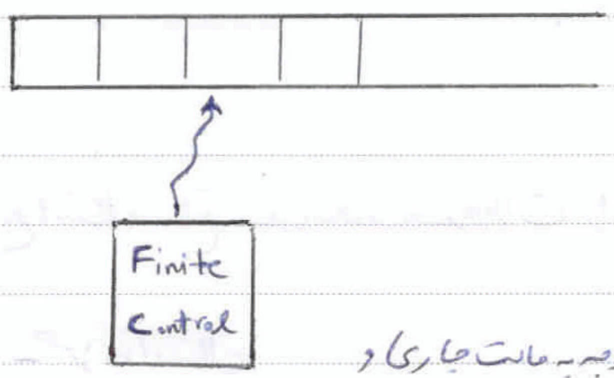
Subject :

Year . Month . Date . ( )

«نظریه زبان‌های نوع سوم»

پذیرنده زبان‌های نوع سوم: «Finit State Machine»

ماشین حالات متناهی از دو قسمت تشکیل شده است. یک نوار و یک کنترلر.



یک نوار (Tape) نیمه متناهی می‌باشد (Semi-Infinite)

در یک جهت حرکت می‌کند.

فقط از روی نوار می‌توان خواند (Read-only)

← ماشین در هر لحظه در یک State قرار دارد. با توجه به حالت جاری و اطلاعاتی که از روی Tape خوانده می‌شود حالت بعدی تعیین می‌شود و هد کنترل روی Tape جلو می‌رود.

✔ یک ماشین حالت متناهی بصورت رسمی به شکل زیر تعریف می‌شود.

$$M = (Q, \Sigma, \delta, q_0, F)$$

Subject:

Year. Month. Date. ( )

Q ⟵ Set of states — مجموعه حالات

Σ ⟵ Alphabet — الفبای زبان

δ ⟵ Transition Function — تابع انتقال

$\delta : Q \times \Sigma \rightarrow Q$

q_0 ⟵ Initial State — حالت اولیه

$q_0 \in Q$

F ⟵ Set of final States — مجموعه حالات نهایی

$F \subseteq Q$

✓ تابع انتقال در واقع همان نحوه تغییر حالات را مشخص می‌کند که به چند صورت مشخص می‌شود.

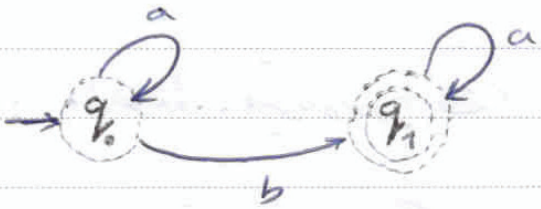
- گراف انتقال
- جدول انتقال
- تابع انتقال

✓ حالات رشته به ترتیب روی نوار ورودی خوانده می‌شود و بعد ابتدا به سمت چپ‌ترین حالت نشان می‌کند و سپس یکی یکی جلو می‌رود.

پذیرش: یک رشته پذیرفته می‌شود اگر در انتهای آن ماشین در یکی از حالات **نهایی** خود متوقف گردد.
و یک رشته پذیرفته نمی‌شود اگر در انتهای آن ماشین در یکی از حالات غیرنهایی خود متوقف گردد و یا به دلایلی نتواند حرکت کند.

← حالت اولیه توسط یک فلش مشخص می‌شود و حالات نهایی توسط دو دایره مشخص می‌شوند.

مثال:



← «گراف انتقال»

abaa: $q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_1 \xrightarrow{a} q_1 \xrightarrow{a} q_1$ ✓

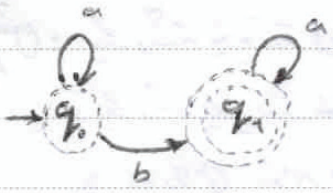
aaaa: $q_0 \xrightarrow{a} q_0 \xrightarrow{a} q_0 \xrightarrow{a} q_0$ ✓

aabb: $q_0 \xrightarrow{a} q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_1 \xrightarrow{b}$ X X (تعریف نشده)

δ	a	b
q_0	q_0	q_1
q_1	q_1	-

← «جدول انتقال»

سوال: ماشین حالت متناهی قدرت مربوط به چه زبانی است.
یعنی چه رشته هایی را به این ماشین بدهیم، در حالت نهایی متوقف می‌شود.

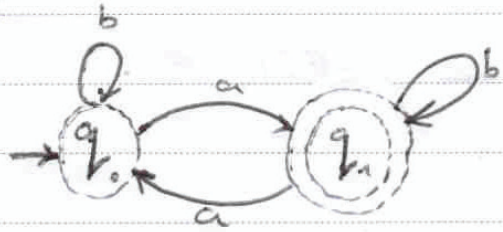


$q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_1 \xrightarrow{a} q_1$

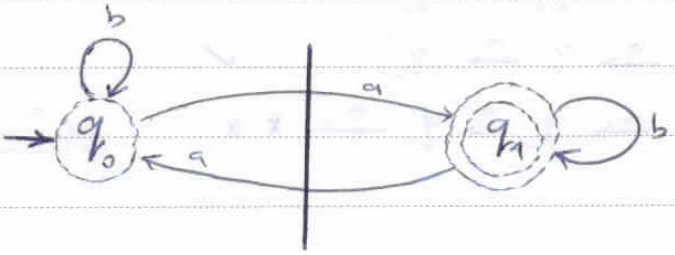
زن loop ↙

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

مثال: زبان مربوط به ماشین روبرو چیست؟



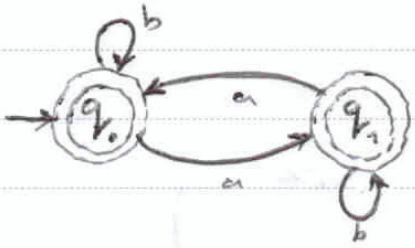
بدست به ماشین قدرت به تقدیر می‌گیریم.



برای آنکه رشته معتبر باشد، ماشین در حالت نهایی متوقف شود باید تعداد a ها فرد باشد.

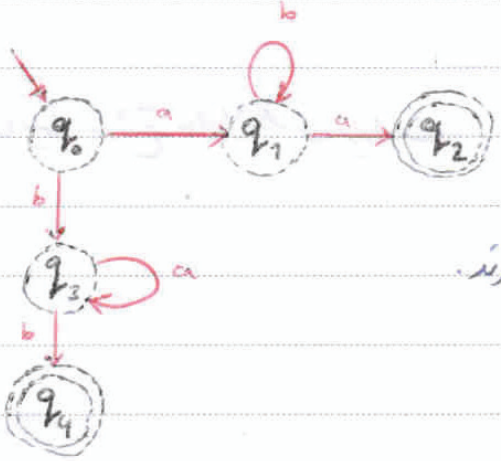
$L = \{w \mid w \in \{a,b\}^*, n_a(w) \bmod 2 = 1\}$

تعداد b ها نیز مهم نیست هر تعداد b های دلخواهی می‌توانند وجود داشته باشند.



مثال: زبان ماشین مقابل را بدست آورید.

$$L_{q_0} = \{ w \mid w \in \{a,b\}^*, \ n_a(w) \bmod 2 = 0 \}$$
$$L_{q_1} = \{ w \mid w \in \{a,b\}^*, \ n_a(w) \bmod 2 = 1 \}$$
$$L = L_{q_0} \cup L_{q_1} = \{ w \mid w \in \{a,b\}^* \}$$



مثال: زبان ماشین مقابل چیست؟

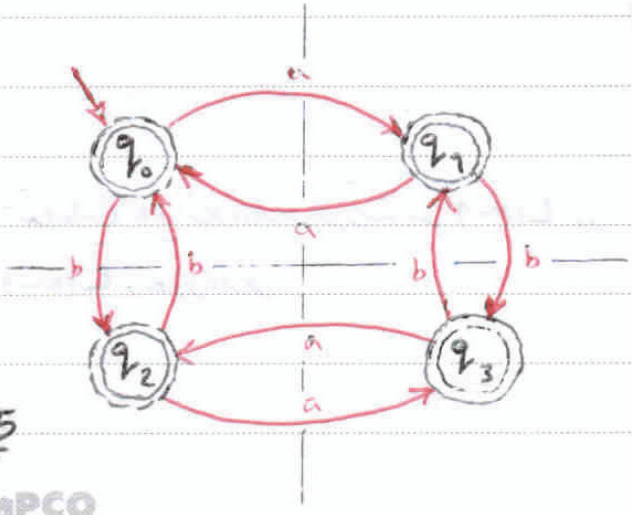
همه را به دو قسمت می توانیم تقسیم کنیم.

مسیرهایی که به q_2 ختم می شوند و مسیرهایی که به q_4 ختم می شوند.

مسیر q_2 ⟸ تعدادی b که دو طرف آن a است.

مسیر q_4 ⟸ تعدادی a که دو طرف آن b است.

$$L_{q_2} = \{ ab^n a \mid n \geq 0 \}$$
$$L_{q_4} = \{ ba^n b \mid n \geq 0 \} \Longrightarrow L = \{ ab^n a \mid n \geq 0 \} \cup \{ ba^n b \mid n \geq 0 \}$$



(*) مثال: زبان ماشین مقابل را بدست آورید.

می توانیم ماشین را به چهار قسمت تقسیم کنیم.

بین یک چهارم معادل توانیم جابجا شویم.

q_3 حالت نهایی ⟹ a فرد و b فرد

q_1 حالت نهایی ⟹ a فرد و b زوج

q_2 حالت نهایی ⟹ a زوج و b فرد

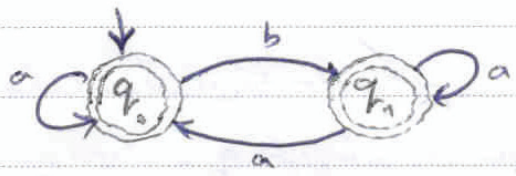
q_0 حالت نهایی ⟹ a زوج و b زوج

a فرد ⟹ a فرد

a زوج ⟹ a زوج

همه رشته‌ها ⟹ a فرد / a زوج

این ماشین به واقع معادل ماشین زیر است.



تعریف: دو ماشین M_1 و M_2 معادل هستند اگر و فقط اگر:

$$L(M_1) = L(M_2) \begin{cases} L(M_1) \subseteq L(M_2) \\ L(M_2) \subseteq L(M_1) \end{cases}$$

بنابراین باید نشان دهیم هر رشته در $L(M_1)$ در $L(M_2)$ وجود دارد و هر رشته‌ای در $L(M_2)$ در $L(M_1)$ وجود دارد.

Subject:
Year. Month. Date. ( )

تابع δ^*

$$\delta^* : Q \times \Sigma^* \longrightarrow Q$$

! δ^* را می‌توانیم بصورت بازگشتی حساب کنیم.

$w = aZ$

$$\delta^*(q, aZ) = \begin{cases} \delta^*(\delta(q,a), Z) & \text{(بازگشت)} \\ \delta(q,a) \quad \text{if } |Z| = 0 & \text{(پایه)} \end{cases}$$

مثال: مقدار روبرو را بدست آورید. (ماشین متناهی مثال قبل)

$$\delta^*(q_0, abab)$$

$$\delta^*(\underbrace{\delta(q_0,a)}_{q_1}, bab) = \delta^*(\underbrace{\delta(q_1,b)}_{q_3}, ab)$$

$$= \delta^*(\underbrace{\delta(q_3,a)}_{q_2}, b) = \delta(q_2, b)$$

! در واقع تابع δ^* یک روش و تعریف ریاضی برای نشان دادن سیر حالتی یک ماشین متناهی است.

❗ تعریف بازگشتی دیگری برای تابع δ^* بصورت زیر است.

$w = za$

$$\delta^*(q, za) \begin{cases} \delta(\delta^*(q, z), a) \\ \delta(q, a) \quad \text{if } |z| = 0 \end{cases}$$

✓ $L(M)$: زبان پذیرفته شده توسط ماشین M که بصورت زیر تعریف می‌شود.

$$L(M) = \{w \mid \delta^*(q_0, w) \in F\}$$

تمرین: تعریف بازگشتی دیگر برای تابع δ^*

$w = xz$

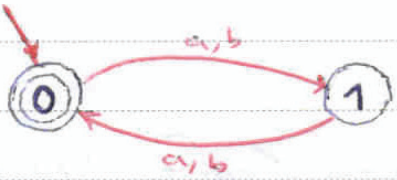
$$\delta^*(q, xz) \begin{cases} \delta^*(\delta^*(q, x), z) \\ ? \quad (\text{یا } \simeq) \end{cases}$$

Subject:
Year. Month. Date. ( )

مثال: ماشین زبان زیر را پیدا کنید. $L = \{w \mid w \in \{a,b\}^*, |w| \bmod 2 = 0\}$

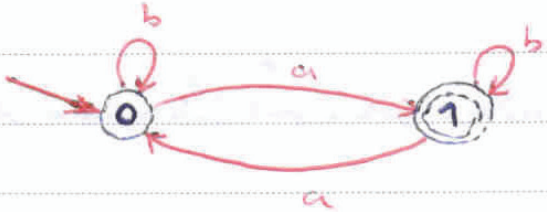
✓ در واقع باید یک الگوریتمی طراحی کنیم که یک رشته از زبان بگیرد و بگوید که آیا در زبان مد نظر ما وجود دارد یا خیر. الگوریتم فوق باید به گونه‌ای باشد که بتوان در قالب ماشین‌های معرفی شده، آن را بیان کرد.

در مثال فوق یک متغیر مثل c با تعدادی صفر داریم. به ازای هر کاراکتری که از رشته می‌خوانیم اگر تعداد آن صفر بود، تعداد آن را یکی و اگر تعداد آن یک بود، تعداد آن را صفر می‌کنیم. در واقع متغیر c در هر لحظه باقی تقسیم طول رشته خوانده شده تا آن لحظه بر 2 را می‌دهد.
اکنون همین الگوریتم را بصورت ماشین حرفی آوریم.



مثال: ماشین زبان روبرو را بدست آورید. $L = \{w \mid w \in \{a,b\}^*, n_a(w) \bmod 2 = 1\}$

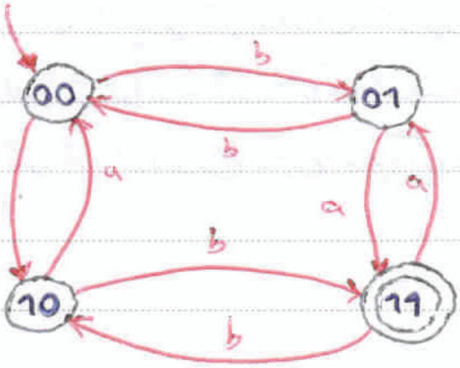
✓ برای طراحی الگوریتم پذیرنده زبان فوق، باز هم یک c با تعداد اولیه صفر در نظر می‌گیریم. این بار زمانی که a می‌خوانیم مقدار c را عوض می‌کنیم.



مثال: ماشین زبان زیر را بدست آورید.

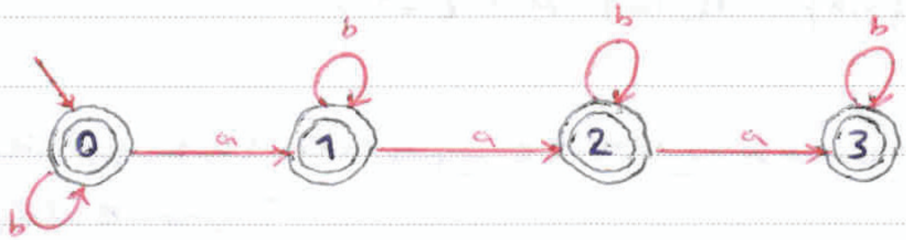
$$L = \{ w \mid w \in \{a,b\}^*,\ n_a(w) \bmod 2 = 1,\ n_b(w) \bmod 2 = 1 \}$$

برای طراحی الگوریتم این مسئله، این دفعه دو متغیر c و d در نظر می‌گیریم که c روی تعداد a ها کار می‌کند و d روی تعداد b ها کار می‌کند.



مثال: ماشین زبانی را بدست آورید که تعداد a ها در آن حداکثر 3 است.

در طول خواندن رشته‌ها، حالاتی که در نظر داشته باشیم حالاتی هستند که در آنها صفر یا یک یا دو یا سه a دیدیم.



اگر بیش از سه عدد a داشته باشیم، ماشین در حالت پذیرش نیست و تعریف نشده است.

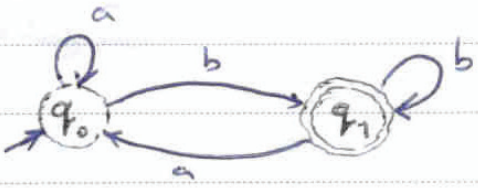
تمرین: ماشین زبان زیر را پیدا کنید.

$$L = \{ w \mid n_a(w) \bmod 3 > n_b(w) \bmod 3 \}$$

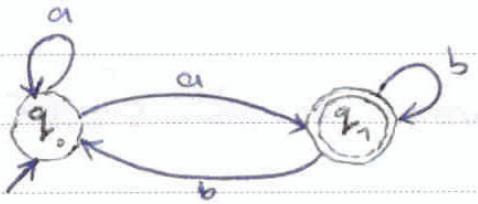
برای طراحی الگوریتم این ماشین باید مد متغیر c_1 و c_2 داشته باشیم در یک سه حالت $\underline{0}$، $\underline{1}$ و $\underline{2}$ را می‌پذیرند که نشان باقی‌ماندهٔ طول a یا b ها بر 3 است.
بنابراین 9 حالت برای ماشین داریم که حالات نهایی آن‌هایی هستند که شرط بالا را داشته باشند.

ماشین‌های پذیرنده برای زبان نوع سوم Finite State Automata

- ماشین‌های متناهی قطعی — Deterministic Finite State Automata — DFA
- ماشین‌های متناهی غیرقطعی — Non-Deterministic Finite State Automata — NDFA

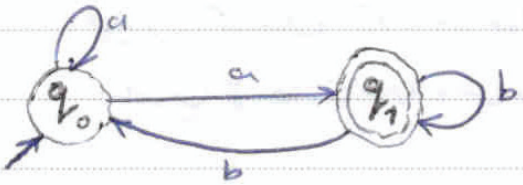


⟵ DFA $(\delta: Q \times \Sigma \to Q)$



⟵ NDFA (از q_0 با a نمی‌توان مستقیماً به q_0 رفت و نمی‌توان به q_1 رفت)

پذیرش در NDFA : ماشین NDFA پذیرنده‌ی زبان‌های نوع سوم، یک رشته پذیرفته‌شده است اگر حداقل یک مسیر با برچسب آن رشته از حالت اولیه به یکی از حالات نهایی وجود داشته باشد.



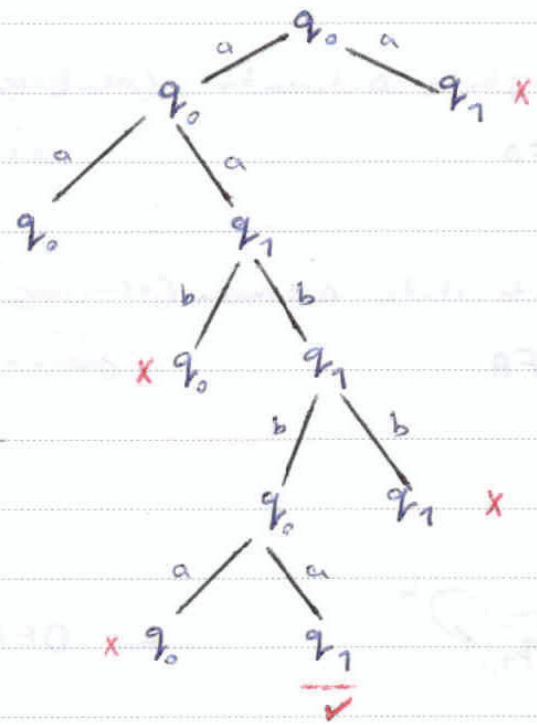
مثال: برای پیدا کردن مسیر می‌توانیم از یک مدت تصمیم‌گیری استفاده کنیم aabba

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

تابع گذار

	a	b
q_0	$\{q_0, q_1\}$	$\{\}$
q_1	$\{\}$	$\{q_0, q_1\}$

جدول گذار



✓ در واقع از روش Back Tracking استفاده نموده‌ایم

! رشته‌هایی که با حداقل یک a شروع می‌شوند.

! تابع دلتا برای ماشین غیرقطعی متناهی (NDFA) بصورت زیر تعریف می‌شود.

$$\delta : Q \times \Sigma \longrightarrow 2^Q$$ (مجموعه توانی Q)

Subject:

Year. Month. Date. ( )

! تابع δ^* نیز بصورت زیر تعریف می‌شود: (NOFA)

$$\delta^* : Q \times \Sigma^* \longrightarrow 2^Q$$

$$\delta^*(q_1, aZ) \begin{cases} \delta^*(\delta(q_1, a), Z) \\ \delta(q_1, a) \quad \text{if } |Z| = 0 \end{cases} \qquad w = aZ$$

$$\delta^*(q_1, Za) \begin{cases} \delta(\delta^*(q_1, Z), a) \\ \delta(q, a) \quad \text{if } |Z| = 0 \end{cases} \qquad w = Za$$

مثال:

$$\delta(q_0, aabba) = \delta^*(\underbrace{\delta(q_0, a)}_{\{q_0, q_1\}}, abba)$$

$$= \delta^*(q_0, abba) \cup \delta^*(q_1, abba)$$

$$= \delta^*(\underbrace{\delta(q_0, a)}_{\{q_0, q_1\}}, bba) \cup \underbrace{\delta^*(\underbrace{\delta(q_1, a)}_{\emptyset}, bba)}_{\emptyset}$$

$$= \delta^*(q_0, bba) \cup \delta^*(q_1, bba)$$

$$= \underbrace{\delta^*\big(\underbrace{\delta(q_0,b)}_{\emptyset}, ba\big)}_{\emptyset} \cup \underbrace{\delta^*\big(\underbrace{\delta(q_1,b)}_{\{q_0,q_1\}}, ba\big)}_{\{q_0,q_1\}}$$

$$= \delta^*(q_0, ba) \cup \delta^*(q_1, ba) = \underbrace{\delta^*\big(\underbrace{\delta(q_0,b)}_{\emptyset}, a\big)}_{\emptyset} \cup \underbrace{\delta^*\big(\underbrace{\delta(q_1,b)}_{\{q_0,q_1\}}, a\big)}_{\{q_0,q_1\}}$$

$$= \underbrace{\delta(q_0, a)}_{\{q_0,q_1\}} \cup \underbrace{\delta(q_1, a)}_{\emptyset} = \{q_0, q_1\}$$

✓ می‌توان تعریف پذیرش رشته‌ها بصورت زیر نیز با استفاده از δ^* تعریف کرد.

$$L(M) = \{x \mid \delta^*(q_0, x) \cap F \neq \emptyset\}$$

تمرین: با استفاده از تعریف دوم δ^* مثال قبل را حل کنید.

تمرین: برنامه‌ای بنویسید که یک رشته و یک ماشین بگیرد (NDFA or DFA) و سپس مشخص کند آیا این رشته در زبان ماشین وجود دارد یا خیر. ۲ نمره

« Table Driven Programming »

در حالت NDFA نیز باید با استفاده از Back Tracking یک پیمایش Pre Order روی درخت انجام دهیم.

تبدیل NDFA به DFA

اگر یک ماشین NDFA داشته باشیم و بخواهیم یک الگوریتم برای بررسی رشته‌ها روی این ماشین انجام دهیم، در اینصورت هزینه الگوریتم بصورت نمایی است.

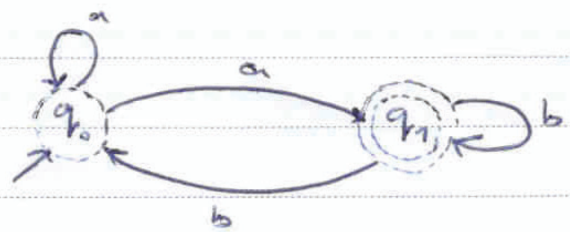
NDFA ⟶ $O(2^n)$

ولی هزینه الگوریتم بررسی رشته‌ها در ماشین DFA متناسب با طول رشته است.

DFA ⟶ $O(n)$

لذا اگر بتوانیم یک ماشین NDFA را به ماشین DFA تبدیل کنیم، بسیار مناسب‌تر است.

ماشین NDFA زیر را در نظر بگیریم و سعی می‌کنیم آن را به ماشین DFA تبدیل کنیم.



Subject:

Year. Month. Date. ( )

Input, $NDFA = (Q, \Sigma, \delta, q_0, F)$ accepting L

output: $DFA = (Q', \Sigma', \delta', q_0', F')$ accepting L

$Q' = 2^Q = \{[q_0], [q_1], [q_0, q_1], []\}$

$q_0' = [q_0]$, $\Sigma' = \Sigma$

F' ⟶ set of all states in Q' which contain at least one state in F

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

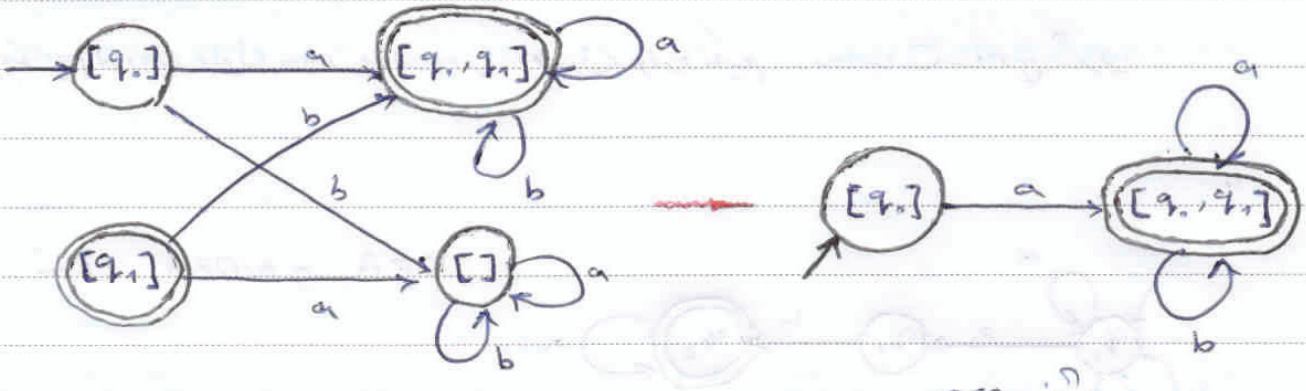
$\delta'([q_{i_0}, q_{i_1}, \dots, q_{i_p}], a) = [P_{j_0}, P_{j_1}, \dots, P_{j_k}]$

$\delta(\{q_{i_0}, q_{i_1}, \dots, q_{i_p}\}, a) = \{P_{j_0}, P_{j_1}, \dots, P_{j_k}\}$

	a	b
$[q_0]$	$[q_0, q_1]$	$[]$
$[q_1]$	$[]$	$[q_0, q_1]$
$[q_0, q_1]$	$[q_0, q_1]$	$[q_0, q_1]$
$[]$	$[]$	$[]$

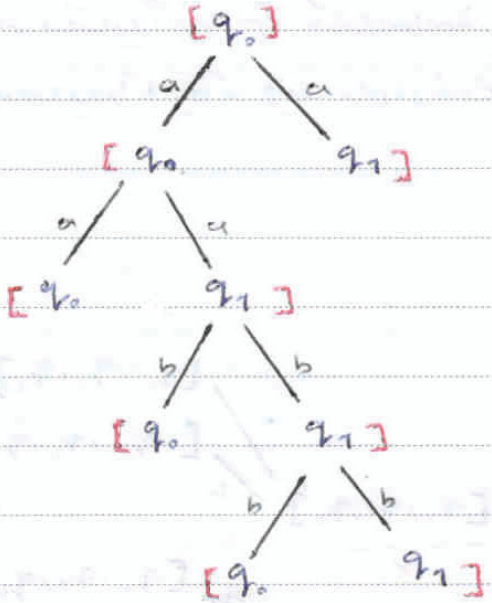
Subject :

Year . Month . Date . ( )

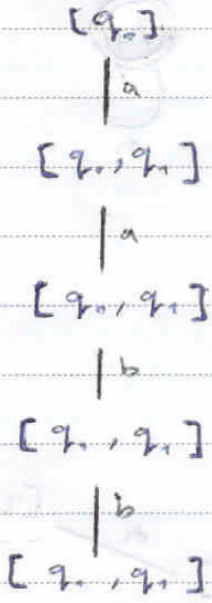


«DFA ماشین»

«aabb»



«NDFA ماشین»



«DFA ساده‌ماشین»

✓ در واقع این ماشین DFA هم دقیقاً همان کار ماشین NDFA را انجام می‌دهد.

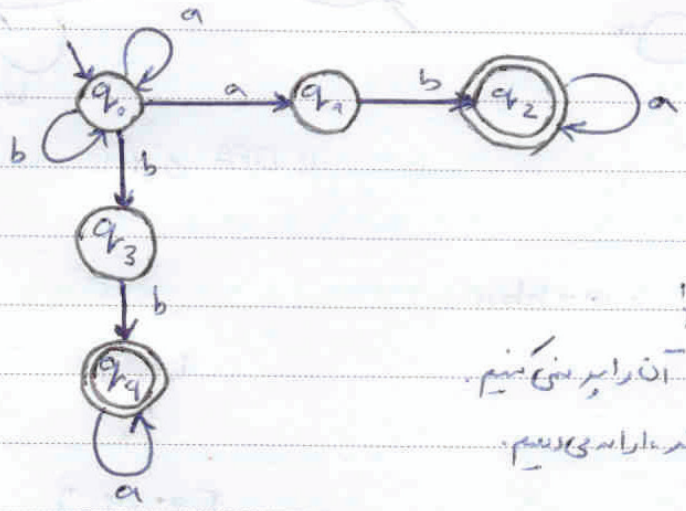
! در عمل اگر کاردینالیتی Q (تعداد اعضای Q) زیاد باشد، محاسبات بسیار پیچیده می‌شود.

Subject:

Year. Month. Date. ( )

! اگر بتوانیم از همان اول حالات اضافه را از جدول وارد نکنیم، محاسبات ساده‌تر می‌شود.

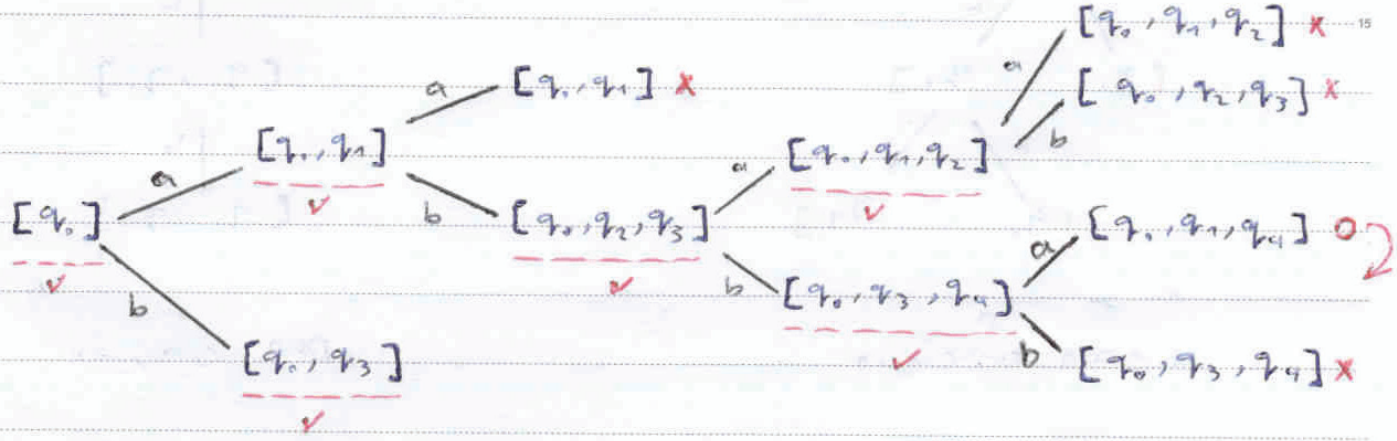
مثال: تبدیل NDFA به DFA

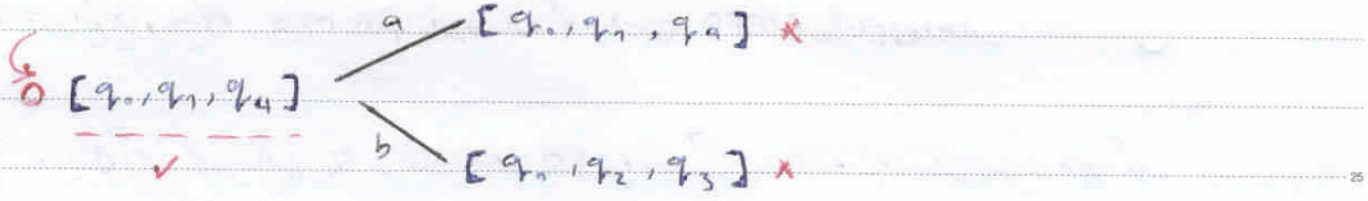


✓ اکنون می‌خواهیم جدول حالات را هوشمندانه‌تر پر کنیم!

✓ هر وقت زیر اگر با حالتی تکراری مواجه شدیم، آن را پر نمی‌کنیم.

✓ وقت زیر را آنجایی که چیزی برای باز شدن نداریم پر می‌کنیم.





Subject:

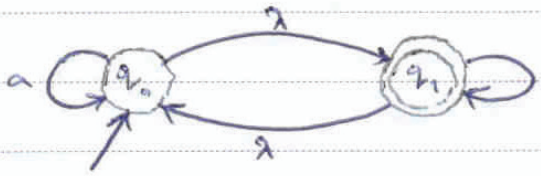
Year. Month. Date. ( )

تمرین: برنامه‌ای بنویسید که یک ماشین NDFA را به ماشین DFA تبدیل کند.

انواع ماشین‌های غیرقطعی

- بدون حرکات لامبدا (λ) — without λ-moves (λ-transition)
- با حرکات لامبدا (λ) — with λ-moves (λ-transition)

ماشین با حرکات λ ⟵ NDFAλ



✓ حرکت λ را می‌توان حرکتی در نظر گرفت که بدون توجه به ورودی یا چیزنگری به حالت دیگری برود. (ماشین انتخاب دارد و می‌تواند با توجه به شرایط به حالت دیگری برود)

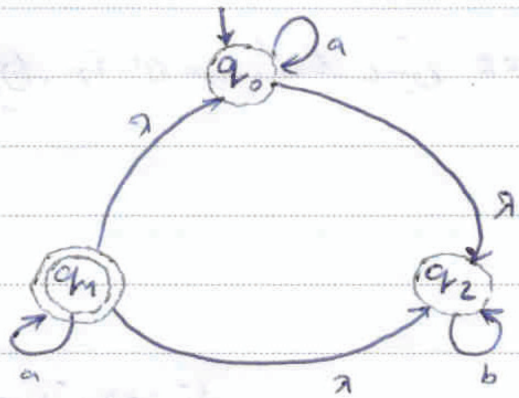
پذیرش ⟵ اگر یک مسیر از حالت اولیه به حالات نهایی با برچسب آن رشته وجود داشته باشد.

✓ برای مثال رشته aaaaba توسط ماشین غیرقطعی بالا با حرکت λ پذیرش می‌شود.
در واقع ماشین بالا هر رشته‌ای را می‌پذیرد!

✓ انتقال λ بدون توجه به آنچه از ورودی خوانده می‌شود، انجام می‌شود

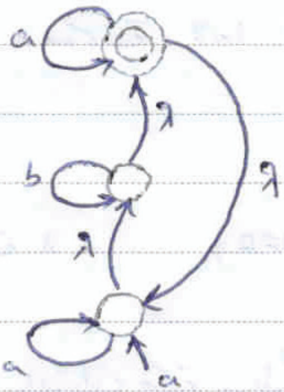
Subject:
Year. Month. Date. ( )

مثال: ماشین $NDFA_\lambda$ زیر رو هیچ رشته‌ای را نمی‌پذیرد.



$\{\} \neq \{\lambda\}$

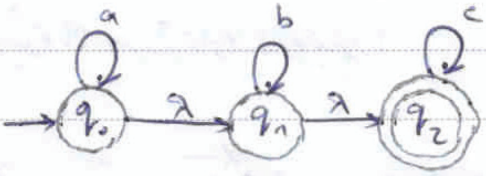
مثال:



ماشین زیر رو هر زبانی شامل a و b را می‌پذیرد.

★ λ - Closure ← مجموعه‌ی تمام حالاتی است که از q قابل حصول است فقط توسط حرکات λ.

(λ - closure یک state شامل خود آن state نیز می‌شود)



$$\lambda\text{-closure}(P) = \bigcup_{q \in P} \lambda\text{-closure}(q)$$

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

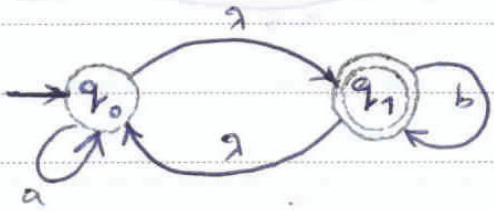
$\lambda\text{-closure}(q_1) = \{q_1, q_2\}$

تبدیل ماشین NDFA$_\lambda$ ~ NDFA

input: NDFA$_\lambda$ $(Q, \Sigma, \delta, q_0, F)$ accepting L

output: NDFA $(Q', \Sigma', \delta', q_0', F')$ accepting L

مثال:



$Q' = Q$, $q_0' = q_0$, $\Sigma' = \Sigma$

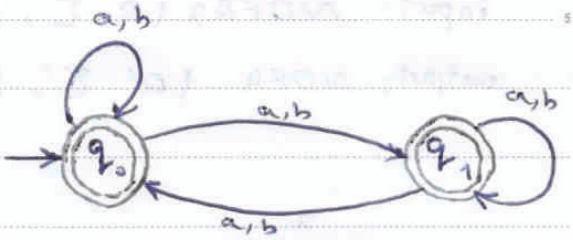
F' { $F \cup \{q\}$ if λ-Closure(q) contains a state in F
{ F otherwise

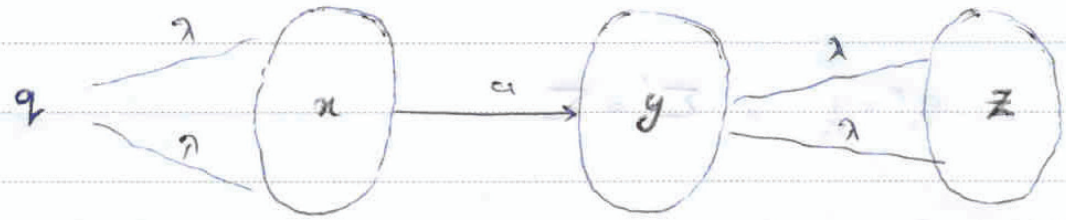
λ-Closure$(q_0) = \{q_0, q_1\}$ ⟶ $F' = \{q_0, q_1\}$

δ:	a	b	λ
q_0	$\{q_0\}$	$\{\}$	$\{q_1\}$
q_1	$\{\}$	$\{q_1\}$	$\{q_0\}$

$$\delta'(q, a) = \lambda\text{-closure}\Big(\delta\big(\lambda\text{-closure}(q), a\big)\Big)$$

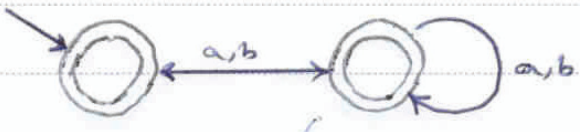
δ':	a	b
q_0	$\{q_0, q_1\}$	$\{q_0, q_1\}$
q_1	$\{q_0, q_1\}$	$\{q_0, q_1\}$



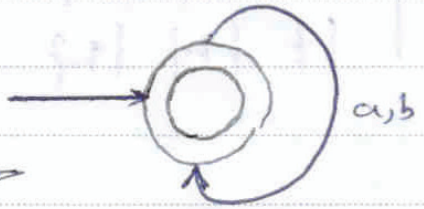


❗ اگر تعداد حالت‌های ماشین با حرکات λ را به یک ماشین بدون حرکت λ تبدیل کنیم، باید یک حرکت مستقیم با ورودی a از q به z داشته باشیم.

✔ یکی از ماشین‌های DFA معادل با ماشین NDFA بالا به صورت زیر است:



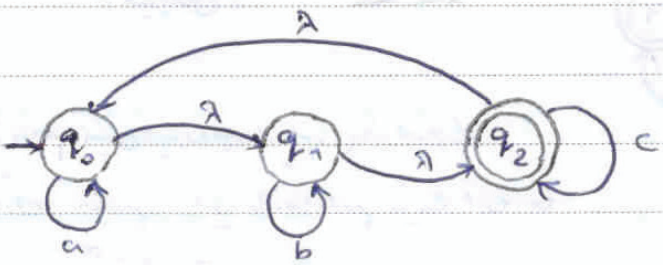
رویه جعبه سازی



Subject:

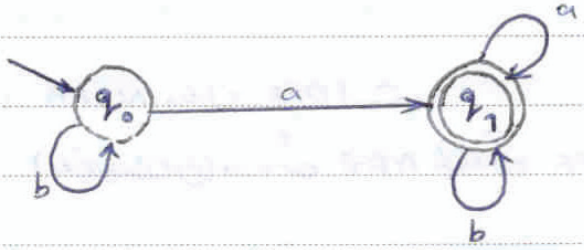
Year. Month. Date. ( )

تمرین: ماشین NDFA زیر را به یک ماشین DFA و سپس به یک ماشین DFA معادل با حداقل حالت تبدیل کنید.



عوامل بوجود آمدن عدم قطعیت

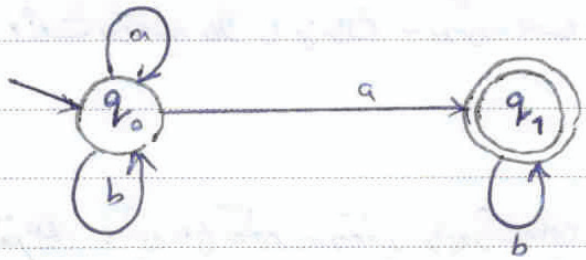
مثال: فرض کنیم می خواهیم یک ماشین پذیرنده برای مجموعه رشته هایی که تعدادکم از آنها دارای حداقل یک a باشد.



«قطعی» ⟵

در این حالت یک سفر روی رشته ورودی انجام می دهیم

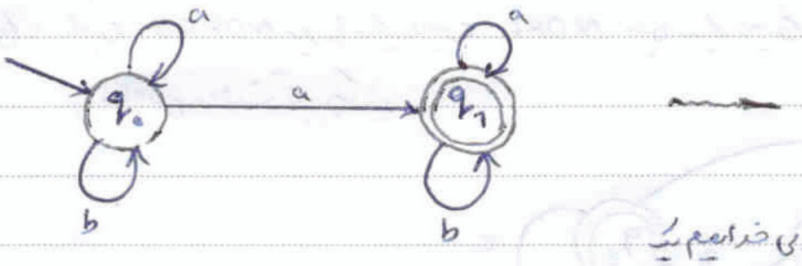
در این a را دیدیم به حالت نهایی می رویم و سپس هر چه a یا b آن به خود دیدیم در همان حالت نهایی می مانیم



«غیر قطعی» ⟵

در این حالت می خواهیم آخرین a را دیدیم به حالت نهایی برویم.

Subject:
Year. Month. Date. ( )



«غیرقطعی»

در این حالت ما ماشین غیرقطعی را داریم چون می‌خواهیم یک a در انتها ببینیم. در این راه حل مهم نیست این a چندمین a در رشته است.

✓ در واقع عدم قطعیت ناشی از نوع نگرش ما و راه حلی است که ارائه می‌دهیم. با استفاده از رویه‌های ارائه شده می‌توان ماشین غیرقطعی را به ماشین‌های قطعی تبدیل کرد.

قضیه: NDFA معادل DFA است.
(یعنی می‌توان برای هر ماشین NDFA یک ماشین DFA معادل ارائه داد)

قضیه: λNDFA معادل NDFA است.
(یعنی می‌توان برای هر ماشین λNDFA یک ماشین NDFA معادل ارائه داد)

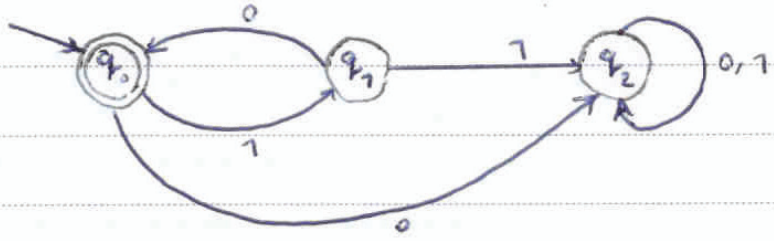
✓ اثبات دو قضیه بالا را می‌توان به روش ساخت ارائه نمود.

تمرین: ماشین‌های قطعی و غیرقطعی برای رشته‌هایی با حداقل دو a طراحی کنید.

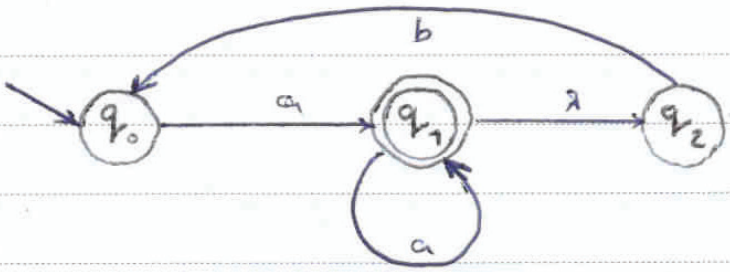
Subject:

Year. Month. Date. ( )

تمرین: زبان ماشین زیر را پیدا کنید.



تمرین: ماشین NDFA زیر را به یک ماشین DFA سپس یک ماشین DFA تبدیل نمایید.



Subject:

Year. Month. Date. ( )

Subject:

Year. Month. Date. ( )

«عبارات منظم و گرامرهای منظم»

Regular Expressions ⟶ عبارات منظم بیان دیگری از زبان‌های تولید شونده یا شناسنده زبان‌های نوع سوم می‌باشند.

Syntant ⟶ عبارات منظم بصورت بازگشتی تعریف می‌شوند.

1. $\emptyset$ یک عبارت منظم است.
2. λ یک عبارت منظم است.
3. $a \in \Sigma$ یک عبارت منظم است.
4. اگر R_1, R_2 عبارت منظم باشند، به اینصورت R_1+R_2, R_1R_2, (R_1), R_1^* نیز عبارات منظم هستند.

مثال: $\Sigma = \{0,1\}$ ⟶ $0 \in \Sigma$, $1 \in \Sigma$

$R = (0+1)^*(1+01)(0+1)^*$ ⟶ یک عبارت منظم که با استفاده از قواعد بالا بدست آمده است.

Subject:

Year. Month. Date. ( )

Semantic ——— Semantic یک عبارت منظم بصورت زیر تعریف می‌شود

1. $\emptyset$ بر $\{\}$ دلالت دارد.
2. λ بر $\{\lambda\}$ دلالت دارد.
3. $a \in \Sigma$ بر $\{a\}$ دلالت دارد.
4. اگر R_1، R_2 به ترتیب دلالت بر $L(R_1)$ ، $L(R_2)$ داشته باشند، به این صورت

$L(R_1 + R_2) = L(R_1) \cup L(R_2)$

$L(R_1 R_2) = L(R_1) \cdot L(R_2)$

$L((R_1)) = L(R_1)$

$L(R_1^*) = L(R_1)^*$

! در واقع Semantic یک عبارت منظم مشخص می‌کند که آن عبارت منظم چه زبانی دلالت می‌کند.

مثال:

$R = (0+1)^*(1+01)(0+1)^*$

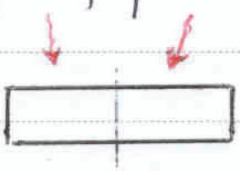
$0 : \{0\}$ ، $1 : \{1\}$ ، $0+1 : \{0\} \cup \{1\} = \{0,1\}$

$(0+1) : \{0,1\}$ ، $(0+1)^* = \{0,1\}^*$

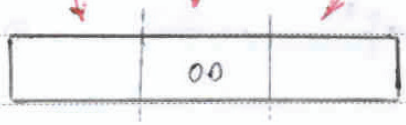
$01 : \{0\}\{1\} = \{01\}$ ، $1+01 : \{1\} \cup \{01\} = \{1, 01\}$

$(1+01) : \{1, 01\}$ ، $(0+1)^*(1+01) = \{0,1\}^*\{1, 01\}$

$(0+1)^*(1+01)(0+1)^* : \{0,1\}^*\{1,01\}\{0,1\}^*$



مثال: $R = (0+1)^*(00)(0+1)^* \longrightarrow \{0,1\}^*\{00\}\{0,1\}^*$



✓ مجموعه تمام رشته‌هایی که حداقل یک صفر دنباله‌شان.

! عبارات منظم را می‌توان بصورت الگوریتم هم در نظر گرفت که تولید کننده رشته‌های آن زبان‌اند.

* ← ستاره

\+ ← انتخاب

$R = (0+1)^*$

while (needed) do
  select 0 or 1

$R = (0+1)^* 00$

while (needed) do
  select 0 or 1
select 0
select 0

00111010011 00111010011

Subject:

Year. Month. Date. ( )

مثال: a^* : $\{a\}^*$ ⟹ رشته‌هایی با هر تعداد a

$(aa)^*$: $\{aa\}^*$ ⟹ رشته‌هایی با تعداد زوج a

$a(aa)^*$: $\{a\}\{aa\}^*$ ⟹ رشته‌هایی با تعداد فرد a

$a(aa)^* \equiv (aa)^*a$ ⟹ دو عبارت منظم معادل

✓ دو عبارت منظم R_1, R_2 معادل (Equvalent) یکدیگرند یا گفته

$L(R_1) = L(R_2)$

مثال: $a(aa)^*a$ ⟹ رشته‌هایی با طول زوج و حداقل 2 تا a

$a(aa)^*a + \lambda = (aa)^*$ ⟹ «یک معادله»

$a(aa)^*(bb)^*$ ⟹ $\{a^{2k+1}b^{2k'} \mid k \geq 0, k' \geq 0\}$

$(aa)^*ab(bb)^* \equiv a(aa)^*b(bb)^* \equiv a(aa)^*(bb)^*b$

Subject :

Year . Month . Date . ( )

مثال :

$$L = \{ a^n b^m \mid (n+m) \text{ is even} \}$$

زبان فوق را می‌توانیم به دو مجموعه افراز کنیم.

$$L_1 = \{ a^n b^m \mid n \text{ and } m \text{ are both even} \}$$

$$L_2 = \{ a^n b^m \mid n \text{ and } m \text{ are both odd} \}$$

$$L = L_1 \cup L_2$$

$$R_1 = (aa)^* (bb)^* \quad , \quad R_2 = a(aa)^* b(bb)^*$$

$$R = R_1 + R_2 \longrightarrow R = (aa)^* (bb)^* + a(aa)^* b(bb)^*$$

مثال :

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

زبان L را به سه زیر مجموعه مشترک تقسیم می‌کنیم.

$$L = \{ a^n b^m \mid n \geq 1, m \geq 3 \} \cup \{ a^n b^m \mid n \geq 3, m \geq 1 \} \cup \{ a^n b^m \mid n \geq 2, m \geq 2 \}$$

اگر رشته‌ای در زبان L باشد و شرایط مجموعه بالا را داشته باشد، حتماً یکی از سه شرط (سه زیرمجموعه بالا) حتماً صدق می‌کند و برعکس اگر رشته‌ای در یکی از سه زیرمجموعه‌های بالا باشد، در زبان L نیز می‌باشد.

$$R = aaa^* bbbb^* + aaaa^* bb^* + aaa^* bbb^*$$

در واقع در مثال بالا ما توانستیم زیرمجموعه‌های مشترک دیگری را هم اضافه کنیم و لذا می‌توانیم بی‌نهایت عبارات منظم برای زبان حذف داشته باشیم.

اما به نظر می‌رسد عبارت منظم بالا حداقل‌تر است که باید وجود داشته باشد.

Subject :

Year . Month . Date . ( )

مثال: $L = \{ w \mid w \in \{a,b\}^* , |w| \bmod 2 = 0 \}$

برای پیدا کردن عبارت منظم زبان بالا رشته‌های با طول زوج را به صورت زیر در نظر می‌گیریم.

رشته با طول زوج از رشته با طول 2 تشکیل شده است که یا aa یا ab یا ba یا bb می‌باشد.

```
while (needed) do
    select
        aa
    or
        ab
    or
        ba
    or
        bb
```

$R = (aa + ab + ba + bb)^*$

$R = \left((a+b)(a+b) \right)^*$

راه حل دیگر

مثال: ★ $L = \{ vwv \mid v, w \in \{a,b\}^* , |v| = 2 \}$

✗ $R = (aa + ab + ba + bb)(a+b)^* (aa + ab + ba + bb)$ ✗

! این راه حل ساده را ابتدا به نظر می‌رسد ولی جواب صحیح نمی‌باشد زیرا هیچ کنترلی نداریم تا زیر رشته‌های 2 تایی آخر یکی باشند.

$R = aa(a+b)^* aa + ab(a+b)^* ab + ba(a+b)^* ba + bb(a+b)^* bb$

Subject:

Year. Month. Date. ( )

معادلات عبارات منظم

فرض کنید α، β، γ عبارات منظم باشند، در اینصورت:

1) $\alpha + \beta = \beta + \alpha$

2) $\alpha(\beta + \gamma) = \alpha\beta + \alpha\gamma$ ، $(\beta + \gamma)\alpha = \beta\alpha + \gamma\alpha$

3) $\alpha\alpha^* + \lambda = \alpha^*$

4) $(\alpha^*)^* = \alpha^*$

5) $(\alpha+\beta)^* = \alpha^*(\alpha+\beta)^*$ ، $(\alpha+\beta)^* = \alpha^*(\alpha+\beta)^*\beta^*$

6) $(\alpha+\beta)^* = (\alpha^*+\beta)^*$ ، $(\alpha+\beta)^* = (\alpha^*+\beta^*)^*$

مثال:

$R = (aa)^*(bb)^* + a(aa)^* b(bb)^*$

$R = (aa)^*(bb)^* + (aa)^* ab(bb)^*$

$R = (aa)^*(\lambda + ab)(bb)^*$

$R = (aa + ab + ba + bb)^*$

$R = \left(a(a+b) + b(a+b)\right)^*$

$R = \left((a+b)(a+b)\right)^*$

Subject :

Year . Month . Date . ( )

اگر α، β، γ عبارات منظم باشند، به اینصورت:

7) $(\alpha+\beta)^* = (\alpha^*\beta^*)^*$

while (needed) do
  select α or β

while (needed) do
  while (needed) do
    select α
  while (needed) do
    select β

8) $\alpha(\beta\alpha)^* = (\alpha\beta)^*\alpha$

9) $(\alpha+\beta)^* = \alpha^*(\beta\alpha^*)^*$ ← عبارت فوق رشته‌ای را می‌تواند تولید کند.

$(\alpha+\beta)^* = \beta^*(\alpha\beta^*)^*$ ← عبارت فوق هم رشته‌ای را می‌تواند تولید کند.

$\alpha^*(\beta\alpha^*)^* = \beta^*(\alpha\beta^*)^*$

✓ برای نشان دادن اینکه دو عبارت منظم معادل نیستند، باید نشان دهیم مثالی هست که در یکی نیست.

مسئله: عبارت منظمی برای زبان زیر بنویسید.

$L = \{w \mid w \in \{0,1\}^*, w \text{ has exactly one pair of consecutive zeros}\}$

برای حل مسئله فوق ابتدا می‌خواهیم عبارت منظمی برای رشته‌هایی بدست آوریم که در آنها هیچ دو صفری پشت سر هم نیایند.

Subject:
Year. Month. Date. ( )

برای این منظور رشته های فرد را به چهار دسته تقسیم می کنیم.

(I) 0 _ 0 ⟵ رشته وسط با ۱ شروع می شود و به ۱ ختم می شود و هیچ دو صفری هم پشت سر هم نیستند

(II) 0 _ 1 ⟵ هر رشته ای که در آن هیچ دو صفری پشت سر هم نباشد را می توان به قطعاتی تقسیم کرد که هر زیر رشته یا ۱۰ است یا ۱.

(III) 1 _ 0

(IV) 1 _ 1

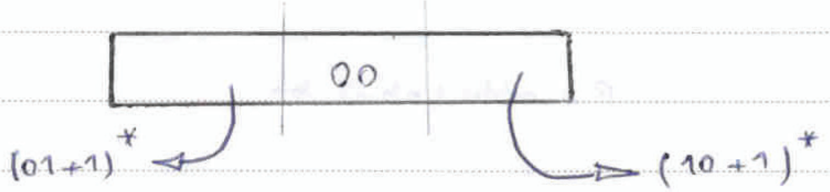
قسمت های I و II را پوشش می دهد

$$R = 0(10+1)^* + (10+1)^* + (01+1)^*$$

قسمت های III و IV را پوشش می دهد

قسمت های II و IV را پوشش می دهد

اکنون برای حل مسئله اصلی بصورت زیر عمل می نماییم



تمرین: یک عبارت منظم برای رشته هایی پیدا کنید که دقیقاً دو صفر پشت سر هم وجود دارد. (سه صفر پشت سر هم قابل قبول است)

Subject:
Year. Month. Date. ( )

***مثال:** یک گرامر نوع سوم خطی از سمت راست داده شده است. گرامرهای نوع سوم خطی از سمت راست، خطی از سمت چپ معادل پیدا کنید.

$S \rightarrow abA$

$A \rightarrow baB$ گرامر خطی از سمت راست ← تغییر در سمت راست عبارات های آمده است.

$B \rightarrow aA \mid bb$

ab baa baa baa baa baa babb

✓ روش تولید کردن های گرامر از اثبات های constructive می باشد.

abb aab aab aab aab aab abb

$S \rightarrow abbA$

$A \rightarrow aaB \mid abb$ $R = abb(aab)^* abb$

$B \rightarrow bA$

abba aba aba aba aba aba bb

$S \rightarrow abbaA$

$A \rightarrow abB \mid bb$ $R = abba(aba)^* bb$

$B \rightarrow aA$

برای ارائه یک گرامر نوع سوم خطی از سمت چپ، رشته بندی بالا را در نظر می گیریم ولی از آخر به اول برای آن گرامر می نویسیم.

$S \rightarrow Abb$, $A \rightarrow Bba \mid abba$, $B \rightarrow Aa$

Subject :

Year . Month . Date . ( )

بدست آوردن NDFA$_\lambda$ معادل RE

می خواهیم برای یک عبارت منظم (Regular Exp) یک NDFA معادل λ عرکات λ پیدا نماییم.

1. $\emptyset$:
2. λ :
3. $a \in \Sigma$:
4.

$L(R_1)$ $L(R_2)$

M_1 M_2

M_1 M_2

$L(R_1 + R_2)$

M_1 M_2

$L(R_1 R_2)$

M_1

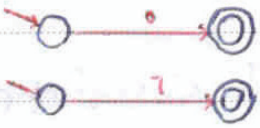
$L(R_1^*)$

Subject:

Year. Month. Date. ( )

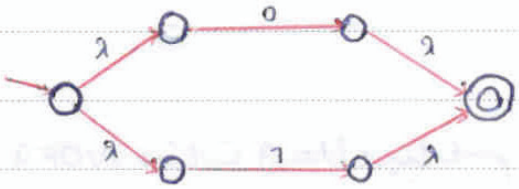
$R = (0+1)^*(10+1)(0+1)^*$:مثال

0 :

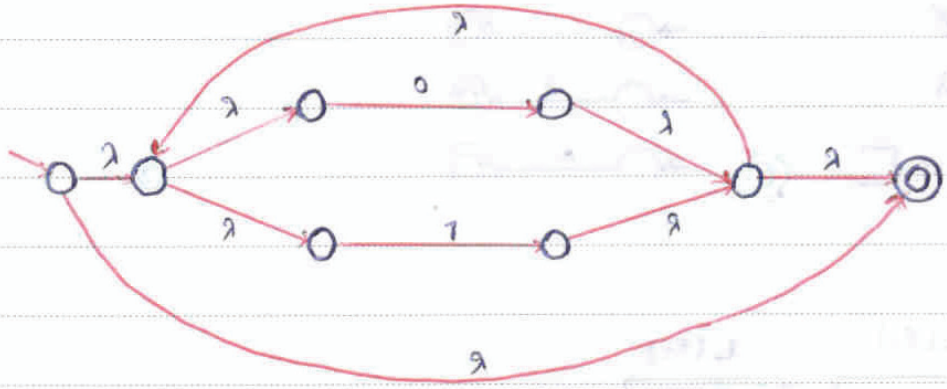


1 :

$(0+1)$:



$(0+1)^*$:



$(10+1)$:

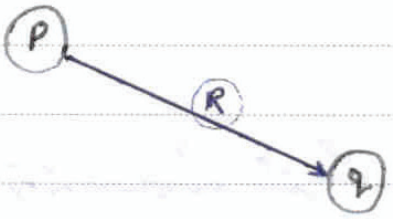
$(0+1)^*(10+1)(0+1)^*$:

Subject:

Year. Month. Date. ( )

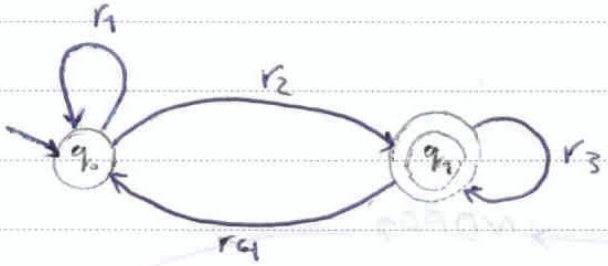
بدست آوردن RE معادل DFA

Generalized Transition Graph:



در گراف انتقال تعمیم یافته، برچسب Arc ها یک عبارت منظم (یک مجموعه از رشته ها) می باشد (اگر بین حالت P با حالم q یک رشته بیابد که عبارت R آن را نمایندگی می کند، آنگاه به حالت q می رویم)

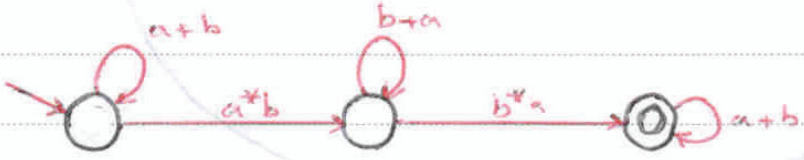
مثال:



$$R = r_1^* r_2 (r_4 r_1^* r_2 + r_3)^*$$

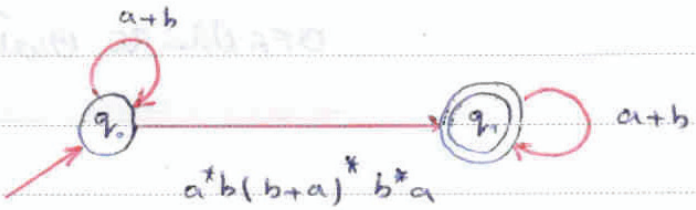
✓ در واقع باید منطق پشت را متوجه شویم که چه رشته هایی را می پذیرد.

مثال:



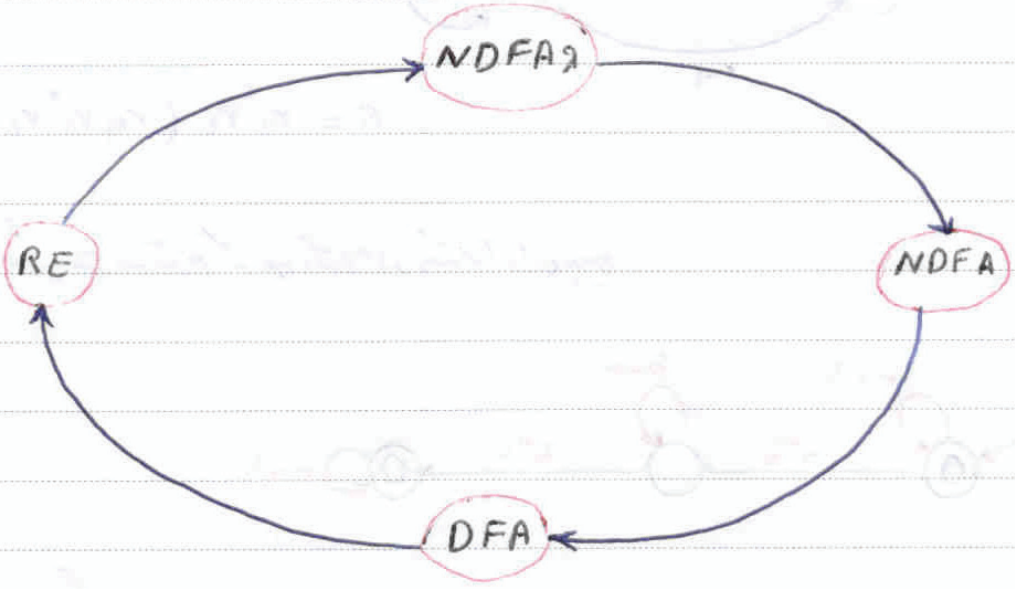
! حالت اولیه را حالت نهایی را نگه می داریم و سایر حالات را حذف می کنیم.

به اینصورت یکسری میروند برای حذف سایر حالات از بین می روند. این حالات حذف شده را ابقاء می کنیم.



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

✓ اگر تعداد حالاتی که حذف می‌شوند بیشتر باشد، باید تک تک Node ها را حذف کرد در هر مرحله میانی حذف شده را ایجاد نمود.

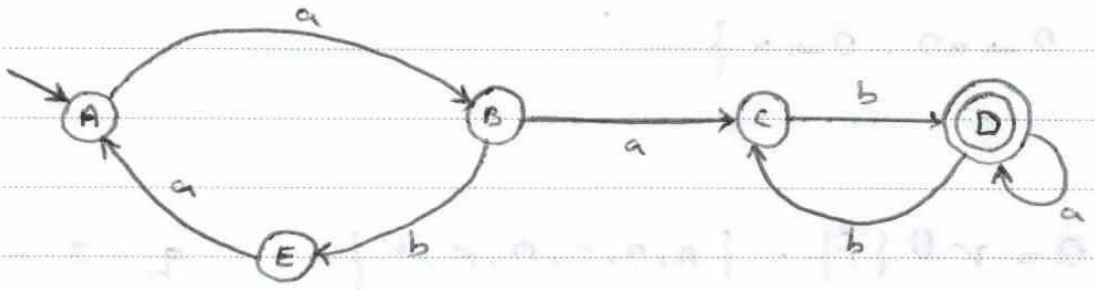


تبدیلات گرامرهای منظم (RG)

NDFA = (Q , Σ , δ , q_0 , F) accepting L

RG = (V, T, P, S) defing L

مثال:



V = Q = {A, B, C, D, E} , S = q_0 = A , T = Σ

P = ?

- Add $q \to ap$ to P if $\delta(q,a) =$
- Add $q \to ap$ to P if $\delta(q,a) =$
  $q \to a$

P = { A → aB , B → bE , E → aA , B → aC
C → bD , C → b
D → bC
D → aD , D → a }

Subject:

Year. Month. Date. ( )

RG = (V , T , P , S) defing L

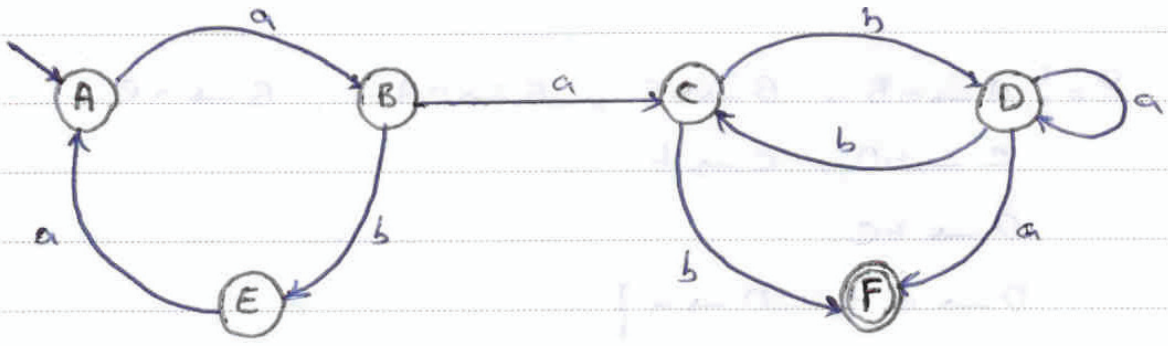
NDFA = (Q , Σ , δ , q_0 , F) accepting L

P = { A → aB , B → bE , E → aA , B → aC
C → bD , C → b
D → bC
D → aD , D → a }

Q = V ∪ { F } = { A, B, C, D, E, f } , q_0 = S = A

F = { F }

δ : - Create δ(q, a) = p if q → ap is in P
- Create δ(q, a) = f if q → a is in P



62

Subject:

Year. Month. Date. ( )

مثال: یک گرامر منظم برای عبارت $R = aa^*b$ بنویسید.

$S \rightarrow aA$

$A \rightarrow aA \mid b$

مثال: یک گرامر نوع سوم برای عبارت $R = aa^*b + a$ بنویسید.

$S \rightarrow a \mid aA$

$A \rightarrow aA \mid b$

* مثال: یک گرامر برای عبارت $R = a(ba^*b + a)^*$ ارائه دهید.

$S \rightarrow aA$

$A \rightarrow aA \mid bB \mid \lambda$

$B \rightarrow aB \mid bA$

* گرامرهای نوع سوم ← $\begin{cases} A \rightarrow xB \\ A \rightarrow x \end{cases}$ یا $\begin{cases} A \rightarrow Bx \\ A \rightarrow x \end{cases}$ $\quad x \in T^+$

✓ در واقع در این تعریف جدید به جای یک علامت خاص (a) یک رشته (x) داریم.

در واقع با استفاده از بازگشت می‌توانیم تعداد دلخواهی از رشته را بصورت مجموعه‌ای از اعداد بنویسیم که در آن تعدادی علامت خاص می‌باشند.

$A \rightarrow x_1 x_2 \cdots x_n B$

$A \rightarrow x_1 X_1$ $\qquad$ $X_{n-2} \rightarrow x_{n-1} X_{n-1}$

$X_1 \rightarrow x_2 X_2$ $\qquad$ $X_{n-1} \rightarrow x_n B$

Subject:

Year. Month. Date. ( )

Subject:
Year. Month. Date. ( )

«فصل چهارم»

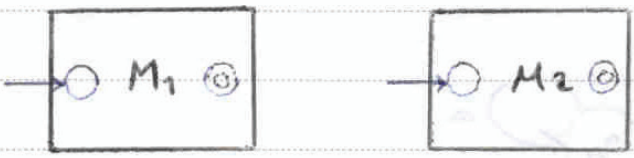
«خصوصیات زبانهای منظم - خصوصیات بسیاری زبان های نوع سوم»

تذکر: خانواده زبان های نوع سوم تحت عملگرهای U ، $\circ$ ، $*$ بسته است.

اگر L_1 و L_2 زبان های نوع سوم باشند، در اینصورت $L_1 \cup L_2$ ، $L_1 L_2$ ، L_1^* از نوع سوم است.

Constructive proof.

L_1 L_2



✓ چون L_1 و L_2 هردو از نوع زبان های نوع سوم هستند لذا یک پذیرنده برای هرکدام وجود دارد. همانطور که دیدیم برای اجتماع این دو زبان یک پذیرنده میسازیم لذا اجتماع این دو زبان نسبت به عملگر U بسته است.

✓ همچنین بطور مشابه می توان با استفاده از دو پذیرنده فوق، پذیرنده های جدیدی برای عملگرهای $\circ$ و $*$ نیز طراحی نمود لذا L_1 و L_2 نسبت به عملگرهای $\circ$ و $*$ نیز بسته می باشند.

! تاکنون 5 نوع تعریف برای زبان های نوع سوم داریم (DFA ، NDFA ، NDFA$_\lambda$ ، RE ، RG) که به عنوان ابزاری جهت اثبات قضایا می توانند بکار گرفته شوند.

Subject:

Year. Month. Date. ( )

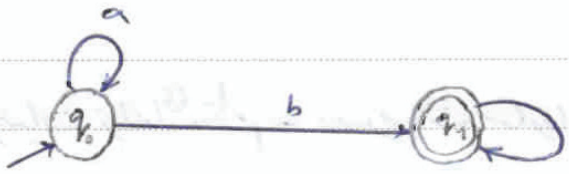
قضیه: خانواده زبان‌های نوع سوم تحت عمگر متمم بسته است.

متمم یک زبان نوع سوم، یک زبان نوع سوم است.

اگر L نوع سوم باشد، به اینصورت $\bar{L}$ متمم آن نوع سوم است.

Given: $FA = (Q, \Sigma, \delta, q_0, F)$ accepting L

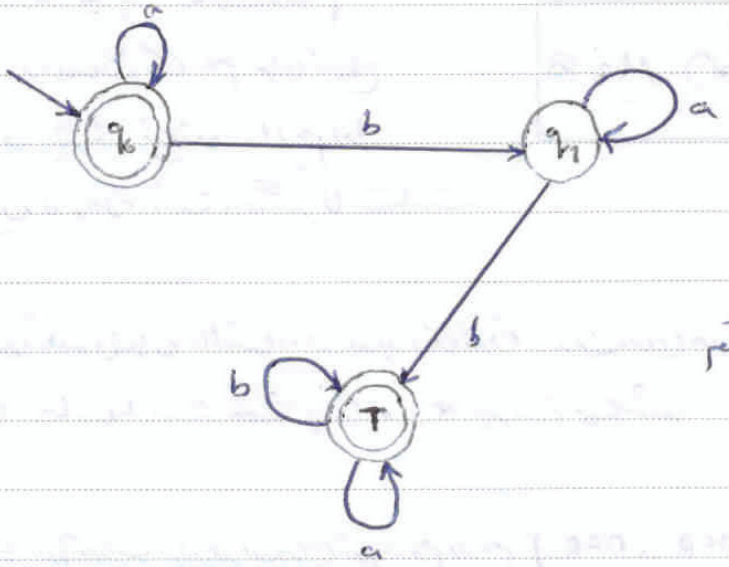
Find: $\overline{FA} = (\bar{Q}, \Sigma, \bar{\delta}, \bar{q}_0, \bar{F})$ accepting $\bar{L}$



$$\bar{Q} = Q \quad , \quad \bar{q}_0 = q_0 \quad , \quad \bar{F} = Q - F \quad , \quad \bar{\delta} = \delta$$

! در حالت q_1 به ازای b Transition نداریم. لذا بدون آنکه ماشین زبان پذیرفته را تغییر دهیم، حالت Trap را به آن اضافه می‌کنیم.

(حالت‌های تهی و غیرتهی متمم عرض می‌شود (حالات غیرتهایی ماشین اول، حالات تهایی ماشین قبلی است)



□ بنابراین یک پذیرنده متمم برای زبان متمم ارائه دادیم.

Subject :

Year . Month . Date . ( )

قضیه : خانواده زبان‌های نوع سوم تحت عملگر $\cap$ بسته است.

اثبات با استفاده از : De'Morgan's Law ←

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

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

نشان می‌دهیم سمت راست عبارت ندت هست
در قضیه قبلی از نوع زبان‌های نوع سوم است.
(یعنی این عبارت یک ماشین ساخت برای پذیرنده اشتراک دو زبان می‌دهد)

اثبات به روش دیگر: ←

Given : $FA_1 = (Q_1, \Sigma_1, \delta_1, q_0^1, F_1)$ accepting L_1

$FA_2 = (Q_2, \Sigma_2, \delta_2, q_0^2, F_2)$ accepting L_2

Find : $FA = (Q, \Sigma, \delta, q_0, F)$ accepting $L_1 \cap L_2$

L_1 :

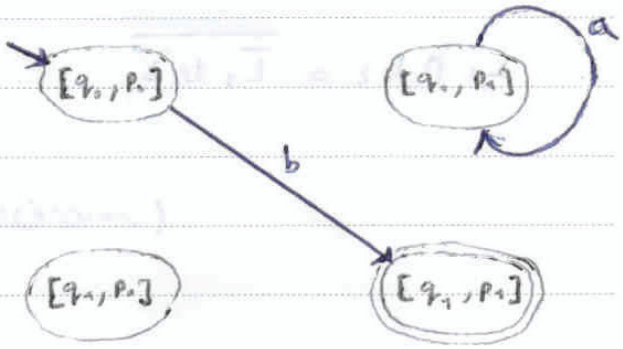
L_2 :

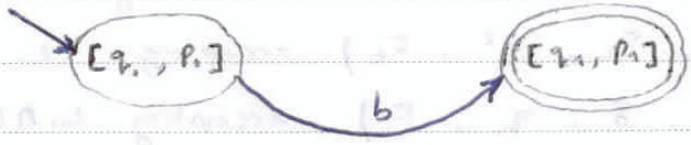
$$Q = Q_1 \times Q_2 = \{ [q_0, P_0], [q_0, P_1], [q_1, P_0], [q_1, P_1] \}$$

$$q_0 = [q_0, P_0] \quad , \quad F = \{ [r, k] \mid r \in F_1, k \in F_2 \}$$
$$= \{ [q_1, P_1] \}$$

δ: $\delta([r,k],a) = [r',k']$ iff $\delta_1(r,a) = r'$ and $\delta_2(k,a) = k'$

δ	a	b
$[q_0, p_0]$	–	$[q_1, p_1]$
$[q_0, p_1]$	$[q_0, p_1]$	–
$[q_1, p_0]$	–	–
$[q_2, p_1]$	–	–





تمرین: برنامه بنویسید که FA1, FA2 را بگیرد، تعدادی رشته هم بگیرد و شفاف کند که آیا ماشین اشتراک این دو زبان رشته‌ها را می‌پذیرد یا نه؟ (باندفی DFA)

$L(FA_1) \cap L(FA_2)$

در واقع در ماشین بصورت موازی کار می‌کنند و سپس یک ناظر در پایان شفاف می‌کند ماشین در کدام حالت است و نهایتا نتیجه‌گیری می‌کند.

Subject:

Year. Month. Date. ( )

✓ اگر L_1 و L_2 زبان‌های نوع سوم باشند، در اینصورت $L_1 - L_2$ نیز یک زبان نوع سوم است.

هموومورفیزم

Homomorphism

$\Sigma_1 = \{a,b\}$, $\Sigma_2 = \{0,1\}$

$h: \Sigma_1 \to \Sigma_2^+$ ⟿ «مانند یک نگاشت»

قضیه:

✓ خانواده زبان‌های نوع سوم تحت هموومورفیزم بسته است.

اگر L نوع سوم باشد، $h(L)$ نیز نوع سوم است.

$h(L) = \{h(w) \mid w \in L\}$

$\Sigma_1 = \{a,b\}$, $\Sigma_2 = \{0,1\}$ ⟿ $h(a) = 00$, $h(b) = 1$

$L = \{a^n b^m \mid n,m \geq 0\}$ ⟿ $h(L) = \{(00)^n (1)^m \mid n,m \geq 0\}$

$S \to abA$, $A \to aaA \mid b$ ⟿ $S \to h(a)h(b)A$, $A \to h(a)h(a)A \mid h(b)$

$S \to 001A$, $A \to 0000A \mid 1$

$h(aba) = h(a)h(b)h(a) = 00100$

✓ خانواده زبان‌های نوع سوم تحت اجتماع متناهی و اشتراک گسسته بسته است. یعنی اگر $L_1, L_2, \ldots, L_n$ زبان‌های نوع سوم باشند، $\bigcup_{i=1}^{n} L_i$, $\bigcap_{i=1}^{n} L_i$ نوع سوم می‌باشند. (112,5 P)

(۱۱۲، 6P) تمرین ترینه — "Symmetric diffrence"

$L_1 \ominus L_2 = \{x \mid x \in L_1 \text{ or } x \in L_2 \text{ but not in both}\}$

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

زبان‌های نوع سوم نسبت به عملگر تفریق قرینه نیز بسته می‌باشند.

عملگرهای Right Quotient, Left Quotient (تمرین کتاب)

اگر L یک زبان منظم است در این صورت زبان $L_1 = \{uv \mid u \in L,\ |v| = 2\}$ نیز یک زبان منظم است. $u, v \in \{a, b\}$

زبان L که فرض یک زبان نوع سوم و زبان محدود فوق نیز یک زبان نوع سوم است پس Concatination آن این دو زبان نیز سوم است. $L_1 = L \cdot \{ab \mid bb \mid ba \mid aa\}$

(۱۱۲، 13) اگر L نوع سوم است در این صورت زبان $L_1 = \{uv \mid u \in L,\ v \in L^R\}$ نیز یک زبان نوع سوم است. $(L_1 = L \cdot L^R)$

! خانواده زبان‌های نوع سوم تحت عملگر معکوس بسته است. معکوس یک زبان نوع سوم، خودش یک زبان نوع سوم است.
$L^R = \{x^R \mid x \in L\}$

Subject:

Year. Month. Date. ( )

✓ از روی ماشین زبان L می‌توان ماشین زبان معکوس L^R را ساخت. بدین صورت که حالات نهایی را اولیه و عکس می‌کنیم و نیز Transition های گراف آن را نیز معکوس می‌کنیم.

✓ از روی گرامر یک زبان نوع سوم مثل L نیز می‌توان گرامر زبان معکوس آن (L^R) را نیز ساخت.

L:

$S \rightarrow abA$

$A \rightarrow baA \mid bba$

$\rightsquigarrow$

L^R:

$S \rightarrow Aba$

$A \rightarrow Aab \mid abb$

✓ $Head(L) = \{ x \mid xy \in L \text{ for some } y \in \Sigma^* \}$ (113,6P)

نوعی prefix های سازنده اعضای زبان

$L = \{abb, bab, bb\} \rightsquigarrow Head(L) = \{\lambda, a, ab, abb, b, ba, bab, b\}$

! اگر یک زبان نوع سوم باشد، در اینصورت زبان Head(L) نیز از نوع سوم است.

✓ اگر ماشین زبان L را داشته باشیم، می‌توانیم ماشین زبان head(L) را نیز بدست آوریم. کافیست تمام حالات نهایی را حالت غیرنهایی کنیم.

مسائل قابل تصمیم گیری ممکن زبان های نوع سوم

★ یک مثال از مسائل تصمیم گیری دیمی اینکه بتوانیم الگوریتمی برای حل آن ارائه دهیم.

- مسئله عضویت برای زبان های نوع سوم قابل تصمیم گیری است. Membership problem
(طبق تعریف یک ماشین پذیرنده برای زبان نوع سوم وجود دارد)

- این مسئله که یک زبان نوع سوم تهی باشد یا نه، قابل تصمیم گیری است.

- این مسئله که یک زبان نوع سوم تهی باشد یا نه، قابل تصمیم گیری است.

- مسئله تساوی در زبان نوع سوم قابل تصمیم گیری است.

✓ در ماشین یک زبان نوع سوم اگر در مسیرهای ماشین حالت اولیه و حالت نهایی cycle وجود داشته باشد زبان نامتناهی می باشد.

? مسئله تساوی: اگر دو زبان متناهی باشند می توان در زبان متناهی تساوی دو زبان را بررسی کرد. $L_1 \stackrel{?}{=} L_2$

$\eta \in L_1 \Rightarrow \eta \in L_2$ $\quad$ $\eta \in L_2 \Rightarrow \eta \in L_1$

Subject:

Year. Month. Date. ( )

$L = (L_1 \cap \bar{L}_2) \cup (\bar{L}_1 \cap L_2)$ ⟵ زبان ساخت این زبان با الگوریتم فوق تناهی می باشد.

if $L = \emptyset$ then $L_1 = L_2$

else $L_1 \neq L_2$

قضیه pumping (تزریقی)

فرض کنید L یک زبان نوع سوم تناهی باشد، در اینصورت یک عدد مثبت صحیح m وجود دارد بطوریکه برای هر $w \in L$ ، $|w| \geq m$ ، آن را میتوان بصورت xyz تجزیه کرد (شرایط تجزیه: $|xy| \leq m$, $|y| \geq 1$) بطوریکه:

$$xy^i z \in L$$

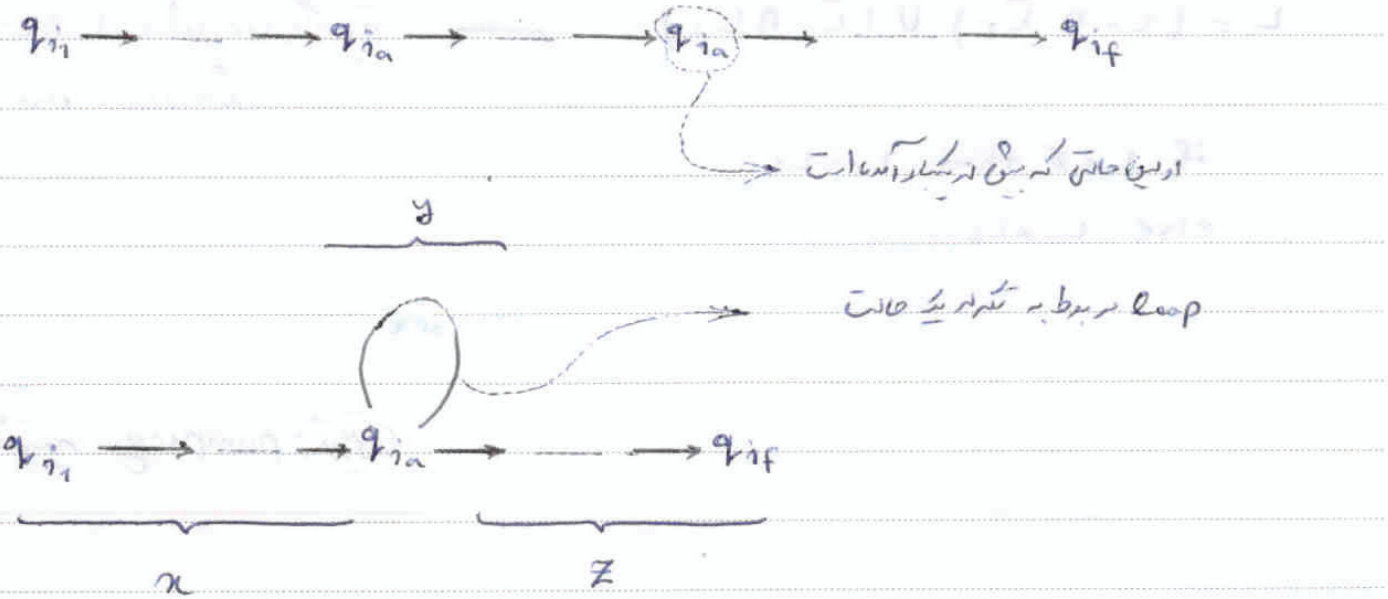
⟵ $i = 0, 1, \ldots$ برای تعداد دفعات

اثبات: چون فرض شده چون L یک زبان نوع سوم است، بنابراین دارای یک پذیرنده می باشد.

DFA ⟵ $\{q_1, q_2, \ldots, q_m\}$ $|w| \geq m$

$q_{i_0} \rightarrow q_{i_1} \rightarrow q_{i_2} \rightarrow \cdots \rightarrow q_{i_f}$ ⟵ مسیری که طی می شود تا یک رشته پذیرفته شود.

- q_i ها عضو مجموعه $q_1 \ldots q_m$ می باشند.
- اگر رشته ای بیش از m طول داشته باشد، چون اعمال لانه کبوتری در مسیر بالا حالات یا حالاتی وجود دارند که بیش از یکبار ظاهر شده اند.



✓ با استفاده از لم عکس قضیه فوق، اگر بتوانیم رشته‌ای از یک زبان را پیدا کنیم که به هیچ صورتی نتوان آن را مانند بالا تجزیه کرد، به آن صورت زبان فوق یک زبان نوع سوم نمی‌باشد.

مثال: برای کاربرد قضیه pumping و اثبات این که زبان مورد بررسی نوع سوم نیست، از یک بازی استفاده می‌کنیم.

$L = \{a^n b^n \mid n > 0\}$

✓ در این بازی دو نفر یک نوبت بازی انجام می‌دهند:

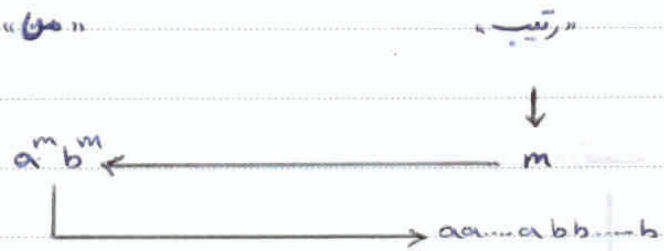
ابتدا رقیب یک m انتخاب می‌کند. سپس من از بین نامتناهی رشته بزرگتر مساوی m یک رشته انتخاب می‌کنم و به رقیب می‌دهم. رقیب طبق قضیه pumping یک تجزیه از رشته داده شده می‌دهد و آن را می‌شکند.

اگر من بتوانم برای تمام شکاندن‌های رقیب یک زوج رشته بدهم که رشته آن عضو زبان نباشند، من برنده‌ام.

رقیب: رقیب می‌خواهد از کار من جلوگیری کند.

من: من می‌خواهم اثبات کنم که زبان نوع سوم نیست

74



✓ باید یک تابع فرانسم کنیم که بتوان تابع بعدی حرکات ترتیب باشد

✓ من باید حرکات مرتبی را انجام دهد تا برای ترتیب تعداد حالات زیادی پیش بیاید.

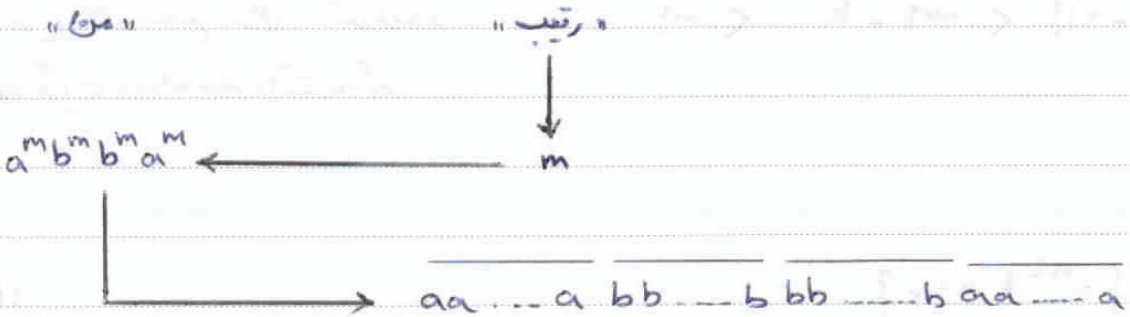
← انتخاب $i=0$

$$a^{k'}(a^k)^{i}a^{k''}b^m \longleftarrow a^{k'}a^k a^{k''}b^m \quad (1 \le k \le m)$$

! باید به ازای تمام حالات نشان داده شود که تجزیه امکان پذیر نیست در غیر اینصورت به اثبات، ممکن است یک زبان نوع سوم، یک زبان غیر نوع سوم تشخیص داده شود.

مثال:

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

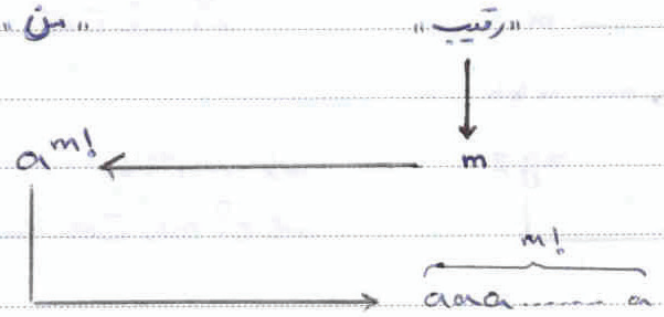


$$x = a^k \quad , \quad y = a^{k'} \quad , \quad Z = a^{k''} b^{2m} a^m$$

$$a^k (a^{k'})^i a^{k''} b^{2m} a^m$$

مثال:

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



$x = a^{k'}$ $y = a^{k}$, $z = a^{k''}$ $(1 \leq k \leq m)$

$a^{k'}(a^{k})^{i}a^{k''}$ ⟶ به ازای $i=0$ اثبات می‌کنیم که تغییر درست نیست

$i=0$ ⟶ $a^{m!-k}$ اکنون ثابت می‌کنیم $m!-k$ نمی‌تواند فاکتوریل هیچ عددی نیست.

$(m-1)! < m! - k < m!$ ⟶ بین دو فاکتوریل متوالی، نه فاکتوریل عدد صحیحی دیگری نمی‌تواند قرار داشته باشد.

تمرین:

$L = \{ a^{n^2} \mid n \geq 0 \}$

$L = \{ a^{n} \mid n \text{ is not a prime} \}$

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

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

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

$L = \{ a^n \mid n \text{ is a prime} \}$

$L = \{ a^{2^k} \mid k \geq 0 \}$

Subject:

Year. Month. Date. ( )

۶.۱۱۶ - این مسئله که یک زبان نوع سوم زیر مجموعه‌ای از یک زبان نوع سوم دیگر است، قابل تصمیم‌گیری است.

$L_1 \overset{?}{\subseteq} L_2$

ساخت این تمت در زبان تناهی قابل اثبات است. $L_3 = L_1 \cap L_2$

if $L_3 = L_1$ Then $L_1 \subseteq L_2$

else $L_2 \subseteq L_1$

مثال: این مسئله که $\lambda \in L$ است، قابل تصمیم‌گیری است.

یک DFA برای زبان می‌سازیم. اگر حالت اولیه، حالت نهایی بگیرد، λ عضو زبان است.

✓ زبان پالیندروم ← زبانی که رشته‌های آن پالیندروم است یعنی رشته‌های آن قرینه هستند.

مثال: این مسئله که یک زبان پالیندروم است یا نیست، قابل تصمیم‌گیری است.

$L_3 = L^R$

if $L_3 = L$ then L is palindrom

else L is not palindrom

Subject :

Year . Month . Date . ( )

« تئوری زبان های نوع دوم »

گرامرهای نوع دوم :

$A \longrightarrow \alpha$

$A \in V$

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

- اشتقاق از چپ (leftmost derivation)
- اشتقاق از راست (Rightmost derivation)
- درخت اشتقاق (derivation tree)
- ابهام (ambiguity)

مثال : $S \longrightarrow AB \;,\; A \longrightarrow aAla \;,\; B \longrightarrow bBla$

اشتقاق از چپ

$S \Rightarrow AB$

$\Rightarrow aAB$

$\Rightarrow aaAB$ ⟵ LMD

$\Rightarrow aaaB$

$\Rightarrow^* aaaba$

اشتقاق از راست

$S \Rightarrow AB$

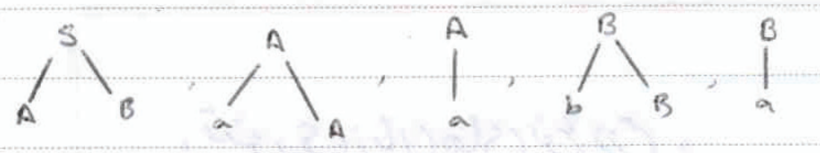
$\Rightarrow AbB$

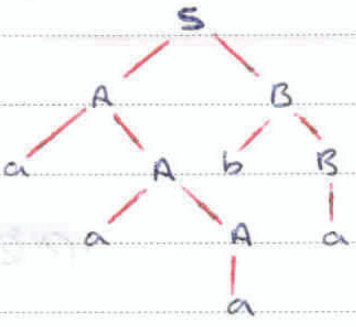
$\Rightarrow Aba$ ⟵ RMD

$\Rightarrow aAba$

$\Rightarrow^* aaaba$

درخت اشتقاق:

P = {  }



✓ اگر برگ‌ها را از سمت چپ به راست بخوانیم، درخت اشتقاق آن رشته بدست می‌آید.

رشته aaaba ←

- فرآیند بررسی « parsing » هم گفته می‌شود ← «parse tree»

* ابهام: یک گرامر نوع دوم مبهم است اگر بتواند حداقل یک جمله تولید کند که آن جمله دارای حداقل دو درخت اشتقاق (دو LMD یا دو RMD) باشد.

مثال: نشان دهید گرامر زیر مبهم است.

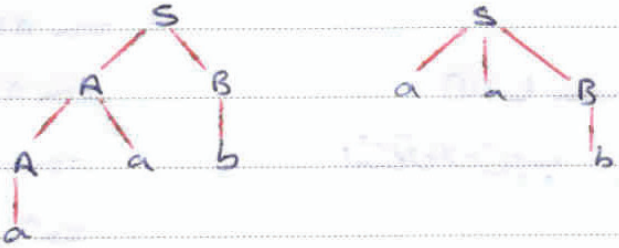
$S \rightarrow AB \mid aaB$

$A \rightarrow a \mid Aa$

$B \rightarrow b$

بنابراین باید به گونه‌ای جمله‌ای با خصوصیات بالا پیدا کنیم.

aab



Subject:

Year. Month. Date. ( )

مثال: یک گرامر عنید مجموعه معادل برای گرامر زیر پیدا کنید. (حذف ابهام)

$S \to AB \mid aaB$

$A \to a \mid Aa$

$B \to b$

الگوریتم خاصی وجود ندارد! باید به گونه‌ای ابهام گرامر را پیدا کنیم و سپس آن را حذف کنیم (Remove ambiguity)

(I) $S \Rightarrow AB$

$\Rightarrow AaB$

$\Rightarrow aaB$

$\Rightarrow aab$

(II) $S \Rightarrow aaB$

$\Rightarrow aab$

⟿ LMD

(I) $S \Rightarrow AB$

$\Rightarrow Ab$

$\Rightarrow Aab$

$\Rightarrow aab$

(II) $S \Rightarrow aaB$

$\Rightarrow aab$

⟿ RMD

آنرون به زبان یک گرامر معادل هستیم که همان رشته‌ها را تولید کند ولی مبهم نباشد

$S \to AB$

$A \to a \mid Aa$

$B \to b$

با حذف قسمت $S \to aaB$ قدرت تولید گرامر عوض نمی‌شود و همان رشته‌ها تولید می‌شوند ولی دیگر ابهام به وجود نمی‌آید.

✓ اگر در مورد یک زبان نوع ابهام وجود داشته باشد، برای رشته‌هایی از آن زبان می‌توان بیش از یک مشتق راست‌ترین رسم نمود که این مشتقات معانی متفاوتی را از این رشته استنباط کرد.

مثال:

E → I (expression)

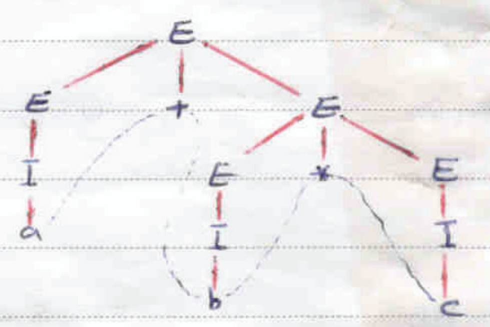
E → E + E

E → E * E

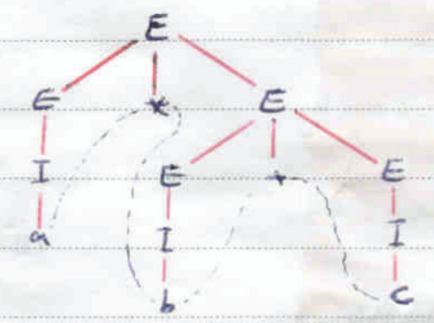
E → (E)

I → a | b | c

(I)



(II)



در واقع ما می خواهیم یک رشته، درخت آن را بسازیم و سپس کد کنیم و اگر برای یک رشته دو درخت وجود داشته باشد، دو کد متفاوت تولید خواهد شد. (کدها از زیر درخت های پایین تر به بالاتر ساخته می شوند)

(I) LOAD B , MUL C , ADD A

(II) LOAD B , ADD C , MUL A

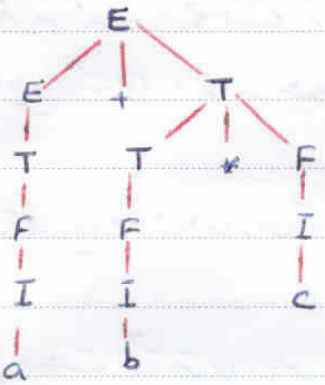
! راه حل مشکل تعیین دادن اولویت به عملگرها می باشد.

E → T	F → (E)	E ⇝ expression
T → F	I → a | b | c	T ⇝ Term
F → I		F ⇝ Factor
E → E + T		I ⇝ Identifier
T → T * F		

Subject:

Year. Month. Date. ( )

گرامر فوق یک گرامر غیر مبهم است. برای مثال برای رشته a + b * c درخت تجزیه را رسم کنید.

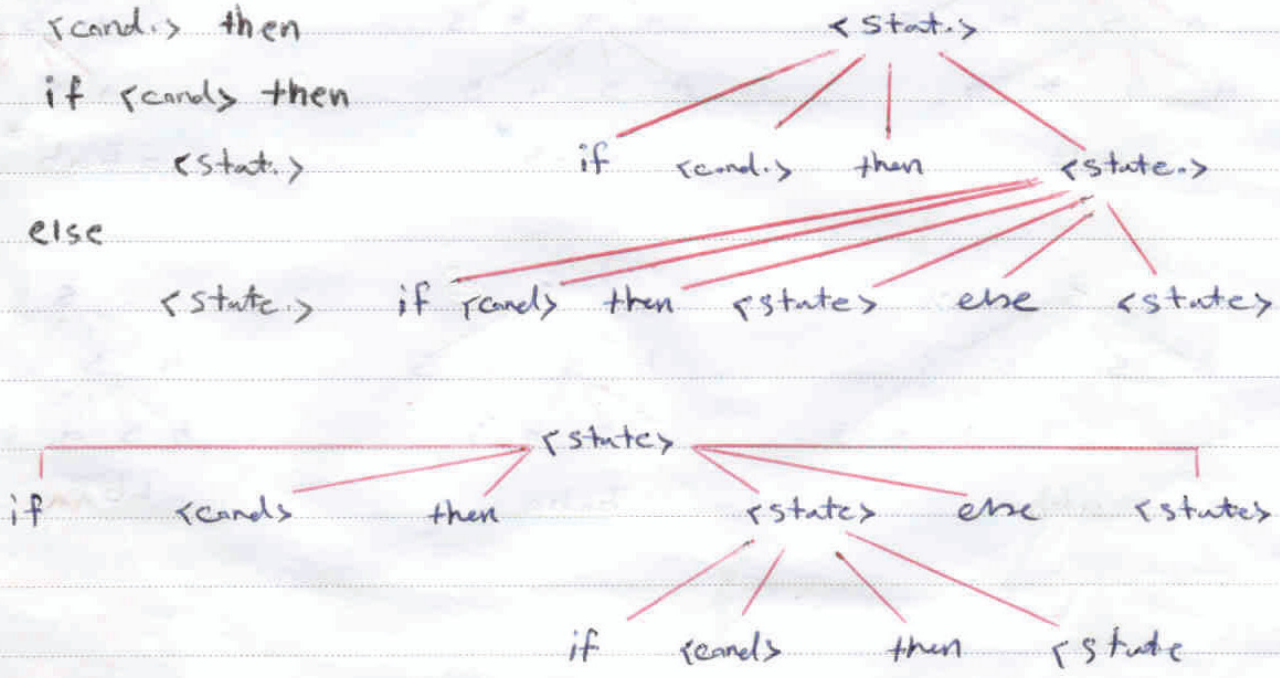


در واقع این گرامر الزام می‌کند که عملگر با اولویت بیشتر (در اینجا ضرب) در زیر درخت‌های پایین‌تر تولید بگیرد.

مثال: گرامر جملات شرطی:

<statement> → if <condition> then <statement> |
if <condition> then <statement> else <statement> |
<other statement>

if <cond.> then
if <cond> then
<stat.>
else
<state.>



83 ← ادامه‌ی if ها را خودتان بکشید

Subject:
Year. Month. Date. ( )

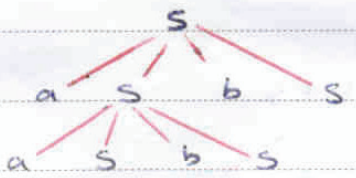
مثال: نشان دهید گرامر زیر مبهم است. $S \to aSbS \mid bSaS \mid \lambda$

باید رشته‌ای پیدا کنیم که دو درخت اشتقاق داشته باشد.
برای جستجوی چنین رشته‌ای، ابتدا در مجموعه رشته‌های با طول کوتاه‌تر به دنبال رشته می‌گردیم.

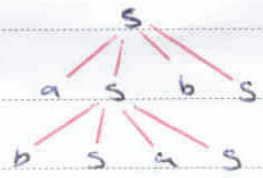
کوتاه‌ترین جملات این زبان که دو درخت اشتقاق دارند. $\{ab, ba\}$

اکنون به سراغ جملات بزرگتر می‌رویم. دومین کوتاه‌ترین جملات دارای طول چهار هستند.
یک روش این است که تعداد درخت‌هایی که منجر به رشته‌هایی با طول چهار می‌شوند را لیست کنیم و اگر جمله‌ای دارای دو درخت اشتقاق باشد، خود را نشان می‌دهد.

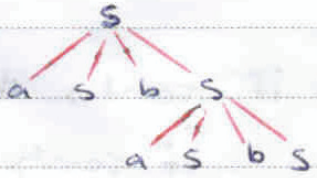
اکنون می‌خواهیم الگوریتمی پیدا کنیم که تعداد درخت‌های تولید کننده رشته‌هایی با طول چهار را تولید کند.



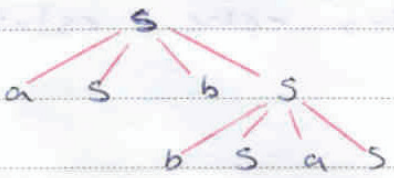
aabb



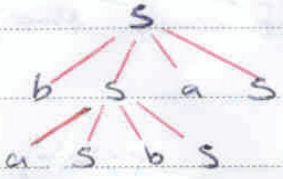
abab*



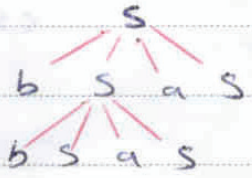
abab*



abba



baba



bbaa

← روش دیگر برای پیدا کردن جملاتی با دو درخت اشتقاق، این است که با استفاده از شمارش نشان دهیم چنین جملاتی وجود دارند.

گرامر فوق مربوط به زبانی است که رشته‌هایی با تعداد مساوی a و b دارد.

$\frac{4!}{2! \times 2!} = 6$ → تعداد رشته‌ها با طول چهار

پس چون 6 رشته با طول چهار داریم ولی 8 درخت اشتقاق داریم که رشته‌هایی با طول چهار تولید می‌کنند، لذا دو رشته وجود دارند که دارای دو درخت اشتقاق هستند.
(استفاده از اصل لانه کبوتری)

سوال: چند رشته با طول 6 وجود دارند که دارای بیش از یک درخت اشتقاق می‌باشند.

تعداد درخت‌های اشتقاق برای جمله‌های با طول 3 → $8 \times 3 \times 2 = 48$

- تعداد S هایی که می‌توانند باز شوند
- تعداد حالاتی که بعدی می‌تواند باز شود
- تعداد درخت‌های با طول 2

تعداد رشته‌هایی با طول 3 در این زبان → $\frac{6!}{3!\,3!} = 20$

طبق اصل لانه کبوتری: هر 20 رشته با طول 3، دارای بیش از یک درخت اشتقاق هستند.

Subject:

Year. Month. Date. ( )

Subject :

Year . Month . Date . ( )

★ متغیر قابل استفاده (useful) : متغیر A قابل استفاده است اگر:

(I) $A \overset{*}{\Rightarrow} w \quad , \quad w \in T^*$

(II) $S \overset{*}{\Rightarrow} \alpha A \beta \quad , \quad \alpha, \beta \in (V \cup T)$

قاعده قابل استفاده: قاعده‌ای است که تمام متغیرهای حاضر در آن قابل استفاده می‌باشند.

قاعده بدون استفاده: قاعده‌ای است که حداقل یک متغیر در آن بدون استفاده است.

پیدا کردن متغیرهای قابل استفاده

input : $G = (V, T, P, S)$ ← متغیرهای سمت راست قاعده اول

output : Set of all variables X such that $X \overset{*}{\Rightarrow} w \quad , \quad w \in T^*$

$OLDV = \emptyset$

$NEWV = \{ X \mid X \rightarrow w \quad , \quad w \in T^* \}$

while $(OLDV \neq NEWV)$ do

Begin

$OLDV \leftarrow NEWV$

$NEWV \leftarrow OLDV \cup \{ X \mid X \rightarrow \alpha \text{ for some } \alpha \in (T \cup OLDV)^* \}$

End

NEWV is the set of all variables X, such that $X \overset{*}{\Rightarrow} w$, $w \in T^*$

Subject:

Year. Month. Date. ( )

مثال:

$S \rightarrow a \mid aA \mid B \mid C$

$A \rightarrow aB \mid \lambda$

$B \rightarrow Aa$

$C \rightarrow cCD$

$D \rightarrow ddd$

$OLDV = \{\}$, $NEWV = \{S, A, D\}$

ورود loop اول:

$OLDV = \{S, A, D\}$, $NEWV = \{S, A, B, D\}$

$OLDV = \{S, A, B, D\}$, $NEWV = \{S, A, B, D\}$

loop پایان: متغیرهای سمت راست

$\{S, A, B, D\}$ ⟵ «متغیرهای مفید»

* الگوریتم دوم: متغیرهای مفید از قبله مهم

$OLDV = \{\}$

$NEWV = \{S\}$

while $(OLDV \neq NEWV)$ do

Begin

$OLDV \leftarrow NEWV$

$NEWV \leftarrow OLDV \cup \{X \mid A \rightarrow \alpha, A \in OLDV \text{ and } X \text{ appears in } \alpha\}$

End

▷ NEWV is the set of all variables.....

Subject :

Year. Month. Date. ( )

مثال:

$S \to a \mid aA \mid BC$

$A \to aB \mid \lambda$

$B \to Aa$

$C \to cCD$

$D \to dold$

OLDV	NEWV
$\{\}$	$\{S\}$
$\{S\}$	$\{S\} \cup \{A, B, C\}$
$\{S, A, B, C\}$	$\{S, A, B, C\} \cup \{D\}$
$\{S, A, B, C, D\}$	$\{S, A, B, C, D\} \cup \{\}$

! برای اینکه متغیرهای مفید از دور در جنبه اول و دوم را پیدا کنیم، می‌توانیم ابتدا متغیرهای مفید از جنبه اول و سپس متغیرهای مفید از جنبه دوم را پیدا کنیم (حالت عکس جواب متفاوتی خواهد داشت)

مثال: $S \to a \mid aA \mid BC$, $A \to aB \mid \lambda$, $B \to Aa$, $C \to cCD$, $D \leftarrow dold$

✓ I: متغیرهای مفید از جنبه اول ← $\begin{cases} S \to a \mid aA \mid B \\ A \to aB \mid \lambda \end{cases}$, $B \to Aa$, $\begin{cases} D \to dold \end{cases}$ ← متغیرهای مفید از جنبه دوم ← $\{S, A, B\}$

✗ II: متغیرهای مفید از جنبه دوم ← $\begin{cases} S \to a \mid aA \mid BC \\ A \to aB \mid \lambda \\ B \to Aa \end{cases}$, $\begin{cases} C \to cCD \\ D \to dold \end{cases}$ ← متغیرهای مفید از جنبه اول ← $\{S, A, B, D\}$

89

← لذا ترتیب صحیح پیدا کردن متغیرهای مفید از جنبه اول و سپس جنبه دوم است.

Subject:
Year. Month. Date. ( )

★ قواعد واحد (unit production): قواعد دارای تولیدی بصورت $A \longrightarrow B$ هستند

مثال (تمرین): «حذف قواعد واحد»

$S \longrightarrow Aa \mid B$

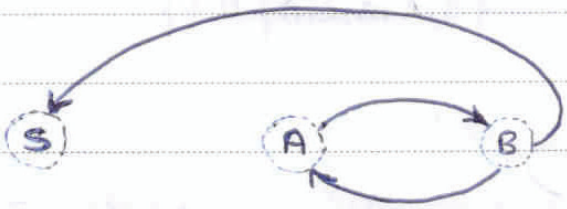
$B \longrightarrow A \mid bb$

$A \longrightarrow a \mid bc \mid B$

قواعد غیر واحد: $S \longrightarrow Aa$, $B \longrightarrow bb$, $A \longrightarrow a \mid bc$

قواعد واحد: $S \longrightarrow B$, $B \longrightarrow A$, $A \longrightarrow B$

سپ گراف وابستگی را رسم کنیم



$S \overset{*}{\Rightarrow} A$

$S \overset{*}{\Rightarrow} B$

$A \overset{*}{\Rightarrow} B$

$B \overset{*}{\Rightarrow} A$

سپ تولید را مجدداً رسم می‌کنیم.

$S \longrightarrow Aa \mid a \mid bc \mid bb$

$B \longrightarrow bb \mid a \mid bc$

$A \longrightarrow a \mid bc \mid bb$

Subject :

Year . Month . Date . ( )

حذف قواعد λ

متغیر پوچ (Nullable) : متغیر A، پوچ (Nullable) است اگر : $A \overset{*}{\Rightarrow} \lambda$

الگوریتم پیدا کردن متغیرهای پوچ :

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

Output : Set of Nullable variables

```
OLDN = ∅
NEWN = { A | A → λ is in P }
while ( OLDN ≠ NEWN ) do
  Begin
    OLDN ← NEWN
    NEWN ← OLDN ∪ { A | A → α , all symbols in α is in OLDN }
  End
```

مثال: $S \to ABa \mid BBb$, $B \to CD \mid \lambda$, $C \to aC \mid \lambda$

$D \to dD \mid CB$, $A \to BB \mid a$

OLDN = { }	NEWN = {B, C}
OLDN = {B, C}	NEWN = {A, B, C, D}
OLDN = {A, B, C, D}	NEWN = {A, B, C, D}

مثال: « حذف قواعد λ »

$S \rightarrow AaB \mid aaB$

$A \rightarrow \lambda$

$B \rightarrow bbA \mid \lambda$

$\{A, B\}$ ⟵ « مجموعه متغیرهای تهی »

→

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

output: $G' = (V, T, P', S)$ such that P' does not contain λ productions

P' is constructed as follows:

if $A \rightarrow X_1 X_2 \cdots X_n$ is in P then

Add $A \rightarrow \alpha_1 \alpha_2 \cdots \alpha_n$ to

where

- $\alpha_i = X_i$ if X_i is not nullable
- $\alpha_i = X_i$ or λ if X_i is nullable
- not all X_i' are considered nullable

$S \rightarrow a \mid AaB \mid Aa \mid aB$

$S \rightarrow aaB \mid aa$

$B \rightarrow bbA \mid bb$

Subject :

Year . Month . Date . ( )

مثال: $P = \{ S \to ABaC \ , \ A \to BC \ , \ B \to b \mid \lambda \ , \ C \to D \mid \lambda \ , \ D \to d \}$

$\{A, B, C\}$ ⟵ مجموعه متغیرهای پیچ

$S \to ABaC$

$S \to BaC \mid ABaC \mid AaC \mid ABa \mid aC \mid Aa \mid Ba \mid a$

$A \to BC$

$A \to C \mid B \mid BC$

$B \to b \ , \quad C \to D \ , \quad D \to d$

قضیه: حذف قواعد λ ممکن است قواعد واحد جدید ایجاد نماید.

اگر ابتدا قواعد λ را حذف نماییم، سپس قواعد واحد را حذف نماییم، در نهایت قواعد بدون استثناء را حذف نماییم، آنچه که نهایتاً حاصل خواهد شد، گرامری عاری از قواعد λ، واحد و بدون استثناء خواهد بود.

مسئله: 15-769 ، 14-769

Subject:
Year. Month. Date. ( )

مثال: 19 - 170

$$Complexity(G) = \sum_{A \to \gamma \in P} (1 + |\gamma|)$$

✓ حاصل جمع طول ست راست در قاعده بعلاوه 1

به ازای تمام قواعد، پیچیدگی یک گرامر تعریف شده است.

مثال: 20 - 170

✓ گرامر G برای زبان L، Minimal می‌باشد اگر $Complexity(G) \leq Complexity(G')$
برای همه گرامرهای G' برای زبان L، برقرار باشد.

فرم‌های نرمال گرامرهای نوع دوم

نرمال‌های نرمال، فرم‌هایی هستند که سایر فرم‌ها را می‌توان به شکل آنها درآورد!

فرم نرمال چامسکی: یک گرامر به فرم نرمال چامسکی می‌باشد اگر قواعد به فرم زیر باشند.

$$\begin{cases} A \to BC \\ A \to a \end{cases} \qquad A, B, C \in V \;,\; a \in T$$

مثال: «تبدیل به فرم چامسکی»

$S \to ABaCb$

$A \to aA \mid b$

$B \to bB \mid a$

$S \to AB X_a C X_b$

$X_a \to a$

$X_b \to b$

$S \to A X_1$

$X_1 \to B X_2$

$X_2 \to X_a X_3$

$X_3 \to C X_b$

مثال: 5-177

فرم نرمال گیری باخ : تولیدی به فرم زیر که نوع تولید فرم نرمال گیری باخ می باشد.

$A \to aX \qquad a \in T, \quad X \in V^*$

مثال:

$S \to aAB \mid bBB$

$A \to aB \mid b$

$B \to bB \mid a$

Subject:
Year. Month. Date. ( )

مثال: $S \rightarrow aSb \mid ab$

$S \rightarrow aSb \rightsquigarrow S \rightarrow aSB \,,\; B \rightarrow b$

$S \rightarrow ab \rightsquigarrow S \rightarrow aB \,,\; B \rightarrow b$

مثال: $S \rightarrow SS \mid aS \mid b$ حذف قواعد چپ گشتی از سمت چپ.

$$\begin{cases} S \rightarrow aSZ \mid aS \mid bZ \mid b \\ Z \rightarrow S \mid SZ \end{cases}$$

✓ در صورت نیاز باید قواعد واحد را هم حذف کنیم.

$$\begin{cases} A \rightarrow x_1 B x_2 \,,\; B \rightarrow y_1 \mid y_2 \mid \cdots \mid y_n \\ B \rightarrow x_1 y_1 x_2 \mid x_1 y_2 x_2 \mid \cdots \mid x_1 y_n x_2 \end{cases}$$

یک تبدیل ساده:

✓ قواعد چپ گشتی را نیز باید در صورت نیاز حذف کرد.

$Z \rightarrow SZ \mid aSZ \mid aS \mid bZ \mid b$

$Z \rightarrow aSZZ \mid aSZ \mid bZZ \mid bZ$

Subject :

Year . Month . Date . ( )

مسئله 7-177 : «عدم نرمال و الگوریتمی برای آن»

$A \to aB$

$A \to Ba$

$A \to a$ $\quad A, B, C \in V \quad , \quad a \in T$

مسئله 12-195 :

$A \to aBC$

$A \to aB$ $\quad A, B, C \in V \quad , \quad a \in T$

$A \to a$

Subject :

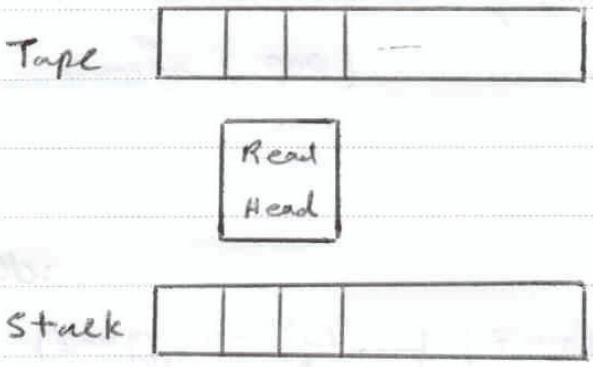
Year . Month . Date . ( )

Subject:

Year. Month. Date. ( )

پذیرنده برای زبان های نوع دوم

Push-Down Automata (PDA) (ماشین های پشته ای)



✓ حالت بعدی ماشین، به حالت جاری ماشین، آنچه که در روی Tape می خواند و آنچه که در Stack قرار دارد، بستگی دارد.

✓ بطور رسمی ماشین های پذیرنده زبان های نوع دوم، بصورت زیر تعریف می شوند.

$$PDA = (Q, \Sigma, T, \delta, q_0, Z, F)$$

- Q: مجموعه حالات
- Σ: الفبای زبان
- T: الفبای Stack
- δ: تابع انتقال
- q_0: حالت اولیه
- $Z \in T$: سربرگ
- F: حالات نهایی

$$\delta: Q \times (\Sigma \cup \{\lambda\}) \times T \longrightarrow \text{a finite subset of } Q \times T^*$$

مثال:

$\delta(q, b, y) = \{(q', xy)\}$ ⟵ push

$\delta(q, a, y) = \{(q'', \lambda)\}$ ⟵ اگر ماشین حالت q باشد و a از نوار ورودی خوانده شود، و نیز y در رأس پشته باشد، ماشین به حالت q'' می‌رود، y از رأس پشته برداشته می‌شود و چیزی هم به جای آن قرار نمی‌گیرد (POP)

مثال:

$(q, bbab, yxnZ) \vdash (q', bab, xyxnZ) \vdash (q'', ab, yyxnZ)$

$\vdash (P, b, yxnZ)$

قسمتی از رشته که تاکنون خوانده شده است.

سر پشته

$(q, bbab, yxnZ) \vdash^{*} (P, b, yxnZ)$

(q, ω, γ)

- q: حالت جاری ماشین
- ω: قسمت خوانده نشده رشته ورودی
- γ: قسمت پشته

Subject:

Year. Month. Date. ( )

* **پذیرش توسط حالت نهایی:** در صورتی که پس از خواندن تمام کاراکترهای رشته ورودی از حالات اولیه به یکی از حالات نهایی برسیم، رشته ورودی پذیرفته می‌شود و در غیر اینصورت اگر در حالت نهایی نباشیم یا ماشین به دلایلی متوقف شود، رشته پذیرفته نشده است.

* **پذیرش توسط پشته خالی:** یک رشته پذیرفته می‌شود اگر در انتهای آن پشته خالی باشد

یک رشته پذیرفته می‌شود اگر در انتهای آن پشته خالی شود، یا قبل از رسیدن به انتهای رشته و خالی شدن پشته ماشین متوقف شود.

مثال: همانطور که پیش‌تر نشان دادیم، زبان روبرو از نوع زبان‌های نوع دوم است.

$$L = \{a^n b^n \mid n > 0\}$$

✔ در استاندارد است یک الگوریتم High Level ارائه دهیم که فقط با استفاده از پشته رشته‌های این زبان را بپذیرد. پس سعی می‌کنیم به طریقی این الگوریتم را به ماشین آن تبدیل کنیم.

✔ در این مثال فرض است الفبای a را اگر پشته خالی بود در پشته push می‌کنیم پس تا زمانی که سر پشته a باشد، a بفرستیم، همچنان a را در پشته push می‌کنیم. پس اگر b فرستادیم و سر پشته a بود، پشته را pop می‌کنیم تا پشته خالی شود. این عمل تا خالی شدن پشته، نشان دهنده این است که تعداد a و پس تعداد b ها برابر است.

$$\delta(q_0, \lambda, Z) = \{(q_f, Z)\}$$
$$\delta(q_0, a, Z) = \{(q_1, aZ)\}$$
$$\delta(q_1, a, a) = \{(q_1, aa)\}$$
$$\delta(q_1, b, a) = \{(q_2, \lambda)\}$$
$$\delta(q_2, b, a) = \{(q_2, \lambda)\}$$
$$\delta(q_2, \lambda, Z) = \{(q_f, Z)\}$$

→ در انتهای رشته (λ یا)

Subject:
Year. Month. Date. ( )

$(q_1, aaabb, Z) \vdash^{*} (q_2, aZ)$ ✗

نسخه‌ای برای این حالت وجود
ندارد و رشته پذیرفته نمی‌شود.

$(q_0, abab, Z) \vdash^{*} (q_2, ab, Z)$ ✗ نسخه‌ای برای این حالت هم
تعریف نشده است. لذا رشته
پذیرفته نمی‌شود.

سوال: برای مثال قبل نشان کنید که حالات q_0 حذف شوند و فقط حالات نهایی به همان حال باقی بمانند.
آکنون نگاهید ماشین جدید چه زبانی را می‌پذیرد.

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

❗ حالات تشویش باری نمی‌کنند، می‌توانیم بدون تغییر حالت
از q_0 به q_1 منتقل پذیرش را انجام دهیم.

$\delta(q_0, a, Z) = \{(q_1, aZ)\}$
$\delta(q_0, b, Z) = \{(q_1, bZ)\}$
$\delta(q_1, a, a) = \{(q_1, aa)\}$
$\delta(q_1, a, b) = \{(q_1, \lambda)\}$
$\delta(q_1, b, a) = \{(q_1, \lambda)\}$
$\delta(q_1, b, b) = \{(q_1, bb)\}$

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

Subject:
Year. Month. Date. ( )

✔ برای اینکه بتوانیم نشان دهیم یک زبان نوع دوم است، باید نشان دهیم که یک پذیرنده برای آن وجود دارد و اگر به هیچ طریقی نتوان چنین پذیرنده‌ای را طراحی کرد، زبان نوع دوم نیست و نوع اول است.

عدم قطعیت

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

زبان تولید مجموعه رشته‌های پالیندروم است و لذا نوع زبانی آن نوع دوم است.

✔ تصمیم گرفتیم وسط رشته می‌رویم، بنابراین تا وسط به پشته push می‌کنیم و پس از آن هر حرفی که آنجا می‌خواند با آنچه که در پشته است یکی باشد، pop می‌کند تا پشته خالی شود.

! اما از آنجایی که از قبل نمی‌دانیم وسط رشته کجاست، ماشین با این نوع تصمیم‌گیری از خود نشان دهد و تضمین دهد که در کدام حالات ممکن است در وسط رشته حدس زده باشد. این کار با استفاده از عدم قطعیت امکان‌پذیر است.

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

$\delta(q_0, b, Z) = \{(q_1, bZ)\}$

$\delta(q_1, b, b) = \{(q_1, bb), (q_2, \lambda)\}$

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

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

$\delta(q_1, a, a) = \{(q_1, aa), (q_2, \lambda)\}$

$\delta(q_2, a, a) = \{(q_2, \lambda)\}$

$\delta(q_2, b, b) = \{(q_2, \lambda)\}$

$\delta(q_0, \lambda, Z) = \{(q_f, Z)\}$

$\delta(q_2, \lambda, Z) = \{(q_f, Z)\}$

Subject:
Year. Month. Date. ( )

✓ ماشین فوق از نوع ماشین های «Non-Deterministic» می باشد.

✓ اگر بخواهیم زبان مثال قبل بصورت wcw^R باشد با افزودن دو قسمت زیر مشکل عدم قطعیت حل می شود.

$$\delta(q_1, c, a) = \{(q_2, a)\}$$
$$\delta(q_1, c, b) = \{(q_2, b)\}$$

Subject:

Year. Month. Date. ( )

تعریف: یک زبان نوع دوم تعلقی است، اگر بتوان برای آن یک DPDA طراحی کرد

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

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

Deterministic Context Free Language ⟶ DCFL

تعریف: یک زبان نوع دوم غیر تعلقی است، اگر بتوان برای یک PDA تعلقی طراحی کرد.

$L = \{ww^R \mid w \in \{a,b\}^*\}$. $L = \{a^n b^m c^k \mid n=m \text{ or } m=k\}$

Non-Deterministic Context Free Language ⟶ NCFL

طراحی PDA

برای طراحی یک ماشین پذیرنده زبان‌های نوع دوم، صرفاً نشان دادن اینکه یک الگوریتم High Level وجود دارد که فقط با استناده Stack نشده را حل می‌کند، به معنی وجود چنین ماشینی است.

مثال: (۵ تا مثال حل می‌ماند) $L = \{a^n b^{2n} \mid n \geq 0\}$

$\delta(q_0, \lambda, Z) = \{(q_f, Z)\}$ ----

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

$\delta(q_1, a, a) = \{(q_1, aaa)\}$ --

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

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

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

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

گفتیم a ها را ابتدا در پشته قرار دهیم. پس
b ها را هم در پشته قرار می‌دهیم.
از روی c ها رد می‌کنیم.
پس دوباره در برخورد با b ها، اگر سر پشته b بود آن را پاک می‌کنیم.
نهایتاً در برخورد با a ها، اگر سر پشته a بود، آن را پاک می‌کنیم.
اگر پشته خالی بود، یعنی رشته پذیرفته شده است.

Subject:

Year. Month. Date. ( )

مثال: برای زبان زیر تحت هیچ شرایطی نمی‌توان یک ماشین پشته‌ای طراحی کرد. زیرا در هر مرحله باید اطلاعات مورد نیاز سر پشته باشد که چنین شرایطی برقرار نیست.

$L = \{ a^n b^m c^p a^n b^m \mid n,m,p \geq 0 \}$ ✗

مثال: برای زبان زیر هم تحت هیچ شرایطی نمی‌توان یک ماشین پشته‌ای طراحی کرد. زیرا از اطلاعات موجود در پشته فقط یکبار می‌توانیم استفاده کنیم.

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

مثال: زبان روبرو هم از نوع ماشین‌های پشته‌ای نیست. زیرا با برداشتن اطلاعات از سر پشته، اطلاعات مورد نیاز در مراحل بعد، از بین می‌روند.

$L = \{ ww \mid w \in \{a,b\}^* \}$ ✗

مثال: این زبان هم از نوع ماشین‌های پشته‌ای نیست زیرا نمی‌توان تساوی سمت راست را با پشته بررسی نمود.

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

مثال: تساوی به قسمت مشخص شده با خط چین را نمی‌توان بررسی کرد.

$L = \{ \underbrace{a^n b^m c^r} \; d^q \; \underbrace{c^r \; a^n b^m} \}$ ✗

مثال: تساوی $a^n b^n$ را به راحتی می‌توان با پشته بررسی نمود.

$L = \{ a^3 b^n c^n \mid n \geq 0 \}$

برای بررسی وجود دقیقا 3 عدد a، خودمان یک حالت جدید ایجاد می‌کنیم تا بعد از دیدن 3 عدد a وارد آن حالت بعد شویم.

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

$\delta(q_1, a, Z) = \{(q_2, Z)\}$

$\delta(q_2, a, Z) = \{(q_3, Z)\}$

Subject:
Year. Month. Date. ( )

مثال: ابتدا فقط یک b در پشته قرار می‌دهیم و سپس مانند مثال قبلی مشابه عمل می‌کنیم.

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

طراحی PDA از روی گرامر

برای طراحی یک ماشین PDA از روی گرامر، گرامر باید به فرم نرمال گریباخ (GNF) باشد.

$A \longrightarrow a X$

$a \in T$

$x \in V^*$

$A \in V$

$\Rightarrow \delta(q_1, a, A) = \{(q_1, X)\}$

مثال:

$S \rightarrow aA \ ,\quad A \rightarrow aABC \mid bB \mid a$

$B \rightarrow b \ ,\quad C \rightarrow c$

$S \rightarrow aA: \quad \delta(q_1, a, S) = \{(q_1, A)\}$

$A \rightarrow aABC: \quad \delta(q_1, a, A) = \{(q_1, ABC)\}$

$A \rightarrow bB: \quad \delta(q_1, b, A) = \{(q_1, B)\}$

$A \rightarrow a: \quad \delta(q_1, a, A) = \{(q_1, \lambda)\}$

Subject:

Year. Month. Date. ( )

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

$C \to c$: $\delta(q_1, c, C) = \{(q_1, \lambda)\}$

✔ دو حالت جدید هم خودمان برای δ ایجاد می‌کنیم.

$\delta(q_0, \lambda, Z) = \{(q_1, SZ)\}$ (یکی برای اینکه رشته با طول صفر سریعا خوانده شود و

$\delta(q_1, \lambda, Z) = \{(q_f, Z)\}$ یکی هم برای اینکه به پایان رشته برسیم)

$S \Rightarrow aA \Rightarrow aaABC \Rightarrow aabBBC \Rightarrow aabbBC \Rightarrow aabbbC \Rightarrow aabbbc$

$(q_0, aabbbc, Z) \vdash$

$(q_1, aabbbc, SZ) \vdash$

$(q_1, abbbc, AZ) \vdash$

$(q_1, bbbc, ABCZ) \vdash$

$(q_1, bbc, BBCZ) \vdash$

$(q_1, bc, BCZ) \vdash$

$(q_1, c, CZ) \vdash$

$(q_1, \lambda, Z) \vdash (q_f, \lambda, Z)$

✔ در واقع یک تناظر یک به یک بین قواعد اشتقاق و ماشین وجود دارد.

Subject:
Year. Month. Date. ( )

مثال: $L = \{ a^n b^m c^k \mid n=m \text{ or } m=k \}$

✓ زبان فوق از نوع زبان‌های نوع دوم غیرمتقلی است.

- این زبان از اجتماع (Union) دو زبان زیر حاصل می‌شود.

$$\{ a^n b^n c^k \mid n,k \geq 0 \} \cup \{ a^n b^k c^k \mid n,k \geq 0 \}$$

برای هر یک از زبان‌های فوق بطور جداگانه ماشین‌های متقلی وجود دارد. اما زبان با اجتماع دو زبان فوق است. لذا ماشین ما باید نوعی هوشمندی از خود نشان دهد و تشخیص دهد رشته ورودی از کدام یک از دو زبان بالا آمده است.

مثال: $L = \{ a^n b^m \mid n=m \text{ or } m=2n \}$

بر این مثال هم مانند مثال قبل

زبان L از اجتماع دو زبان

$L = \{ a^n b^n \mid n \geq 0 \} \cup \{ a^n b^{2n} \mid n \geq 0 \}$

پدید می‌آید که هر یک به تنهایی

ماشین‌های متقلی دارند و یک ماشین غیرمتقلی برای کل زبان L وجود دارد که از خود هوشمندی نشان می‌دهد و تشخیص می‌دهد رشته ورودی از کدام زبان است تا وارد ماشین آن شود.

مثال: PDA زیر چه زبانی را پذیرا است (10-187).

$\delta(q_0, a, Z) = \{ (q_1, a), (q_2, \lambda) \}$

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

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

$\delta(q_1, a, b) = \{ (q_2, \lambda) \}$

Subject:

Year. Month. Date. ( )

✔ برای حل مسائلی به صورت ترتیبی، باید منطق PDA را روی یک برنامه Trace (ردیابی) کرد.

از این عبارت به یاد می‌گیرید که زبان تهی از اجتماع دو زبان پذیری آید.
بنابراین هر یک از دو زبان را باید جداگانه بررسی کنیم

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

✔ زبان‌های نوع سوم تهی زیر مجموعه‌ای از زبان‌های نوع دوم تصفی (DCFL) می‌باشند. یعنی برای همه آنها ماشین‌های حقیقی وجود دارد.

Subject :

Year . Month . Date . ( )

Subject:

Year. Month. Date. ( )

« خصوصیات زبان‌های نوع دوم »

گرامر خطی : یک زبان نوع دوم خطی است، اگر بتوان برای آن یک گرامر نوع دوم خطی بدست آورد.

Linear CFL

یک گرامر نوع دوم خطی است اگر در سمت راست هر قاعده این گرامر، حداکثر یک متغیر ظاهر شده باشد.

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

$P = \{ S \rightarrow aSb \mid ab \}$

زبان غیر خطی : یک زبان نوع دوم غیر خطی است، اگر نتوان برای آن یک گرامر خطی بدست آورد.

$P = \{ S \rightarrow SS \mid aSb \mid bSa \mid \lambda \}$

$P = \{ S \rightarrow aSbS \mid bSaS \mid \lambda \}$

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

Subject:

Year. Month. Date. ( )

قضیه pumping برای زبان های نوع دوم

فرض کنید L یک زبان نوع دوم و نامتناهی باشد. در این صورت یک عدد صحیح مثبت m وجود دارد، بطوریکه برای $w \in L$ ، $|w| \geq m$ ، آن را بدین صورت $uvxyz$ تجزیه کرد. (شرایط تجزیه: $|vy| \geq 1$ ، $|vxy| \leq m$) بطوریکه:

$$\forall i : \quad uv^i xy^i z \in L$$

$i = 0, 1, 2, \ldots$

✓ از قضیه فوق می توان برای نشان دادن اینکه یک زبان از نوع دوم نیست، استفاده نمود.

مثال:

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

«من» «رقیب»

$a^m b^m c^m$ ← m

↓ → aa...abb...bccc

$u(vxy)z$

✓ من باید بتوانم تمام حرکات رقیب را پوشش داده باشم.

Subject :

Year . Month . Date . ( )

✓ چون هیچ شرطی روی w و z نداریم، لذا اینکه vxy را به صورت یک قطعه متوالی در نظر می‌گیریم که حالات مختلفی می‌تواند داشته باشد. این قطعه یا روی a یا روی b یا روی a و b یا روی b و c قرار می‌گیرد ولی روی a، b و c هم‌زمان قرار نمی‌گیرد.

تمرین: نشان دهید زبان $L = \{ww \mid w \in \{a,b\}^*\}$ از نوع دوم نمی‌باشد.

قضیه pumping برای زبان‌های نوع دوم خطی

فرض کنید L یک زبان نوع دوم خطی و نامتناهی باشد. در اینصورت یک عدد صحیح مثبت m وجود دارد بطوریکه برای هر $w \in L$ ، $|w| \geq m$ ، آن را می‌توان به صورت $uvxyz$ تجزیه کرد (شرایط تجزیه: $|vy| \geq 1$ ، $|vxy| \leq m$) بطوریکه:

$$\forall i : \quad uv^i xy^i z \in L$$

$i = 0, 1, 2, \ldots$

✓ از قضیه فوق نیز می‌توان برای نشان دادن اینکه یک زبان نوع دوم خطی نیست، استفاده کرد.

Subject:
Year. Month. Date. ( )

مثال: نشان دهید زبان روبرو، نوع دوم خطی است

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

«رتبه» «من»

$a^m b^m b^m a^m$ ← m

$\rightarrow$ aa...a bb...b cc...c

خصوصیات بسته‌ای زبان‌های نوع دوم

قضیه: خانواده زبان‌های نوع دوم تحت عملگر U بسته است.

یعنی اگر L_1 و L_2 دو زبان نوع دوم باشند، $L_1 \cup L_2$ نیز نوع دوم است.

برای اثبات قضیه ذکر چند نکته لازم است. نشان دادن اینکه زبانی نوع دوم است، می‌تواند با نوشتن قواعدی گرامری برای اجتماع آن دو باشد. (Constructive proof)

Subject :

Year . Month . Date . ( )

- قضیه: خانواده زبان‌های نوع دوم تحت عمگر $\cap$ بسته نیست.

برای اثبات قضیه فوق از مثال نقض استفاده می‌کنیم.

$L_1 = \{a^n b^n c^m \mid n,m \geq 0\}$ ⟵ زبان نوع دوم ✓

$L_2 = \{a^n b^m c^n \mid n,m \geq 0\}$ ⟵ زبان نوع دوم ✓

$L_1 \cap L_2 = \{a^n b^n c^n \mid n \geq 0\}$ ⟵ یک زبان نوع اول ✗

- قضیه: خانواده زبان‌های نوع دوم تحت عمگر متمم بسته نیست.

برای اثبات از برهان خلف و قضیه دمورگان استفاده می‌کنیم.

- قضیه: خانواده زبان‌های نوع دوم تحت عمگر «اشتراک منظم» بسته می‌باشد.

Regular Interseetion

⟵ یعنی اشتراک یک زبان نوع دوم با یک زبان نوع منظم، یک زبان نوع دوم است.

(اثبات 5-8 در صفحه 223)

مثال: نشان دهید زبان مقابل دوم است. $L = \{a^n b^n \mid n \geq 0,\ n \neq 100\}$

Subject:
Year. Month. Date. ( )

✓ برای نشان دادن اینکه یک زبان نوع دوم است می‌توان از: خصوصیات بسته‌ی زبان‌های نوع استفاده کرد و مطلب فوق را نشان داد.

● قضیه: خانواده زبان‌های نوع دوم تحت عملگر «تفریق منظم» بسته است.

Regular difference

حال اگر L یک زبان نوع دوم و R یک زبان نوع سوم باشد،
در اینصورت L-R نوع دوم است.

مثال: نشان دهید زبان مقابل از نوع دوم نیست.

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

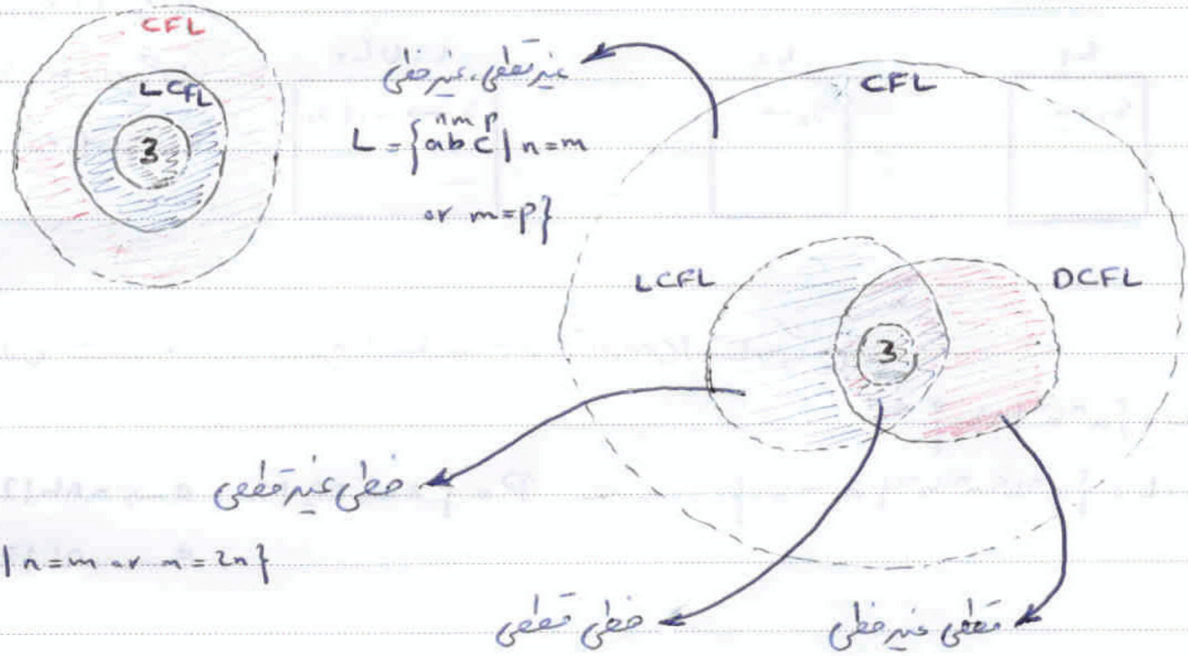
$$\{a^n b^n c^n \mid n \geq 0\} = L \cap L(a^* b^* c^*)$$

یک زبان نوع سوم

✓ فرض کنیم L یک زبان نوع باشد، چون $L(a^* b^* c^*)$ یک زبان نوع سوم است پس با به اشتراک منظم این دو نیز نوع دوم باشد ولی زبان $\{a^n b^n c^n \mid n \geq 0\}$ نوع دوم نیست لذا فرض ابتدایی خلط برده است و L یک زبان نوع دوم نیست (برهان خلف)

Subject:
Year. Month. Date. ( )

✓ خانواده زبان‌های نوع سوم در واقع زیرمجموعه خانواده زبان‌های نوع دوم خطی می‌باشند. (طبق تعریف همه زبان‌های نوع سوم همگی خطی هستند)



غیرقطعی، غیرخطی: $L = \{a^n b^m c^p \mid n = m \text{ or } m = p\}$

خطی غیرقطعی: $L = \{a^n b^m \mid n = m \text{ or } m = 2n\}$

خطی قطعی: $L = \{a^n b^n \mid n \geq 0\}$

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

مثال: نشان دهید که زبان‌های نوع دوم تحت «هومومورفیزم» بسته است.

باید یک رویه ارائه دهیم که با استفاده از آن گرامری ارائه دهیم که $h(L)$ آن زبان را بپذیرد. کافیست تا به جای a، b، c، ... سمبل‌های آنها نیز $h(a)$، $h(b)$، $h(c)$ را جایگزین کنیم.

مثال: نشان دهید که خانواده زبان‌های نوع دوم تحت معکوس بسته است.

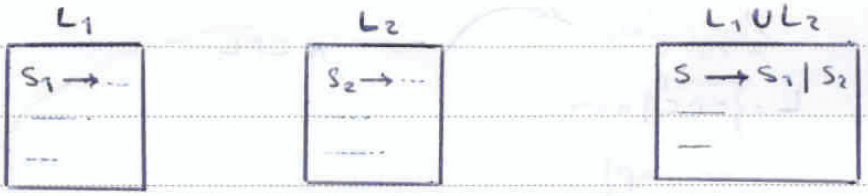
باز هم طبق رویه‌ای، سمت راست قواعد گرامر زبان را معکوس می‌کنیم و گرامر حاصل، گرامر زبان معکوس (L^R) خواهد بود.

Subject:

Year. Month. Date. ( )

مثال: نشان دهید که خانواده زبان های نوع دوم خطی تحت U بسته است ولی تحت Concatenation بسته نیست.

بنابراین می توان با داشتن گرامر دو زبان خطی نوع دوم، گرامر اجتماع آنها را هم بدست آورد.



برای نشان دادن بسته نبودن نسبت به عملگر Concatenation از مثال نقض استفاده می کنیم.

$L = \{a^n b^n \mid n > 0\}$

$L \cdot L = \{a^n b^n a^m b^m \mid n, m > 0\}$ ⟵ $P = \{S \rightarrow AB \mid \lambda \ , \ A \rightarrow aAb \mid \lambda$

$B \rightarrow aAb \mid \lambda\}$

مثال: نشان دهید خانواده های زبان های نوع دوم تحت اشتراک بسته نیست.

$L_1 = \{a^n b^n c^m \mid n, m > 0\}$ ⟵ نوع دوم قطعی

$L_2 = \{a^n b^m c^n \mid n, m > 0\}$ ⟵ نوع دوم قطعی

$L_1 \cap L_2 = \{a^n b^m c^p \mid n = m \text{ or } m = p\}$ ⟵ نوع دوم غیرقطعی

باز هم از مثال نقض استفاده می کنیم.

مسائل قابل تصمیم گیری برای زبان های نوع دوم

- این مسئله که یک زبان نوع دوم تهی باشد قابل تصمیم گیری است. (در صورتی متغیرهای غیرقابل حذف، S باشد)
- این مسئله که یک زبان نوع دوم ناتهی باشد قابل تصمیم گیری است.
- مسئله تساوی دو زبان نوع دوم غیرقابل تصمیم گیری است.

Subject:

Year. Month. Date. ( )

* متغیرهای تکرارشونده: متغیر A تکرارشونده است، اگر: $A \overset{*}{\Rightarrow} \alpha A \beta$

برای یک گرامر داده شده، گراف وابستگی آن را می‌کشیم (node های گراف متغیرهای ما می‌باشند و یال های گراف، قواعد ما باشند)

برای پیدا کردن متغیرهای تکرارشونده، cycle های (دورهای) گراف را پیدا می‌کنیم. تمام متغیرهایی که بر روی حلقه‌ها (دورها) هستند، متغیرهای تکرارشونده هستند.

✓ یک گرامر نوع دوم، یک زبان ناشتاهی را تعریف می‌کند، اگر و تنها اگر حداقل یک متغیر تکرارشونده باشد.

(I) $A \overset{*}{\Rightarrow} w$

(II) $S \overset{*}{\Rightarrow} \alpha A \beta$

(III) $A \overset{*}{\Rightarrow} \alpha' A \beta'$

} ⇝ «تولید بی‌نهایت رشته» $\quad (S \overset{*}{\Rightarrow} \alpha \alpha'^{n} w \beta'^{n} \beta)$

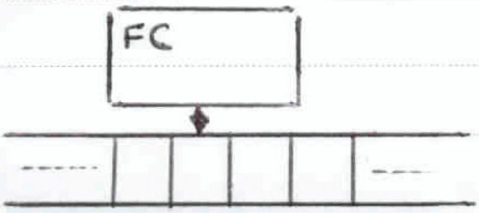
Subject:

Year. Month. Date. ( )

«تئوری زبان‌های نوع اول و صفر»

Turing Machine

ماشین Turing ، ماشین پذیرنده زبان‌های نوع صفر است که توسط شخصی بنام Allan Turing مطرح شده است.



این ماشین از یک نوار نامتناهی از هر دو طرف تشکیل شده است که هم خواندنی و هم نوشتنی است و هد به هر دو طرف نوار می‌تواند حرکت کند.

- infinite Tape
- Read/write Tape
- move in both directions

ماشین Turing بصورت رسمی به شکل زیر تعریف می‌شود.

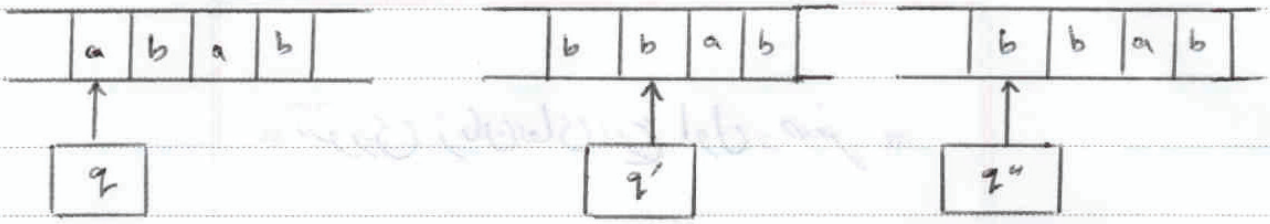
$$TM = (Q, \Sigma, T, \delta, q_0, \square, F)$$

- F → مجموعه حالات نهایی
- Q → مجموعه حالات
- Σ → الفبای زبان
- T → Tape Alphabet الفبای نوار
- $\delta: Q \times T \longrightarrow Q \times T \times \{L, R\}$
- $q_0 \in Q$
- $\square$ → Blank symbol $\square \in T$

Subject:

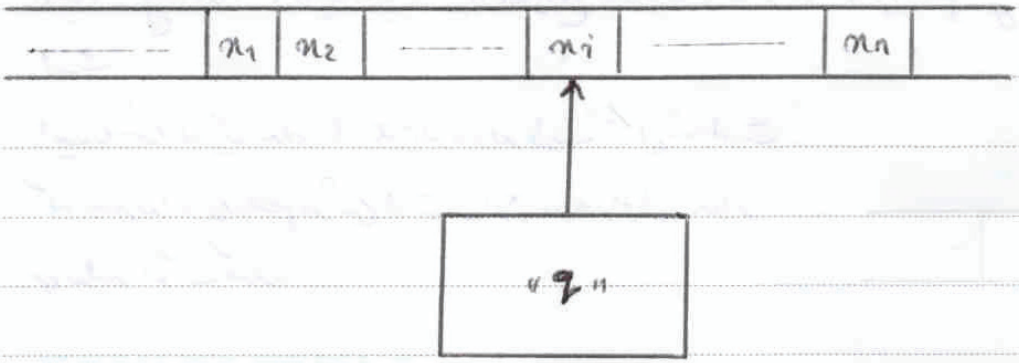
Year. Month. Date. ( )

مثال:



$\delta(q, a) = (q', b, R)$ $\delta(q', b) = (q'', b, L)$

✔ برای سهولت بیشتر از نمایش زیر استفاده می‌کنیم.



$x_1 x_2 \cdots q x_i \cdots x_n \longmapsto x_1 x_2 \cdots a q' x_{i+1} \cdots x_n$

$\delta(q, x_i) = (q', a, R)$

پذیرش: ماشین Turing یک رشته ورودی پذیرفته می‌شود، اگر ماشین با ورودی از حالات نهایی خود شود و پس متوقف شود.

یک رشته ورودی پذیرفته نمی‌شود، اگر ماشین با ورودی از حالات غیر نهایی خود گردد و یا متوقف نگردد.

Subject:

Year. Month. Date. ( )

مثال:

$Q = \{q_0, q_1\}$, $\Sigma = \{a, b\}$, $T = \{a, b, \square\}$

$F = \{q_1\}$

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

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

$q_0\, abab \vdash bq_0\, bab \vdash bbq_0\, ab \vdash bbbq_0\, b \vdash bbb\, bq_0\, \square$

$\vdash bbbq_1\, b$ ✓

مثال:

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

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

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

$q_0\, ab\, aab \vdash aq_1\, baab \vdash q_0\, ab\, aab$

$\vdash aq_1\, baab \vdash q_0\, ab\, aab$

---- ✗

✓ در اینجا loop نامتناهی (ابدی) یا به هیچ وقت متوقف نمی‌شود (Never Halts)

(Infinite loop)

! در Turing چه چیزی نوشته شود و چه نشود، الزامی به حفظ عدم تغییر نوشته اولیه نیست!

مثال: برای زبان روبرو یک پذیرنده Turing طراحی کنید.

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

aa...a bb...b

اولین a را پاک میکنیم و تیک می‌زنیم، به سمت جلو می‌رویم تا اولین b را تیک می‌زنیم سپس مجدداً عقب برمی‌گردیم تا اولین a سمت چپ تیک نخورده را تیک می‌زنیم سپس به جلو می‌رویم تا اولین b سمت راست تیک نخورده را هم تیک می‌زنیم و الی آخر...

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

$$\Sigma = \{a, b\} \quad , \quad T = \{x, y, \square, a, b\}$$

(x برای تیک زدن a ها، y برای تیک زدن b ها استفاده می‌شود)

$$\delta(q_0, a) = (q_1, x, R)$$
$$\delta(q_1, a) = (q_1, a, R)$$
$$\delta(q_1, y) = (q_1, y, R)$$
$$\delta(q_1, b) = (q_2, y, L)$$

$$\delta(q_2, y) = (q_2, y, L)$$
$$\delta(q_2, a) = (q_2, a, L)$$
$$\delta(q_2, x) = (q_0, x, R)$$

$$\delta(q_0, y) = (q_3, y, R)$$
$$\delta(q_3, y) = (q_3, y, R)$$
$$\delta(q_3, \square) = (q_4, \square, R)$$

Subject:

Year. Month. Date. ( )

$q_0 aaaabbbb \vdash x q_1 aaabbbb \vdash xa q_1 aabbb \vdash$

$xaa q_1 abbbb \vdash xaaa q_1 bbbb \vdash xaa q_2 aybbb \vdash$

$xa q_2 aaybbb \vdash x q_2 aaaybbb \vdash q_2 xaaaybbb \vdash$

$x q_0 aaaybbb \vdash xx q_1 aaybbb \vdash xxa q_1 aybbb \vdash$

$xx aa q_1 ybbb \vdash xxaay q_1 bbb \vdash xxaa q_2 yybb \vdash$

$xxa q_2 ayybb \vdash xx q_2 aayybb \vdash x q_2 xaayybb \vdash$

$xx q_0 aayybb \vdash xxx q_1 ayybb \vdash xxxa q_1 yybb \vdash$

$xxxay q_1 ybb \vdash \cdots \vdash \cdots \vdash *$

$xxxx q_0 yyyy \vdash xxxxy q_3 yyy \vdash xxxxyy q_3 yy \vdash$

$xxxxyyy q_3 y \vdash xxxxyyyy q_3 \square \vdash xxxxyyyy \square q_4 \square$ ✓

✓ وقتی ماشین خودش را در حالت ردیه q_0 می‌بیند، مانند فراخوانی مجدد رویه عمل می‌کند.

تمرین: aaaabbb , aabbb , abab

Subject:

Year. Month. Date. ( )

* مثال: Turing Machine زیر چه زبانی را می پذیرد.

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

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

$T = \{a, b, \square\}$

$F = \{q_3\}$

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

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

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

$\delta(q_1, \square) = (q_3, \square, R)$

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

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

✓ دو تا رشته‌ای اول به ما می‌گوید رشته‌هایی با a یا b شروع می‌شوند.

✓ با توجه به زبان‌های داده شده، رشته‌هایی به صورت ab^n تنها توسط ماشین فوق پذیرفته می‌شوند.

✓ رشته‌هایی به صورت $bb^n a$ نیز تنها توسط ماشین فوق پذیرفته می‌شوند.

← رشته‌هایی که با a شروع می‌شوند و پذیرفته می‌شوند، تنها به صورت $\{ab^n | n \geq 0\}$ می‌باشد

← رشته‌هایی به صورت $\{bb^n a | n \geq 0\}$ پذیرفته می‌شوند اما در مورد ادامه رشته هر تابع و تا چندی مشخص نشده است.

بنابراین تمام رشته‌هایی که prefix آنها $bb^n a$ می‌باشد نیز پذیرفته می‌شوند.

$$L = L(ab^*) \cup L(bb^* a(a+b)^*)$$

(!) الزامی وجود ندارد که ماشین در پایان رشته به حالت نهایی خود برود، بلکه در وسط رشته نیز می‌تواند به حالت نهایی برود و سپس چیزهای دیگری را نیز بپذیرد.

Subject:
Year. Month. Date. ( )

« مدل های غیر استاندارد Turing M »

✓ اگر در خانواده از ماشین های پذیرنده مثل c_1 ، c_2 داشته باشیم، در خانواده c_1 ، c_2 را معادل می گوییم.
اگر به ازای هر ماشین پذیرنده در c_1، یک ماشین پذیرنده معادل در c_2 وجود داشته باشد.

مثال: در خانواده ماشین های NDFA ، DFA معادل یکدیگر هستند.
در خانواده ماشین های NDFAλ ، NDFA معادل یکدیگر هستند.
در خانواده NPDA ، DPDA معادل یکدیگر نیستند.

Turing Machine with Stay on option

✓ ماشین های غیر استاندارد Turing یا ماشین های دیگر Turing ، ماشین های Turing ای هستند که از ماشین Turing پایه به وجود آمده اند.

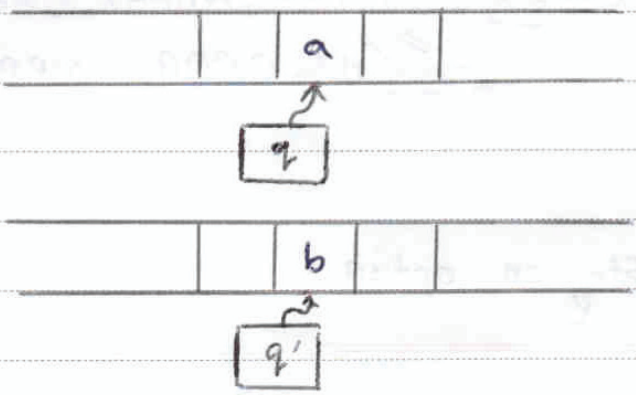
✓ این مدل از ماشین Turing قابلیت باقی ماندن در یک حالت را دارا می باشد.

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

* قضیه: خانواده ماشین‌های Turing با امکان Stay-on معادل خانواده ماشین‌های Turing استاندارد است.

برای اثبات از Simulation استفاده می‌کنیم. باید مرتب را نشان دهیم. اول آنکه هر کاری ماشین تورینگ استاندارد می‌تواند بکند، ماشین تورینگ هم می‌تواند. بالعکس هر کاری که ماشین دیگر انجام می‌دهد، ماشین استاندارد تورینگ هم می‌تواند آن کار را انجام دهد.

$$\delta(q, a) = \delta(q', b, S)$$



ماشین استاندارد تورینگ هم بصورت زیر این کار را انجام می‌دهد:

$$\delta(q, a) = (q'', b, R)$$
$$\delta(q'', c) = (q', b, L)$$

اگر الفبای Tape های n حرف باشد ماشین استاندارد به $n+1$ حرف برای الفبای Tape نیاز است.

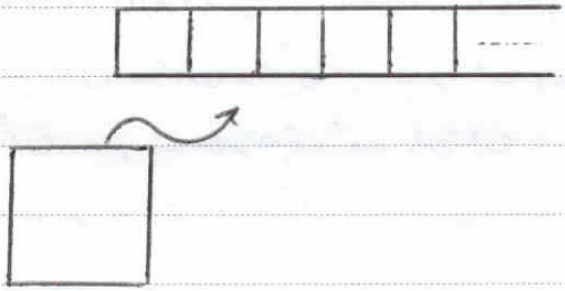
Subject:
Year. Month. Date. ( )

✓ در کامپایل کردن زبان‌های High Level ~ Assembly نیز کار شناسایی را انجام می‌دهیم برای Instruction های سطح بالا به Instruction های سطح پایین تبدیل می‌کنیم.

✓ ماشین‌های غیر استاندارد Turing از نظر قدرت محاسباتی همان قدرت ماشین استاندارد Turing را دارند و قادر به حل همه مسئله‌ای می‌باشند. (تز تورینگ)

Turing Machine with Semi-infinite Tape

تورینگ ماشین با نوار نیمه نشانی بصورت زیر است و نوار آن از یک طرف بسته و از طرف دیگر نا نشانی می‌باشد.

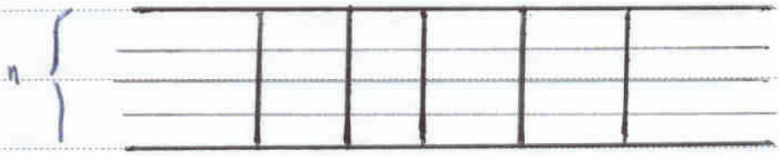


* قضیه: خانواده ماشین‌های Turing با نوار نیمه نشانی معادل خانواده ماشین‌های Turing استاندارد است.

برای اثبات آن کافیست نشان دهیم که نوار نا نشانی را می‌توان با نوار نیمه نشانی Simulate کرد.

Turing Machine with Multiple Track Tape

ماشین‌های Turing نوعی دارای نوارهای چندشیاره می‌باشند.

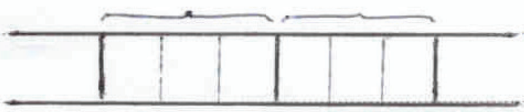


$\delta:$

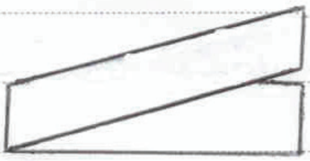
$$Q \times T^n \longrightarrow Q \times T^n \times \{L, R\}$$

* قضیه: خانواده ماشین‌های Turing با نوار نامتناهی چندشیاره معادل با خانواده ماشین‌های Turing استاندارد است.

هرکاری که ماشین استاندارد انجام می‌دهد، ماشین چندشیاره هم قادر به انجام آن است (کافیست فقط از یک نوار آن استفاده کنیم)
بنابراین باید نشان دهیم هر کاری که ماشین چندشیاره انجام می‌دهد، ماشین استاندارد هم انجام می‌دهد.
لذا باید نشان دهیم یک نوار چندشیاره را می‌توان با نوار استاندارد شبیه‌سازی کرد.
برای این کار نیز کافیست تا چندشیار یک خانه را چندخانه روی نوار استاندارد در نظر بگیریم.



✓ برای اثبات قضیه ماشین تورینگ با نوار نیمه‌نامتناهی، نشان دادن اینکه نوار نامتناهی را می‌توان با نوار نیمه‌نامتناهی شبیه‌سازی کرد، از نوار چندشیاره استفاده می‌کنیم.
فرض کنیم یک نوار نامتناهی را از جایی تا بزنیم
تا یک نوار نیمه‌نامتناهی دوشیاره بتوان همان کارها را انجام داد.

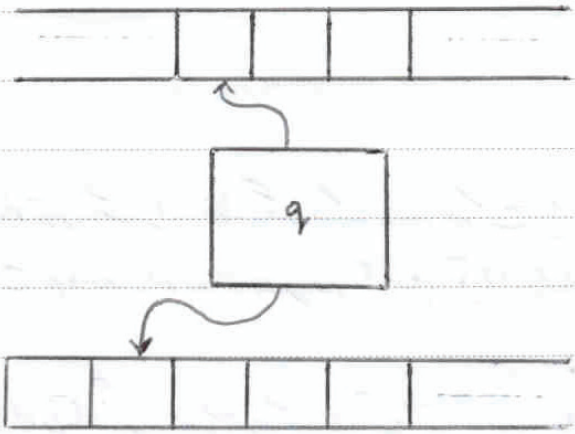


PAPCO

Subject:

Year. Month. Date. ( )

The offline Turing Machine



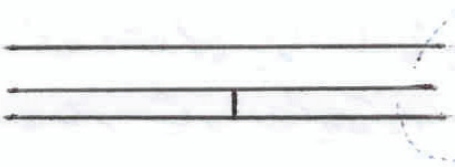
✓ نوار با ورودی از دو طرف بی‌نهایت است ولی نوار کاری فقط از یک طرف بی‌نهایت است

δ:

$Q \times T_1 \times T_2 \longrightarrow Q \times T_1 \times \{L, R\}$

* قضیه: خانواده تورینگ آنلاین معادل با خانواده ماشین‌های تورینگ استاندارد است.

به کاری که ماشین استاندارد انجام می‌دهد، ماشین آنلاین هم می‌تواند آن را انجام دهد. برای اثبات عکس موضوع کافیست نوار یک نوار نامتناهی دو شاره به صورت زیر استفاده کنیم تا بتوانیم به ماشین آنلاین را شبیه‌سازی کنیم.



مثال: زبان $L = \{a^n b^n \mid n \geq 0\}$ را در نظر می‌گیریم با استفاده از پذیرنده تورینگ استاندارد این مسئله پذیرش در زمان $O(n^2)$ حل می‌شود. می‌خواهیم همین مسئله را با تورینگ ماشین آنلاین در زمان $O(n)$ پذیرش کنیم. (رشته را از روی نوار Read-Only می‌خوانیم، و a ها را در نوار نامتناهی کپی می‌کنیم، و وقتی که به b ها رسیدیم، a ها را پاک می‌کنیم)

Subject:
Year. Month. Date. ( )

✓ تورینگ ماشین های غیر استاندارد را می توان با هم ترکیب کرد. ماشین های تورینگ غیر استاندارد جدیدی ساخت.

مثال: یک ماشین تورینگ را در نظر بگیرید که در هر حرکت یا محتوای Tape را عوض می کند یا RW Head را حرکت می دهد ولی نه هر دو با هم. آیا این ماشین معادل ماشین تورینگ استاندارد می باشد؟

به معادل می باشند. هر کاری که ماشین استاندارد انجام می دهد را می توان با در Instruction این ماشین جدید شبیه سازی کرد.

مثال: ماشین تورینگ فوق می تواند یک حالت غیر خاص را با عملیات جای خالی (Blank) جایگزین کنیم.
آیا این ماشین معادل تورینگ ماشین استاندارد می باشد؟

nonerasing Turing Machine

از یک نوار چند نواره با تعداد کافی نوار استفاده می کنیم. حال فرض کنیم می خواهیم یک حالت را Blank کنیم. کل رشته را از نوار اول به نوار بعدی کپی می کنیم به جز آن حالتی که می خواهیم Blank شود.

	a	a	b	b	a	
	a	a	b	a		

Subject:
Year. Month. Date. ( )

تمرین: صفحه 266

Multitape Turing Machine

این ماشین تورینگ دارای چندین نوار نامتناهی خواندنی و نوشتنی است.

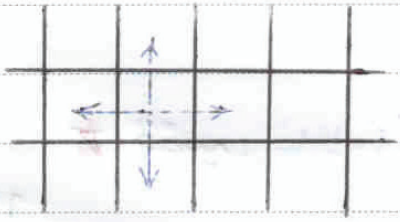
$$\delta: \quad Q \times T_1 \times T_2 \times \dots \times T_n \longrightarrow Q \times T_1 \times T_2 \times \dots \times T_n \times \{ \quad \}$$

* قضیه: ماشین های تورینگ چندنواره معادل خانواده ماشین های تورینگ استاندارد است.

برای سیمولیه کردن چند نوار نامتناهی در یک نوار، کافیت تا از یک نوار چندشاره استفاده کنیم. البته در این حالت مدیریت این چندشاره با یک Head به راحتی خواهد بود.

Multidimensional Turing Machine

در این ماشین های تورینگ نوار ما چندبعدی می باشد. مثلا یک نوار دوبعدی علاوه بر حرکات چپ و راست، حرکات بالا و پایین هم داریم.



Subject:
Year. Month. Date. ( )

* قضیه: خانواده ماشین های تورینگ چندبعدی، معادل خانواده ماشین های تورینگ استاندارد است.

(چیده کردن نوارهای چندبعدی در یک نوار مانند آرایه های چندبعدی می باشد)

تمرین: شکل 4 صفحه 272
شکل 7 صفحه 272

Quene Automata

2PDA (2 Stack Machine)

* آیا ماشین 2PDA معادل تورینگ ماشین است؟

* تمرین: مراحل مربوط به ماشین های PDA، مسائلی مطرح شده است که روی PDA صورت انجام شده است.

Non-Deterministic Turing Machine

$\delta : Q \times T \longrightarrow 2^{Q \times T \times \{L,R\}}$

* قضیه: خانواده ماشین های تورینگ غیرقطعی، معادل با خانواده ماشین های تورینگ قطعی است.

Subject:
Year. Month. Date. ( )

« پذیرنده برای زبان‌های نوع یک »

Linear Bounded Automata (LBA)

یک LBA (پذیرنده زبان نوع یک) یک ماشین تورینگ غیرقطعی است با این محدودیت که برای کلام پذیرفتنی یا عدم پذیرفتنی رشته، نقطه‌ای مورد استفاده نوار، متناسب با طول آن رشته می‌باشد

مثال: قبلا برای این زبان یک ماشین تورینگ طراحی نمودیم
که اگر دقت کنیم متوجه می‌شویم که نقطه مورد استفاده نوار
نوار، نقطه‌ای محدودی می‌باشد.

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

مثال:

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

نوع کنیدی خواهیم برای $n=4$ رشته‌ای از زبان بالا را پذیرفتنی نیم.

✓ بدین منظور از یک نوار مدیاره برای ماشین تورینگ استفاده می‌کنیم که فراتر از یک نقطه‌ی محدود نخواهیم رفت.

Subject:
Year. Month. Date. ( )

✓ تعداد خانه‌های مورد نیاز برای نوار دو شیاره $24 = 4!$ می‌باشد.

روی شیار اول 24 تا a می‌گذاریم. سپس روی شیار دوم دو a می‌گذاریم و روی شیار اول به ازای هر 2 عدد a، یک a را نگه می‌داریم. سپس یک a دیگر روی شیار دوم می‌گذاریم و سپس روی شیار اول به ازای هر سه a یک a را نگه می‌داریم و متعاقباً باز یک a روی شیار دوم می‌گذاریم و باز هم روی شیار اول به ازای هر چهار a، یک a را نگه می‌داریم.

در پایان روی شیار اول فقط یک a و روی شیار دوم، چهار a خواهیم داشت.

تمرین: سعی کنید برای زبان‌های زیر پذیرنده تعریف کنید.

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

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

Subject:
Year. Month. Date. ( )

مثال (P189-E15):

$$\delta: Q \times (\Sigma \cup \lambda) \times T \longrightarrow \text{A finite Subset of } Q \times T^*$$

✓ با توجه به حالت جاری سیستم، علامت روی نوار و علامت سر پشته، حالت بعدی ماشین و علامت سر پشته مشخص می‌شود.

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

رشته‌های با طول صفر، یک یا دو که می‌توانند در سر پشته جایگزین شوند.

آیا محدودیت بالا باعث کاهش توان و قدرت PDA می‌شود ؟؟؟

خیر، زیرا می‌توانیم رشته‌های با طول بیشتر از دو برای جایگزینی در سر پشته را در چند مرحله یکی یکی در سر پشته اضافه کنیم. یعنی می‌توانیم با استفاده از Instruction های بیشتری هر رشته‌ای را در سر پشته قرار دهیم.

مثال: اگر L_1 و $L_1 \cup L_2$ از نوع سوم باشند، آیا L_2 هم از نوع سوم است؟

خیر. براحتی می‌توانیم با مثال نقض نشان دهیم.

$L_1 = L(a^*b^*)$

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

Subject:

Year. Month. Date. ( )

مثال: اگر L_1 و $L_1 \cap L_2$ از نوع سوم باشند، بدین صورت آیا L_2 هم از نوع سوم است؟

خیر. برای رد رد هم می‌توانیم از مثال نقض استفاده کنیم.

$$L_1 = \{ w \mid w \in \{a,b\}^* , |w| \bmod 2 = 1 \}$$
$$L_2 = \{ a^n b^n \mid n \geq 1 \}$$