Subject:
Year. Month. Date. ( )

Theory

اصطلاحات (Terminology)

- الفبا (Alphabet) ← مجموعه‌ای متناهی از علامت‌ها

$\Sigma = \{a, b, c\}$, $\Sigma = \{0, 1\}$

- رشته (string , word , Sentence) ← دنباله‌ای از علامت‌ها

'00' , '111' , '001'

- طول رشته (length) ← $x = '001' \Rightarrow |x| = 3$

- رشته تهی (λ) ← رشته‌ای با طول صفر $|\lambda| = 0$

عملگر هایی رشته‌ها: (operation on strings)

- پیوند (concatenation)

$x = '001'$, $y = '111' \Rightarrow xy = x.y = '001111'$

$\rightarrow x\lambda = \lambda x = x$

$\rightarrow |xy| = |x| + |y|$

دانشکده مهندسی کامپیوتر و فناوری
اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

Reverse
(Reversal) عملگر معکوس

$x = '1011' \Rightarrow x^R = '1101'$

چند تعریف برگشتی برای عملگر معکوس پیدا کنید ←

قسمت پایه

$$(a_1 a_2 \dots a_n)^R = \begin{cases} \text{Basic: } a_n & \text{if } n = 1 \\ \text{Recursive Step: } a_n (a_1 \dots a_{n-1})^R \end{cases}$$

قسمت برگشتی

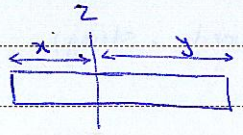
$x = a_1 a_2 \dots a_n \Rightarrow$ طولش n است و n تا علامت دارد.

$(a_1 a_2 \dots a_n)^R = a_n (a_1 a_2 \dots a_{n-1})^R$

مثال → $(abcde)^R = e(abcd)^R = ed(abc)^R = \dots = edcba$

$$(a_1 a_2 \dots a_n)^R = \begin{cases} a_n & \text{if } n = 1 \\ (a_2 \dots a_n)^R a_1 \end{cases}$$

مثال → $(abcde)^R = (bcde)^R a = (cde)^R ba$



$z = xy \Rightarrow z^R = y^R x^R$

$$(xy)^R = \begin{cases} y^R x & \text{if } |x| = 1 \\ y^R x^R \end{cases}$$

در واقع استقرا روی طول x است

مثال → $(abcdef)^R = (def)^R (abc)^R$

$= (ef)^R d (abc)^R$

$= fed(abc)^R$

$= fedcba$

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

هر رشته را می‌توان به بخش‌هایی تحت سلسله (با استفاده از λ)

برای تجزیه بعدی هر رشته را به سه قسمت تقسیم می‌کنیم

$z = uxy \Rightarrow z^R = y^R x^R u^R$

$z = (ux)y \Rightarrow z^R = y^R (ux)^R \Rightarrow z^R = y^R x^R u^R$

تمرین: پیاده‌سازی این ایده‌های برگشتی

اول باید بسیم رشته را چطوری در حافظه ذخیره کنیم ← به لیست‌های پیوندی

روابط عملگر برگشتی

if $z = xy$ then $z^R = y^R x^R$

$(x^R)^R = x$, $\underbrace{(x)^{R\cdots R}}_{\text{زوج}} = x$, $\underbrace{(x)^{R\cdots R}}_{\text{فرد}} = x^R$

پالیندروم (Palindrome) ← از هر طرف خوانده شود یکسان باشد

'abba' , 'ababa'

- x یک پالیندروم است اگر $x = x^R$

← نشان دهید اگر x یک پالیندروم است در این صورت xx هم یک پالیندروم است

فرض : $x = x^R$

حکم : $(xx) = (xx)^R$

اثبات → $(xx)^R = x^R x^R = xx$ → به حکم رسیدیم

← نشان دهید اگر x یک پالیندروم است در این صورت $\underbrace{xxx\ldots x}_{n \text{ مرتبه}}$ نیز یک پالیندروم است

← اگر xx یک پالیندروم است در این صورت x یک پالیندروم است

← اگر $xx\ldots x$ یک پالیندروم است در این صورت x یک پالیندروم است.

x یک پالیندروم است $\Longleftrightarrow$ $\underbrace{xx\ldots x}_{n \text{ مرتبه}}$ یک پالیندروم است

x یک پالیندروم است اگر و فقط اگر $\underbrace{xx\ldots x}_{n \text{ مرتبه}}$ یک پالیندروم است

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

← x و y و z دارای چه ترایطی باشند تا xyz یک پالیندروم باشد

← x و y و z دارای چه ترایطی باشند تا xy^Rz^R یک پالیندروم باشد

← $((x^R)^R y^R)^R \overset{?}{=} yx^R$ ⇒ درست است

زبان (Language)

مجموعه ای محدود یا نامحدود از رشته ها

$\Sigma = \{0,1\} \Rightarrow L = \{000, 10, 11\}$ (finite language) زبان محدود

$\Rightarrow L = \{0, 00, 000, \dots\}$ (infinite language) زبان نامحدود

عملگرهای روی زبان ها:

اجتماع: (Union)

$L_1 \cup L_2 = \{x \mid x \in L_1 \text{ or } x \in L_2\}$

اشتراک:

$L_1 \cap L_2 = \{x \mid x \in L_1 \text{ and } x \in L_2\}$

concatenation: قسمت اول قسمت دوم

$L_1 . L_y = \{xy \mid x \in L_1 \text{ and } y \in L_2\}$

$L_1 = \{00, 11\}$, $L_2 = \{10, 01\}$

$L_1 . L_2 = \{\underbrace{00}_{L_1}\underbrace{10}_{L_2}, 0001, 1110, 1101\}$

$|L_1 . L_2| \leq |L_1| * |L_2|$ ← ممکن است تکراری باشد.

↳ $L_1 = \{0, 00\}$

$L_2 = \{00, 000\}$ $\Rightarrow L_1 . L_2 = \{000, \cancel{0000}, \cancel{0000}, 00000\}$

دانشکده مهندسی کامپیوتر و فناوری
اطلاعات دانشگاه صنعتی امیر کبیر

Subject:
Year. Month. Date. ( )

3:30 3830

$\Sigma = \{0, 1\} \Rightarrow$ concats کردن Σ ها (point 1

$\Sigma\Sigma = \Sigma^2 = \{00, 01, 10, 11\} \Rightarrow$ رشته‌هایی که از 0 و 1 تشکیل شده باشند و دارای طول 2 باشند

$\Sigma\Sigma^2 = \Sigma^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$

$\vdots$

$\Sigma\Sigma^{n-1} = \Sigma^n = \{ \checkmark \} \Rightarrow$ رشته‌هایی که از 0 و 1 تشکیل شده باشند و دارای طول n باشند

$\Sigma^* \Rightarrow$ star closure of Σ ، Σ بستار ستاره

$\Sigma^* = \bigcup_{i=0}^{\infty} \Sigma^i = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots$

یعنی تمام مجموعه‌های دیگر هم ریخته می‌شود

Σ^* ← مجموعه‌ی تمام رشته‌هایی که از صفر یک تشکیل شده باشند. (شرط طول در این جا برداشته می‌شود)

positive closure of Σ $\quad \Sigma^+ = \bigcup_{i=1}^{\infty} \Sigma^i = \Sigma^1 \cup \Sigma^2 \cup \ldots$ بستار مثبت Σ

مجموعه تمام رشته‌هایی که از صفر یک تشکیل شده باشند و دارای طول برابر مساوی یک باشند

$\Longrightarrow \Sigma^* = \Sigma^+ \cup \{\lambda\}$ و $\Sigma^+ = \Sigma^* - \{\lambda\}$

مکمل متمم یک زبان. $(\bar{L})$

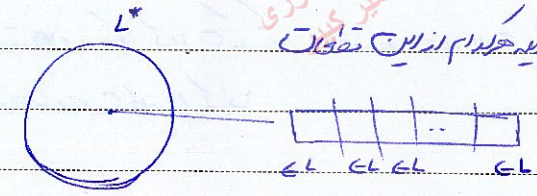
$\bar{L} = \Sigma^* - L$ متمم یک زبان که روی Σ تعریف شده باشد.

L اگر روی Σ تعریف شده باشد زیر مجموعه‌ای Σ^* (زیر مجموعه‌ای) است.

بستار یک زبان (closur)

$L^* = \bigcup_{i=0}^{\infty} L^i = \overset{\lambda}{L^0} \cup \overset{L}{L^1} \cup L^2 \cup \ldots$

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر



هر مجموعه در L^* را می‌توان به قطعاتی شکست به طوری که هر کدام از این قطعات عضوی است از L

$LL = \{xy \mid x \in L \text{ and } y \in L\}$ $\Rightarrow$ بنابراین

$LLL = \{xyz \mid x \in L \text{ and } y \in L \text{ and } z \in L\}$

Subject:
Year. Month. Date. ( )

اگر $L=\{001, 100, 01\}$ ، آیا $1000101111000 \overset{?}{\in} L^3$ هست یا نه؟

در L^3 هست اگر باز هم بتوانیم آنرا به سه قسمت تبدیل کنیم پس نتیجه می‌گیریم در L^3 هم هست

1|001|01|1001|0 ⟵

هم متعلق به L^2 و هم متعلق به L^1 ⟵ 0|01

اگر به دو قسمت توانستیم بشکنیم

← λ هم می‌تواند به صفر قطعه تبدیل کرد پس عضو L^* است.

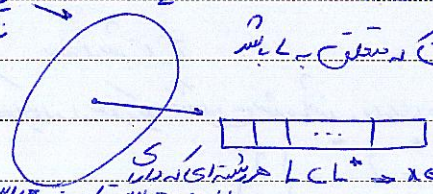
← یک مجموعه نامتناهی ⟵ $L=\{x \mid x\in\{0,1\}^*, |x| \bmod 2 = 0\}$

طول هر کدام زوج باشد

متشکل از صفر و یک باشد

آیا $L \overset{?}{=} L^*$

این هم یک مجموعه نامتناهی



یک عضو اختیاری از L^* را بردارید می‌شود به قطعاتی شکست که متعلق به L باشد

برای اثبات $L=L^*$ باید ① $L\subset L^* \to x\in L \to x\in L^*$ هر رشته‌ای که دارای طول زوج است عضو L^* است

② $L^*\subset L \to x\in L^* \to x\in L$ هر عضوی از L^* دارای طول زوج است

① بدیهی است و از تعریف می‌دید پس همیشه برقرار است $L\subset L^*$ است

② برای اثبات این عضوی از L^* انتخاب می‌کنیم به قطعاتی می‌شکنیم که عضو L هستند هر رشته در L دارای طول زوج است، رشته‌ای داریم که از به هم چسبیدن قطعاتی با طول زوج درست کردیم پس خودش هم دارای طول زوج است کل همان عضو صدایی را دارد که جزء دارد.

رشته اصلی

قطعات

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

در مجموعه‌های نامتناهی با چک کردن تک تک عضوها می‌توان اثبات کرد ولی در مجموعه‌های نامتناهی چون بی‌نهایت عضو دارند نمی‌شود این کار را کرد.

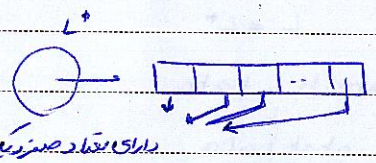
Subject:
Year. Month. Date. ( )

مثال) $L=\{x \mid x\in\{0,1\}^*, n_0(x)=n_1(x)\} \Rightarrow L \overset{?}{=} L^*$

تعداد یک‌ها — تعداد صفرها در رشته x

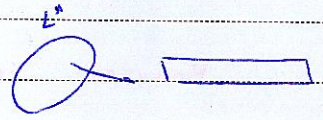
$L=L^* \leftarrow L\subset L^*$ ، $x\in L \Rightarrow x\in L^*$ ، هر رشته‌ای دارای تعداد مساوی ۰ و ۱ باشد عضوی از L^* است

$L^*\subset L$ ، $x\in L^* \Rightarrow x\in L$ ، هر عضوی از L^* دارای تعداد مساوی ۰ و ۱ باشد



رشته‌ای متشکل از قطعاتی با تعداد صفر یک مساوی ⇐ خودش دارای تعداد صفر یک مساوی

مثال) $L=\{x \mid x\in\{0,1\}^*, |x| \bmod 2 = 1\}$ $\quad L \overset{?}{=} L^*$



عضوی از پس در L^* ⇐ 001

در L^* است پس دارای جفتی ⇐ 001001

طول زوج است در L نیست

001 ∈ L
0 ∈ L
⇒ 0010 ∈ L^* ⇐ دارای طول زوج در L نیست

در این جا مثال نقض پیدا کردیم

مثال) $L=\{x \mid x\in\{0,1\}^*, |x| \bmod 2 = 0 \text{ or } |x| \bmod 3 = 0\}$ (and)

طولش یا مضربی از 2 یا مضربی از 3 است

رشته‌ای با طول 2 و 3 در L است ولی وقتی concate می‌کنیم طولش 5 می‌شود که یک عضو L نیست

← and از جمع‌بندی رشته‌های با طول 6 رشته‌ای با طول مضرب 6 تشکیل می‌شود پس and این‌ها در $L=L^*$ برقرار است.

$L=\{x \mid x\in\{0,1\}^*, |x| \text{ is a prime}\}$

$|x| \Rightarrow$ عدد اول 3
$|x| \Rightarrow$ عدد اول 3 $\Rightarrow |x\cdot x| \Rightarrow 6 \Rightarrow$ اول نیست

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

مثال) $L = \{x \mid x = uv \ , \ u,v \in \{a,b\}^* , \ |u| = |v|\}$ $\Rightarrow$ $L \overset{?}{=} L^*$

این تمرین دیگری برای بیان زوج است پس در نتیجه $L = L^*$ برقرار است.

شکستن یک رشته به دو قسمت مساوی تضمین می‌کند که طول رشته زوج باشد

$L = \{x \mid x = uv \ , \ u = v^R , \ u,v \in \{a,b\}^*\}$ $\Rightarrow$ $L \neq L^*$

$L = \{x \mid x = uv \ , \ u,v \in \{a,b\}^* , \ v = u\}$ $L \neq L^*$

علاوه بر تساوی طول خودشان هم باید مساوی باشند $abab \ , \ baba$

$abab \ baba$ ← دیگر خودشان مساوی نیستند

جلسه سوم:

عملگر معکوس: Reversal

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

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

$L = \{011, 101, 110\} \longrightarrow L^R = \{110, 101, 011\}$

مسئله) $L_1 = \{x \mid x \in \{0,1\}^* , \ |x| \bmod 2 = 0\}$, $\Sigma = \{0,1\}$

$L_2 = \{x \mid x \in \{0,1\}^* , \ |x| \bmod 2 = 1\}$

$L_3 = \{x \mid x \in \{0,1\}^* , \ |x| \bmod 3 = 1\}$

$L_1 . L_2 = \{xy \mid x \in L_1 \text{ and } y \in L_2\}$

$L_1 . L_2 \overset{?}{=} L_2$ $\Rightarrow$ تمام کلمه‌های بدست آمده این کلمه چیزی جز L_2 نیست

رشته‌های دارای طول زوج ↳ طول زوج + طول فرد = طول فرد

$L_1^* . L_2 = L_2$ $\Rightarrow$ L_1^* همان L_1 است پس مسئله بالایی است

$L_1^* = L_1$

$L_1^* \cup L_2 = \{0,1\}^*$

$L_1 \rightarrow$ طول زوج

$L_2 \rightarrow$ " فرد

$\Rightarrow$ هر رشته‌ای چون هر رشته طول زوج دارد یا فرد

تمام کلمه‌های تشکیل شده از رشته‌هایی که از چپ و راست شکل شده‌اند

دانشگاه صنعتی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

$L_1^R \cup L_2^R = L_1 \cup L_2$

L_1^R ← مجموعه تمام رشته‌های با طول فرض اگر بخواهیم هر رشته را بنویسیم چون طولش زوج است در L_1 است پس $L_1^R = L_1$

L_2^R ← به همین دلیل گفته شده $L_2^R = L_2$

$(\underbrace{L_1^*}_{L_1})^R \cup L_2 = L_1 \cup L_2$

L_1

$(\underbrace{(L_1^*)^R}_{L_1})^* = L_1$

L_1

گرامر (Grammer) (تعیین زبان)

تا چه قواعدی چگونه رشته‌ها را بدست می‌آوریم

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

V: مجموعه متغیرها (variable ، non_Terminal) غیرپایانی، حروف بزرگ انگلیسی A, B, X, Y

T: مجموعه علامات پایانی (Terminal) ← با حروف کوچک انگلیسی a, b, 0, 1 (همان Σ)

P: مجموعه قواعد production ، Rules ، Rewritting Rule)

$\alpha \rightarrow \beta$ α می‌تواند با β جایگزین شود، β می‌تواند جایگزین α شود (α می‌رود به β)

S: $S \in V$ ، (Starting Symbole)

مثال) گرامری که یک زبان را تعریف کند

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

$V = \{S, A, B\}$ ← علامت اولیه

$T = \{a, b\}$

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

$S = S$ (Starting Symbol)

فرایند جایگذاری با شروع از اولین دیجیت که می‌تواند رشته متغیری وجود داشته باشد به رشته حاصل پایان می‌یابد

یک نهایت رشته است

1	$S \Rightarrow AB$	4	$\Rightarrow aaaB$
2	$\Rightarrow aAB$	5	$\Rightarrow aaabB$
3	$\Rightarrow aaAB$	6	$\Rightarrow aaabb$

اشتقاق derivation for aaabb

زنجیره‌های میانی Sentential Form

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

$\Rightarrow$ ← منظور اشتقاق در یک مرحله

$\overset{*}{\Rightarrow}$ ← اشتقاق در صفر یا بیشتر مرحله

$AB \overset{*}{\Rightarrow} aaabb$ ⇝ aaabb قابل رسیدن در مراحل AB

جمله‌ای هم نیست

← کوتاه‌ترین AB را برای مثال به دست بیاوریم

$S \Rightarrow AB$

$\Rightarrow aB$

$\Rightarrow ab$

اگر دو تا قاعده رشته باشیم که سمت چپشان یکی باشد می‌تواند با هم یکی کنیم مثلاً

$A \to a$
$A \to aA$ $\Rightarrow$ $A \to a \mid aA$ (or)

$B \to b$
$B \to bB$ $\Rightarrow$ $B \to B \mid bB$ (or)

این یک روش تولید است ← یک الگوریتم است. ← مسئله

یک زیرمجموعه

$S \to AB$ ← رشته‌هایی به طول مثال ۱۰ که داریم از concatnation رشته‌های از لانه A و رشته‌های از لانه B تشکیل شده

A ← تمام رشته‌هایی که اگر از A شروع کنیم به آنجا خواهیم رسید. رشته‌هایی که با a شروع می‌شوند و رشته‌ای از نوع a تعداد ی خودش به مثال دارد.

$A \to$

$A \Rightarrow a$

$A \Rightarrow aA, \Rightarrow aa$

$A \Rightarrow aA, \Rightarrow aaA, \Rightarrow aaa$

B ← B هم مثل A یک تعریف بازگشتی است، همان تعداد ی b دارد، در نتیجه کل زبان می‌شود

تفاوت ربطی به هم ندارند

$L = \{a^n b^m \mid n, m \geq 1\}$ ⇝ روش بهتری برای بیان زبان رسمی (formal)

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

مثال)

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

$V = \{S, A, B\}$

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

indirect recursion

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

C ← شرط سوم

$S = S$

| A | B | ⇒ مجاز یعنی تعدادی طبیعی هم

$A \Rightarrow a$

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

$A \Rightarrow aC,\ \Rightarrow aaA,\ \Rightarrow aaaC,\ \Rightarrow aaaaA,\ \Rightarrow aaaaa$

تعداد a ها زوج خواهد بود

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

شرط جدید ← $C \to a$

در این شرط باید دارای تعداد زوجی a خواهیم بود (احتمالاً)

$L_1 = \{a^{2k} b^n \mid k \geq 1,\ n \geq 1\}$

$L_S \to$ مجموعه تمام رشته‌هایی که اگر از S شروع کنیم به آن می‌رسیم

$L_S = L_A . L_B$

رشته‌های شروع شده با A ↳ رشته‌های شروع شده با B

$L_A = \{a\} L_C$ ⇒ رشته‌هایی که از C شروع کنیم به آن می‌رسیم

$L_C = \{a\} L_A$

↳ یک زبان C ساده‌تر دیگر (زیرزبان C)

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

PAPCO

Subject:
Year. Month. Date. ( )

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

یک CFG برای این زبان بنویسید ← بی‌نهایت رشته ← در روابط حداقل یک رابطه بازگشتی وجود داشته باشد

روابط غیر پایانی تنها که نیاز تولید بی‌نهایت رشته است. ← اولین مرحله اینست که یک تعریف پایانی پیدا کنیم

تعریف 1

یکی از اینها خواهد بود $\begin{cases} aa \\ ab \\ ba \\ bb \end{cases}$

اگر رشته‌ای دارای طول زوج باشد یا رشته‌ای با طول 2 تشریع می‌شود و ادامه آن رشته‌ای با طول زوج است که طولش بزرگتر از صفر باشد

$A \to ab \mid bb \mid ba \mid aa$

$S_e \to AS \mid ab \mid bb \mid ba \mid aa \to (S \to abS \mid bbS \mid baS \mid aaS \mid ab \mid bb \mid ba \mid aa)$

قسمت پایانی تعریف — قسمت پایه تعریف (Base)

یک تعریف زبان است با طول زوج که طول صفر را شامل نمی‌شود.

$S_e \to \lambda$ ← چون لازم برای تولید رشته‌ای با طول صفر

تعریف 2

یا رشته‌ای با طول زوج تشریع می‌شود و در آخر رشته‌ای با طول 2 می‌آید

$A \to ab \mid bb \mid ba \mid aa$

$S_e \to SA \mid ab \mid bb \mid ba \mid aa$ (زوج)

تعریف 3

$S_e = S_1 \mid S_2$

S_1: زوج | زوج یا S_2: فرد | فرد

هر رشته با طول زوج دارای یکی از این دو خاصیت است

تعریف برای رشته‌ای با طول زوج داشتیم باید برای طول فرد تعریفی پیدا کنیم الان باید این را حل کنیم

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

زوج | رشته با طول 1 (a یا b)

$S_o \to aS_e \mid bS_e$

$S_e \to abS_e \mid bbS_e \mid baS_e \mid aaS_e \mid ab \mid bb \mid ba \mid aa$ — راهی برای رشته‌های با طول فرد

$S_e \to \lambda$ ← برای تولید رشته‌ای با طول 1

PAPCO

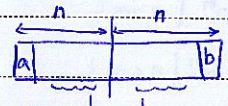
دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

جلسه چهارم:

$L = \{a^n b^n \mid n \geq 1\}$ ← تمام رشته‌هایی که به تعداد a ها به دنبال b ها به دنبال تعداد برابر آن است

بعنوان اولین قدم ← یافتن یک تجزیه برگشتی



دارای عمل حضور میان تولید اول

پایه برگشت

$S \rightarrow aSb \mid ab$

برای تولید $aaabbb$ ←

$S \Rightarrow aSb$
$\Rightarrow aaSbb$
$\Rightarrow aaabbb$

← چطوری فهمیم گرامری که نوشتیم صحیح است؟

باید نشان دهیم $L = L(G)$ ← زبان مورد نظر
زبان تولید شده توسط گرامر G

$L \subseteq L(G)$ ، $x \in L \Rightarrow x \in L(G)$ ، هر رشته‌ای از L توسط G تولید می‌شود

عضوی از $L(G)$ است

$L(G) \subseteq L$ ، $x \in L(G) \Rightarrow x \in L$ ، هر رشته‌ای که توسط G تولید می‌شود عضوی از L می‌باشد

دارای حضور عمومی یا است

$L = L(G)$

I ← اسکی بیچی راست چپ تنها از S شروع می‌کنیم با asb یا ab جایگزین می‌شود پس دارای تعداد مساوی ab است.

I ← اگر چیزی بیشتر یا کمتر تولید کند دیگر در زبان مورد نظر نیست

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

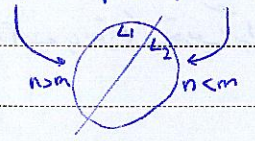
Subject:

Year. Month. Date. ( )

مثال: تعداد رشته‌های با تعداد a و b که تعداد آنها با هم برابر نیست

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

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



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

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

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

$= \{a^p a^m b^m \mid m \geq 0, p \geq 1\}$ → از concat دو رشته حاصل می‌شود ۱- دارای تعداد a ۲- به صورت رشته تعلق

$a^p \cdot (a^m b^m)$

$S_1 \rightarrow AB$

$A \rightarrow aA \mid a$ ← برای تولید تعداد a های مختلف

$B \rightarrow aBb \mid \lambda$

لامبدا

$L_2 = \{a^n b^m \mid n < m\}$ ⟹ $S_2 \rightarrow CD$

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

$C \rightarrow aCb \mid \lambda$

$D \rightarrow bD \mid b$

شرح کلی:

$S \rightarrow S_1 \mid S_2$

$S_1 \rightarrow AB$

$A \rightarrow aA \mid a$

$B \rightarrow aBb \mid \lambda$

$S_2 \rightarrow CD$

$C \rightarrow aCb \mid \lambda$

$D \rightarrow bD \mid b$

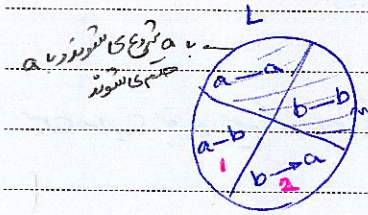
دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی شریف

Subject: Toalid

Year. Month. Date. ( )

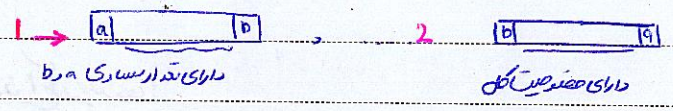
مثال) در یک a و b های تولید mixed باشند

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



این زبان در حقیقت اجتماع ۲ زبان است.

این دو دارای توان به دو قسمت شکست که دارای خصوصیت زبان کل باشند

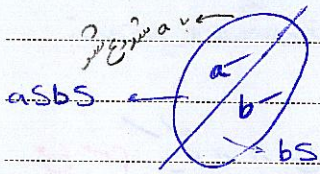


recursive base

$S \to \underbrace{aSb \mid bSa \mid SS}_{} \mid \lambda$

راه حل دیگر:

اگر رشته‌ای با a شروع شود و تعداد حرف a با b ها برابر باشد حتماً یک b پیدا می‌شود که بین دو رشته با تعداد a و b مساوی محصور شده و بالعکس



$S \to aSbS \mid bSaS \mid \lambda$

تمرین: دو گرامر G_1 و G_2 معادل (equivalent) هستند اگر $L(G_1) = L(G_2)$ و با شرط‌های مجموعه‌ها داریم:

$L(G_1) \subset L(G_2)$: $x \in L(G_1) \Rightarrow x \in L(G_2)$ ، هر رشته‌ای که توسط G_1 تولید شود توسط G_2 هم تولید می‌شود

$L(G_2) \subset L(G_1)$: $x \in L(G_2) \Rightarrow x \in L(G_1)$ ، هر رشته‌ای که توسط G_2 تولید می‌شود توسط G_1 هم تولید می‌شود

$L(G_1) = L(G_2)$

دانشکده مهندسی کامپیوتر و فناوری اطلاعات
دانشگاه صنعتی امیرکبیر

G_1: $S \to aSb \mid bSa \mid SS \mid \lambda$

G_2: $S \to aSb \mid bSa \mid SS \mid SSS \mid \lambda$

G_2 همه‌ی قواعد G_1 را دارد پس هرچی G_1 تولید کند G_2 هم می‌تواند تولید کند

آیا هرچی که G_2 تولید کند G_1 هم تولید می‌کند؟ (G_2 قاعده SSS را دارد که G_1 ندارد باید اثبات کنیم)

Subject:
Year. Month. Date. ( )

می‌توانیم نشان دهیم SSS را می‌توانیم با قواعد G_1 بدست آوریم

G_1: $S \Rightarrow SS \Rightarrow SSS$ ⇒ در دو مرحله به SSS می‌رسید

G_2: $S \Rightarrow SSS$

پس به این نتیجه می‌رسیم که هر چیزی که G_2 تولید کند G_1 هم تولید می‌کند

جلسه پنجم : 7,12

انواع گرامرها:

گرامرها به چند نوع تقسیم می‌شوند
- نوع سوم (منظم) Regular
- نوع دوم (مستقل از متن) context free
- نوع اول (حساس به متن) content Senselive
- نوع صفر (بدون محدودیت)

تفاوت این چهار شکل قواعدی است که در هر یک قواعد مراحلیم درست

نوع سوم: یک گرامر از نوع سوم (منظم) است اگر قواعد به شکل زیر باشند

متغیر سمت راست در انتهای سمت راست ظاهر شود

خطی از سمت راست: $A \rightarrow aB$ | خطی از سمت چپ: $A \rightarrow Ba$ | $A, B \in V$

right linear: $A \rightarrow a$ | left linear: $A \rightarrow a$ | $a \in T$

گرامرهایی از نوع سوم:

$S \rightarrow Sa \mid Sb \mid a \mid b$ ⇒ خطی از سمت چپ

در همه موارد متغیر انتهای راست

- این مثال چیزهای را تولید می‌کند

⇒ تمام رشته‌هایی با طول غیرصفر بسازید که از a و b تشکیل شده‌اند

دانشکده مهندسی کامپیوتر و فناوری
اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

علامت نهایی — متغیر

$S \to aA \mid bB \mid a$ ⟵ یک حرفی از سمت راست

$A \to aA \mid b$

$B \to bB \mid a$

زبان تولید شده : $\{a\} \cup \{aa^nb \mid n \geq 0\} \cup \{bb^na \mid n \geq 0\}$

نوع دوم (مستقل از متن) اگر قواعد به فرم زیر باشند

$A \to \alpha$ ⟵ سمت چپ باید یک متغیر باشد

$A \in V$ و $\alpha \in (V \cup T)^*$

α جزء شده‌هایی است از متغیر و علامت‌های پایانی تشکیل شده است، و هم‌ستاره می‌شود

گرامری از نوع دوم

$S \to SS \mid aSb \mid bSa$

نوع اول (حساس به متن) اگر قواعد به فرم زیر باشند

$\alpha \to \beta$ ⟵ طول سمت چپ حداکثر برابر طول سمت راست باشد

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

خط عنوان جمله‌ای که حش پیدا می‌کند یا ثابت می‌ماند یا بیشتر می‌شود اگر که حش پایه ممکن است نوع جمله‌ای که حش پایه (مشترک)

گرامری از نوع اول

1 $S \to abc$

2 $S \to aXbc$ ⟵ α می‌تواند هر چیزی باشد به غیر از λ $(\alpha \in (V \cup T)^+)$

3 $Xb \to bX$

4 $Xc \to Ybcc$ ⟵ حساس به متن ⟵ X می‌تواند با Ybc جایگزین شود مگر اینکه در این سمت قرار گیرد

5 $bY \to Yb$

6 $aY \to aaX$

7 $aY \to aa$ ⟵ هر دو تعداد مساوی است

زبانی که تولید می‌کند $= \{a^nb^nc^n \mid n \geq 1\}$

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی شریف

Subject:
Year. Month. Date. ( )

تولید رشته aaaabbbbcccc توسط گرامر مثال (گرامر نوع اول) : از 2 چون اگر 1 باشد abc به دست می‌آید

$S \Rightarrow aXbc$; $\overset{3}{\Rightarrow} abXc$; $\overset{4}{\Rightarrow} abYbcc$, $\Rightarrow aYbbcc$

, $\Rightarrow aaXbbcc$, $\Rightarrow aabXbcc$, $\Rightarrow aabbXcc$

, $\Rightarrow aabbYbccc$, $\Rightarrow aabYbbccc$, $\Rightarrow aaYbbbccc$

, $\Rightarrow aaaXbbbccc$, $\Rightarrow aaabXbbccc$, $\Rightarrow aaabbXbcccc$

, $\Rightarrow aaabbbXccc$, $\Rightarrow aaabbbYbcccc$, $\Rightarrow aaabbYbbcccc$

, $\Rightarrow aaabYbbbcccc$, $\Rightarrow aaaYbbbbcccc$, $\Rightarrow aaaabbbbcccc$

اثبات صحیح بودن گرامر ←

① نشان دهید $S \overset{*}{\Rightarrow} a^n X b^n c^n$.

← جمله‌هایی به این شکل از S قابل وصل هستند

برای اثبات از استقرا استفاده می‌کنیم

فرض : $S \overset{*}{\Rightarrow} a^n X b^n c^n$

حکم : $S \overset{*}{\Rightarrow} a^{n+1} X b^{n+1} c^{n+1}$

طی یک مرحله

پایه استقرا $\Leftarrow n=1$: $S \Rightarrow aXbc$ ←

قسمت برگشتی $S \overset{*}{\Rightarrow} a^n X b^n c^n$

$S \Rightarrow a^n X b^n c^n \overset{*}{\Rightarrow} a^n b^n X c^n = a^n b^n X c c^{n-1} \Rightarrow a^n b^n Y b c c c^{n-1} = a^n b^n Y b c^{n+1}$

$\Rightarrow a^n Y b^{n+1} c^{n+1} = a^{n-1} a Y b^{n+1} c^{n+1} \Rightarrow a^{n-1} aaX b^{n+1} c^{n+1}$

$\Rightarrow a^{n+1} X b^{n+1} c^{n+1}$

② نشان دهید $S \overset{*}{\Rightarrow} a^n b^n c^n$

← با استفاده از قضیه قبل برای استفاده از 6 از 7 در آخر استفاده می‌کنیم

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

نوع صفر (بدون محدودیت) اگر قواعد به فرم زیر باشند

مثل گرامر نوع اول بدون محدودیت طول

$\alpha \to \beta$

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

گرامری از نوع صفر:

$S \to ABaC$

$Ba \to aaB$

$BC \to DC \mid E$

$aD \to Da$

$AD \to AB$

$aE \to Ea$

$AE \to \lambda$ ... طول β کمتر از α

زبانی که تولید می‌کند: $L = \{a^{2^n} \mid n \geq 1\}$

تمرین: گرامری برای این زبان بنویسید

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

انواع زبان‌ها:

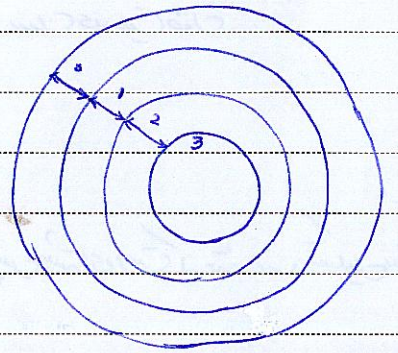
زبان‌ها هم به ۴ دسته تقسیم می‌شوند:

- نوع سوم (منظم) زبانی که بتوان برای آن گرامر نوع سوم نوشت
- نوع دوم (مستقل از متن) زبانی که بتوانیم گرامر نوع سوم آن را نوع دوم بنویسیم
- نوع اول (بتوانیم نوع سوم و دوم بنویسیم. نوع اول به دو بنویسیم)
- نوع صفر بتوان گرامری نوع سوم، دوم، اول نوشت

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

Hierarchy of Languages — سلسله مراتب زبان ها



هر زبانی که در نوع 3 دارد توسط نوع 2 پوشش داده می شود

$L_3 \subseteq L_2 \subseteq L_1 \subseteq L_0$

ناشی از تقارنی برای آن وجود دارد

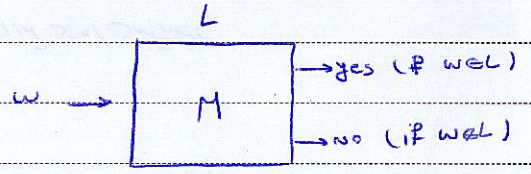
پس غلط نیست بگوییم هر زبان نوع 3 نوع 2 است ... ← با توجه به تقارنی برای آن ها

یک نام خانواده یک زبان که چندین خانواده ای که به آن متعلق است، است

جلسه ششم 7.15

ماشین ها: Automaton - Automata

فرض کنیم زبانی مانند L و رشته w باشیم. ماشین رشته ای را می گیرد و می گوید آیا عضو L است یا نه



یک پذیرنده چیزی نیست جز یک الگوریتم عضویت (membership). عضویت رشته را در آن زبان چک می کند

↳ Recognizer (تشخیص دهنده)

انواع پذیرنده ها:

- پذیرنده زبان های نوع سوم — Finite State Atuomata
- پذیرنده برای زبان های نوع دوم — push down Atuomata
- پذیرنده برای زبان های نوع اول — Linear Bounded Atuomata
- پذیرنده برای زبان نوع صفر — Taring Machine

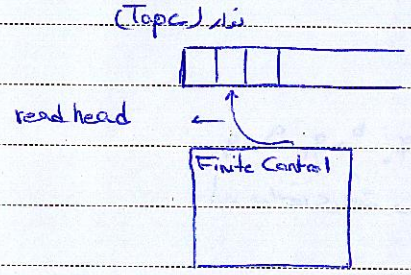
دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

فصل 2:

پذیرنده برای زبان های نوع سوم: Finite state Automata

این پیشین از سه جزء تشکیل شده. یک نوار (Tape) که readOnly است و semi-infinit (یک طرف بسته و یک طرف نامتناهی است)



جزء دوم: Finite Control

از طریق read head اطلاعات را از نواری خواند

از بین حالت های مختلف متناهی همیشه تنها در یک حالت

است و از روی چیزهایی که می خواند حالت خود را تغییر می دهد.

رشته را در Tape قرار می دهیم، علامت به علامت می خواند و حالت خود را تغییر می دهد، اگر در انتهای رشته در یکی از حالت های نهایی باشد رشته مورد قبول است.

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

Q: مجموعه حالات set of state

Σ: الفبا (که علامت جزء این مجموعه هستند)

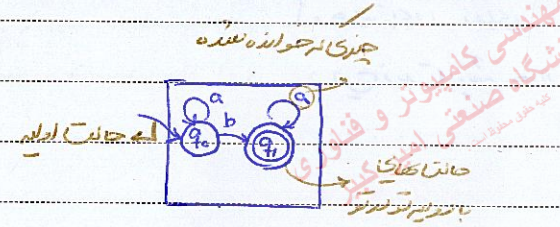
δ: $\delta : Q \times \Sigma \to Q$ ← تابع انتقال Transition Function

روش های مختلف برای نشان دادن δ است: 1. Graph 2. table 3. تابع

q_0: حالت اولیه starting state

F: $F \subseteq Q$ ، مجموعه حالات نهایی Set of Final state

روش نشان دادن توسط Graph:



دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

توسط table:

	a	b
q_0	q_0	q_1
q_1	q_1	—

Subject:
Year. Month. Date. ( )

توسط تابع:

$\delta(q_0, a) = q_0$

$\delta(q_0, b) = q_1$

$\delta(q_1, a) = q_1$

یک مثال:

q_0

a	b	a

Read head

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

در انتهای رشته در حالت نهایی هستیم پذیرش اعلام می‌شود.

a	b	a	b

$q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_1 \xrightarrow{a} q_1 \xrightarrow{b} \times$ عدم پذیرش

در این جا b را می‌خواند و نمی‌تواند حرکت کند در متوقف می‌شود و پذیرش اتفاق نخواهد افتاد.

پذیرش: یک رشته پذیرفته می‌شود اگر در انتهای آن ماشین در یکی از حالات نهایی خود قرار گیرد.

عدم پذیرش: یک رشته پذیرفته نمی‌شود اگر در انتهای آن در یکی از حالات غیر نهایی خود قرار گیرد یا قبل از رسیدن به انتهای رشته متوقف شود.

می‌خواهیم ببینیم این ماشین را که مثال زدیم چه زبانی را به ما می‌دهد؟ تبدیل به مسئله‌ای دیگر می‌شود و مسیرهایی را که منجر به رسیدن به حالات نهایی می‌شود و label های آن می‌شود رشته ما.

$q_0 \underset{\text{loop}}{\sim} q_0 \xrightarrow[b]{} q_1 \underset{\text{loop}}{\sim} q_1 \Rightarrow L = \{a^n b a^m \mid n, m \geq 0\}$

تعدادی loop بزند — تعدادی loop بزند

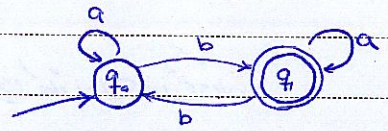
هر رشته‌ای که دارای این فرمت باشد جواب yes را از ماشین می‌گیرد.

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

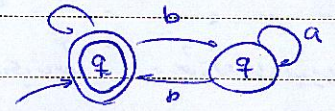
مثال) باید فرمولی

I



هر رشته با تعداد زوج b ، اگر تعداد b ها فرد باشد در کمت رست مستقر خواهد شد.
پس مجموعه تمام رشته‌هایی که هر کدام دارای تعداد زوج b باشند را a و b تشکیل شده باشند

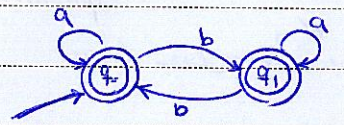
II



در این حالت باید تعداد زوج b باشد تا در حالت نهایی قرار گیرد
تمام رشته‌هایی که از a و b باشد و تعداد b ها زوج باشد

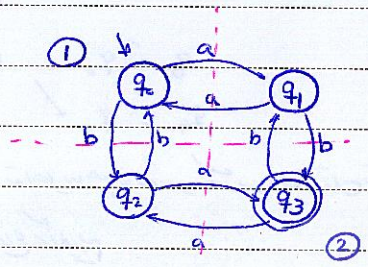
$L_2 = \{ w \mid w \in \{a,b\}^*, n_b(w) \bmod 2 = 0 \}$

III



$\rightarrow L = L_1 \cup L_2 = \{a,b\}^*$

point) از زیاد تعداد حالت های نهایی زبان بزرگتر می‌شود



مثال) مسیرهایی که با a از q_0 به q_3 برسند
مسیرهایی که از 1/4 ① شروع و به 1/4 ② بیایان می‌پذیرند
مرز بین متغیری محور را قطع می‌کند خط افقی ← b
خط عمودی ← a

در هر حالتی باشد می‌توانند حرکت کنند همیشه خود را به انتهای رشته می‌رسانند.
برای اینکه به 1/4 ② برسد باید هر دو محور را به تعداد فردی قطع کند

$L_{q_3} = \{ w \mid w \in \{a,b\}^*, n_{(a)} \bmod 2 = 1, n_{(b)} \bmod 2 = 1 \}$

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

دانشکده مهندسی کامپیوتر و فناوری
اطلاعات دانشگاه صنعتی شریف

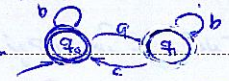
Subject:
Year. Month. Date. ( )

	a	b
q_3	زد	زد
q_1	زد	زوج
q_2	زوج	زد
q_0	زوج	زوج

با تغییر حالت نهایی کلمه زبان فرق می‌کند

$L_{q_3} \cup L_{q_1} \rightarrow$

اگر هم q_3 و q_1 حالات نهایی باشند ← دارای تعداد فردی a باشد ← معادل با

$L_{q_2} \cup L_{q_0} \rightarrow$ 

" " q_2 و q_0 " " " ← " " زوجی a "

اگر همه حالات نهایی باشند ← هر رشته‌ای را می‌پذیرد ← معادل ماشین III

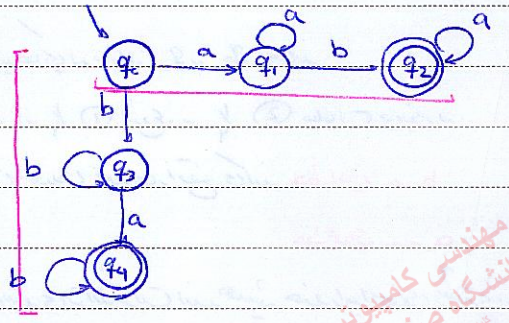
Equivalent class:

دو ماشین M_1, M_2 معادل می‌باشند اگر $L(M_1) = L(M_2)$

یک ماشین M_1 می‌پذیرد

(paint) برای یک زبان می‌توانیم بی‌نهایت ماشین طراحی کنیم ولی اینکه کدام بهتر است بحث جداگانه دارد.

چند مثال:



مسیرهای که به q_0 ← q_3

q_0 ← q_4

این دو مسیر با تداخل با هم ندارند و به دو گروه مستقل تبدیل می‌کنیم و اجتماع می‌گیریم

$$L_{q_2} = \{ a a^n b a^m \mid n, m \geq 0 \} = \{ a^n b a^m \mid n \geq 1, m \geq 0 \}$$

$$L_{q_4} = \{ b^n a b^m \mid n \geq 1, m \geq 0 \}$$

$$L = L_{q_2} \cup L_{q_4}$$

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

در این جا یک حالت را مشخص می‌کردیم که A و C را تعیین می‌کرد

$\delta : Q \times \Sigma \to Q$ (point

حالت — علامت → حالت بعدی را تعیین می‌کند

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

$\delta^*(q, w) = ?$

یک رشته

اگر از q شروع کنیم رشته w را بخوانیم کجا خواهیم بود. برای مثال رشته مقابل داریم

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

یک تعریف بازگشتی برای δ^*

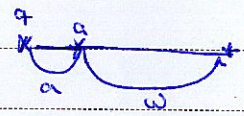
حالت اول

$z = aw$ → باقیمانده‌ی رشته

$\delta^*(q, aw) = ?$

از q شروع کنیم بعد از خواندن رشته aw کجا خواهیم بود

$\delta^*(\delta(q, a), w)$



از q بعد از برداشتن اولین قدم کجا هستیم بعد از آن با خواندن رشته w کجا هستیم

بقیه قدم‌ها را برداریم

$$\delta^*(q, aw) \begin{cases} \text{Recursive}: & \delta^*(\delta(q, a), w) \\ \text{Basis}: & \delta(q, a) \text{ if } |w| = 0 \end{cases}$$

مثال:

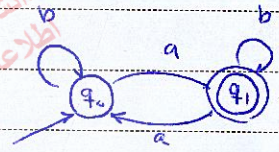
$\delta^*(q_0, \overset{a}{b}\overset{w}{ab}) = ?$

$= \delta^*(\underbrace{\delta(q_0, b)}_{q_0}, \overset{a}{a}\overset{w}{b})$

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

$= \delta^*(\underbrace{\delta(q_1, b)}) = q_1$

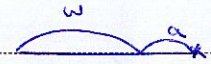
به پایه رسیدیم



دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

یک تعریف بازگشتی دیگر:



$Z = wa$

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

این همان حقیقت اولین صرایه خود رویه است.

$\delta^*(q_0, bab) = ?$

$= \delta(\delta^*(q_0, ba), b) = q_1$

$\delta(\delta^*(q_0, b), a) = q_1$

$\delta(q_0, b)$

q_0

$\delta^*(q_0, bab) = q_1$

پس می‌توانیم نتیجه بگیریم که:

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

تعریف بازگشتی دیگر:

$\Rightarrow \begin{cases} \delta^*(\delta^*(y, u), v) = \delta^*(q, uv) \\ \text{basis: } ? \end{cases}$

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر
کلیه حقوق محفوظ است

Subject:
Year. Month. Date. ( )

جلسه هفتم 7,20

مثال) نحوه طراحی یک پذیرنده برای یک ماشین

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

ابتدا یک شبکه از چپ به راست علامت را می‌خوانند

(علامت را جمع آوری می‌کند تا در آخر بتواند به راست برود

از اضلاع نتیجه گیری کند

پس صفر یک در حرکت است و زوجیت خواندن ← $c=0$

رشته با هر خواندن c را مقدار دهی تغییری بدهد. اگر $c=0$ باشد

طول رشته زوج خواهد شد. هرچه c باشد در آخر باقی مانده طول بر 2 است.

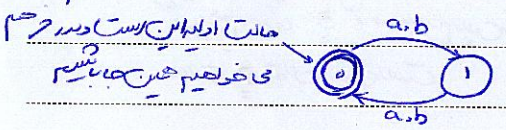
کاری که باید بکنیم این است که این ایده را به الگوریتم تبدیل کنیم:

همین طور که حرکت می‌کنیم c را نظاره می‌کنیم و در آخر c هر چی بود جواب برای ما بدهد

در هر لحظه دو حالت باید پیگیری شود ← $c=0$ ← اگر چه تا بحال خواندم طولش زوج بوده

$c=1$ ← ... فرد بوده

ماشینی که خواهیم رسم در حالت چه را دارد.

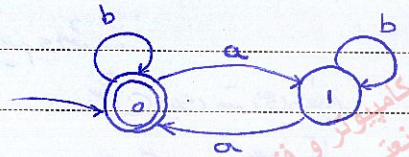


مثال)

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

اینبار c را برای رشته باقی مانده تعداد aها بر 2 استفاده می‌کنیم این جا چون تعداد aها ملاک است اگر a خواندیم

c را به مقدار دهی تغییر می‌دهیم



b در اینجا تغییری در c ایجاد نمی‌کند پس بخواند b

state تغییری نخواهد کرد.

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

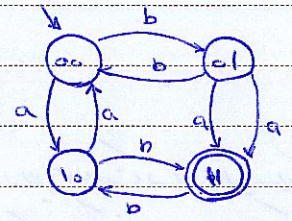
مثال) $L = \{w \mid w \in \{a,b\}^*, n_a(w) \bmod 2 = 1, n_b(w) \bmod 2 = 1\}$

دوباره از همین ایده سنزوری رشته استفاده می‌کنیم اما اینبار دو متغیر استفاده می‌کنیم:

$c_1 = 0 \rightarrow$ باقی‌مانده تعداد a ها بر 2

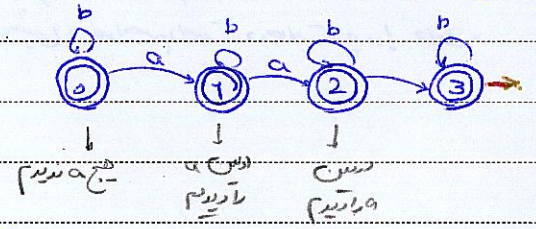
$c_2 = 0 \rightarrow$ باقی‌مانده تعداد b ها بر 2

چهار حالت هم است که در این جا بکار می‌خواهیم بنویسیم



مثال) مجموعه تمام رشته‌هایی که دارای حداکثر 3 تا a باشند

دوباره یک سنزور چپ به راست داریم که اطلاعات را استخراج می‌کنیم و نتیجه گیری می‌کنیم



اگر بگوییم قطعاً 3 تا a ببینیم پس متوقف

حالت 3 قابل قبول است

اگر در حالت 3، a را ببیند جدول برایش تعریف نشده است پس پذیرش ندارد پس اعلام عدم پذیرش می‌کند

انواع ماشین‌های متناهی (FA)

به دو گروه تقسیم می‌شود:

Deterministic finit Automata — ماشین‌های متناهی قطعی

non-deterministic finit Automata — ماشین‌های متناهی غیرقطعی

DFA:

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

اگر حالت را از علامت بدانیم قطعی می‌گوییم حالت بعدی کی است

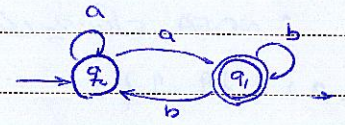
برای هر جفت از حالت و علامت یک حالت مشخص تعیین می‌شود

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

NDFA:

در این حالت مثل قبل نیست یعنی ممکن است برای یک جفت حالت و علامت چند حالت بعدی قرار بگیرد



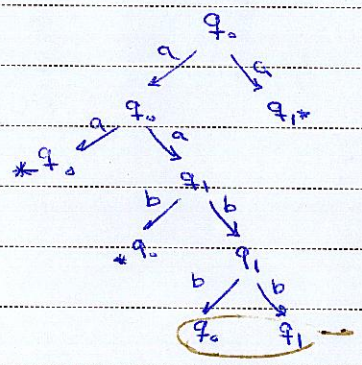
این یک پذیرنده برای یک زبان است اما چه زبانی با شکل کلی چه؟

در حالت های قبل از حالت اولیه شروع می کردیم به حالت نهایی می رسیدیم در این جا ممکن است چندین مسیر با برچسب یکسان وجود داشته باشد که حداقل یکی از آنها به حالت نهایی برسد

یک رشته پذیرفته می شود اگر حداقل یک مسیر از حالت اولیه به یکی از حالات نهایی با برچسب آن مسیری وجود داشته باشد

← برای تشخیص مسیر از درخت تصمیم گیری استفاده می کنیم یا این درخت تمام مسیرها را نشان می دهیم

برای این رشته: aabb



کاربرد هم برای تشخیص رشته در نتیجه back tracking می شود پیمایش preorder درخت.

با این که مسیر را انتخاب رسیدیم که از این راس می کی در حالت نهایی قرار داده است پس این رشته پذیرفته می شود

point: فرض می کنیم این ماشین هوشمند است اگر یک حالت بیند چند مورد می تواند باشد از این راه حل ها استفاده می کند که ماشین را به هدف پذیرفتن رشته ای برساند یا نشان بدهد

NDFA ← ماشین های غیرقطعی دو نوع است.

(without λ-moves) transition — بدون حرکات λ

(with λ-transition) — با حرکات λ

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

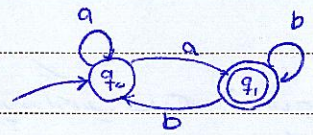
Subject:

Year. Month. Date. ( )

تعریف رسمی NDFA:

$\delta : Q \times \Sigma \to 2^Q$

تعریف برای مثال NDFA:



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

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

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

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

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

$\delta^*(q_0, aabb) = ?$

$z = aw$

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

$z = wa$

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

مثال:

$\delta^*(q_0, aabb) = \delta^*(\delta(q_0, a), abb) = \delta^*(q_0, abb) \cup \delta^*(q_1, abb)$

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

$= \delta^*(q_0, bb) \cup \delta^*(q_1, bb) \cup \{\}$

$= \delta^*(\delta(q_0, b), b) \cup \delta^*(\delta(q_1, b), b) \cup \{\}$

$= \{\} \cup \delta^*(q_0, b) \cup \delta^*(q_1, b) \cup \{\}$

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

دانشکده مهندسی کامپیوتر و فناوری
اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

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

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

حداقل یک مسیر را به حالت نهایی برساند اگر به F در بین از اجتماع اشتراک داشت یعنی
اینکه چندین مسیر را به حالت نهایی برساند که در این حالت هم صحیح است

← اگر از روش بازگشتی $z = wa$ استفاده کنیم ← درخت از پایین به بالا ساخته می‌شود ← این کار داده شود.

① چطوری شود یک ماشین NDFA را به صورت بدون تبدیل کرد به ماشین DFA امکان پذیر دهد؟

② اصلاً چرا ما خواهیم این کار را کنیم؟ پیاده‌سازی اصلی هم نیست چون فقط یکبار استفاده می‌کنیم. این کار را برای ما
تربیت بیشتر شود. کلی است چیزی که الگوریتم آن پیچیده باشد و برنامه‌نویسی زیاد باشد.

$\Theta(2^{|w|})$ هزینه زیادی دارد نسبت به حالت بسته back tracking

حالا اگر یک ماشین متناهی داشتیم هزینه‌اش کمتر بود چون تعداد قدم‌های ما که برای ما برابر با مقدار عددی است که
خوانده می‌شود $\Theta(|w|)$ ← می‌دیم برای الگوریتم‌های ما دیگر قطعی باشد
پس برای جنبه‌ی دیگر یک ماشین غیرقطعی را به یک ماشین قطعی تبدیل کند

جلسه هشتم 7,22

انواع ماشین‌های متناهی

Finite State Automata ← Determinestic

Non-determinetic ← without λ-transition

with λ-transition

without λ-transition : برای پیدا کردن جواب از یک درخت استفاده می‌کردیم و اگر مسیری از حالت اولیه به
حالت نهایی وجود داشته باشد رشته پذیرفته می‌شود.

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:

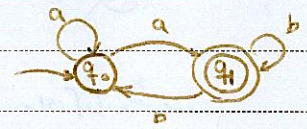
Year. Month. Date. ( )

الگوریتم های تبدیل NDFA به DFA

ورودی الگوریتم ← Given: NDFA $= (Q, \Sigma, \delta, q_0, F)$ accepting L

خروجی الگوریتم ← Output: DFA $= (Q', \Sigma', \delta', q_0', F')$ " "

زبان های یکسان را می پذیرند ⇐



→ $Q' = 2^Q$ ⇝ تمام زیرمجموعه های Q

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

← ماشینی که می سازیم چهار حالت جداگانه دارد.

→ $q_0' = [q_0]$

→ F' = Set off all states in Q' which contains at least one state in F

$F' = \{[q_1], [q_0, q_1]\}$ ⇝ هر کدام تمام کش هایی شامل حالت نهایی باشند

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

← یکی از حالات زیرمجموعه F if $\delta([q_{i_0}, q_{i_1}, \dots, q_{i_k}], a) = \{p_{j_0}, p_{j_1}, \dots, p_{j_k}\}$

δ'

	a	b
$[q_0]$	$[q_0, q_1]$	$[\ ]$
$[q_1]$	$[\ ]$	$[q_0, q_1]$
$[q_0, q_1]$	$[q_0, q_1]$	$[q_0, q_1]$
$[\ ]$ ← از هیچ q ورودی هیچ	$[\ ]$	$[\ ]$

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

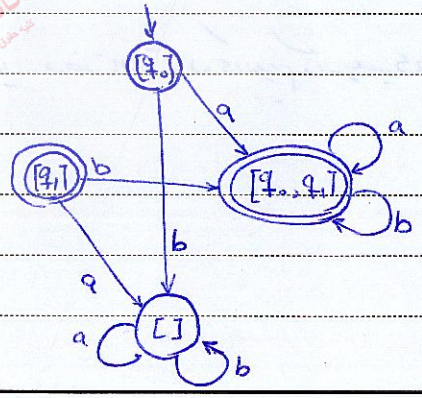
$\delta'([q_0], a) = q_0, q_1$

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

$\Rightarrow \delta'([q_0, q_1], a) = [q_0, q_1]$

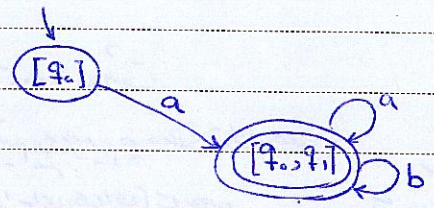
دانشکده مهندسی کامپیوتر و فناوری
اطلاعات دانشگاه صنعتی امیر کبیر

اگر بسیاری در طول هیچ مسیر از حالت اولیه حالت کنونی
قرار نگیرد به درد نمی خورد و بود و نبودش
هم نیست یعنی اگر از حالت اولیه قابل
وصل نباشد



Subject:
Year. Month. Date. ( )

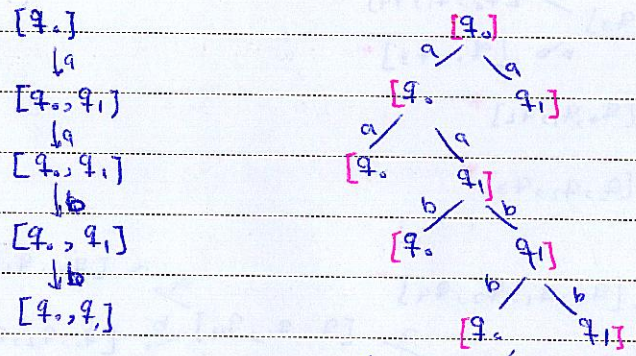
پس با این پیش تبدیل می‌شود (با حذف state هایی که بودن و نبودنشان مهم نیست)



تمام رشته‌هایی که با a شروع می‌شوند ⟸

سوال) چرا از این الگوریتم روی هر NDFA استفاده کنیم یک DFA خواهیم گرفت؟

برای رشته aabb در دو حالت NDFA و DFA درخت پایین را داریم:



اگر اینجا را هم بکشیم طوری که آرک‌ها روی هم قرار گیرند این دو یکی است

(point) اگر تعداد حالات در Q زیاد باشد، تعداد حالات در Q' خیلی خیلی بیشتر از Q خواهد بود (چون نمایی از این پیدا می‌کند) و برای کاربردهای عملی $|Q| = 40, 50$ شود عملی نیست استفاده از این روش عملی نباشد و جدول 2^{50} تا سطر خواهیم داشت. خیلی از مسائل در جدول از نوع بی‌سببی است که نتایج حذف می‌شوند و اگر برای مسائل ای‌ام بگیریم هم نیست. برای حل این دو مشکل از همین روش یک ترفند استفاده می‌کنیم:

در یک جدول هوشمند پر می‌کنیم یعنی آن‌هایی را که لازم داریم و حذف نمی‌شوند پر کنیم

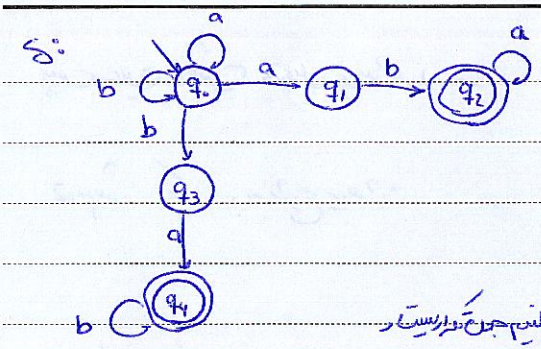
در مثال زیر توضیح داده خواهد شد.

دانشکده مهندسی کامپیوتر دانشگاه صنعتی امیرکبیر
اطلاعات و طاوری

Subject:

Year. Month. Date. ( )

مثال:



حاسبی 32 = 2^5 حالت خواهد داشت

از حالت اولیه شروع می‌کنیم و می‌بینیم چه حالاتی قابل وصول هستند بعد می‌آییم برای هر کدام انجام می‌دهیم

این را تکرار می‌کنیم چون تو در رشته‌ای

این دو حالت از q_0 قابل وصولند

تکراری بودن

$[q_0] \xrightarrow{a} [q_0, q_1]$

$[q_0] \xrightarrow{b} [q_0, q_3]$

$[q_0, q_1] \xrightarrow{a} [q_0, q_1] \cup \{\}$ *

$[q_0, q_1] \xrightarrow{b} [q_0, q_2, q_3]$

$[q_0, q_3] \xrightarrow{a} [q_0, q_1, q_4]$ *

$[q_0, q_3] \xrightarrow{b} [q_0, q_3]$ *

$[q_0, q_2, q_3] \xrightarrow{a} [q_0, q_1, q_2, q_4]$

$[q_0, q_2, q_3] \xrightarrow{b} [q_0, q_3]$ *

$[q_0, q_1, q_2, q_4] \xrightarrow{a} [q_0, q_1, q_2]$ I

$[q_0, q_1, q_2, q_4] \xrightarrow{b} [q_0, q_2, q_3, q_4]$ II

I $[q_0, q_1, q_2] \xrightarrow{a} [q_0, q_1, q_2]$ *

I $[q_0, q_1, q_2] \xrightarrow{b} [q_0, q_2, q_3]$ *

II $[q_0, q_2, q_3, q_4] \xrightarrow{a} [q_0, q_1, q_2, q_4]$ *

II $[q_0, q_2, q_3, q_4] \xrightarrow{b} [q_0, q_3, q_4]$

$[q_0, q_1, q_4] \xrightarrow{a} [q_0, q_1]$ *

$[q_0, q_1, q_4] \xrightarrow{b} [q_0, q_2, q_3, q_4]$ *

$[q_0, q_3, q_4] \xrightarrow{a} [q_0, q_1, q_4]$

$[q_0, q_3, q_4] \xrightarrow{b} [q_0, q_3, q_4]$ *

ترتیب بازکردن کمی است تپولوژی درخت را تغییر دهد

حالت‌هایی که ستاره دارند حالت‌هایی هستند که از q_0 قابل وصول هستند برای هر کدام این‌ها یک حسیر هست.

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

	a	b
$[q_0]$	$[q_0, q_1]$	$[q_0, q_3]$
$[q_0, q_1]$		

حالت‌های ستاره دار

=> این قسمت را از اطلاعات درخت پر می‌کنیم

← برای این مثال خاص چقدر صرفه‌جویی داریم؟ در حالت کورکورانه ← 64 حالت

در این حالت تعداد محاسبات بعد از کلیک هاست ← 18 حالت

هرچه پرانتز بزرگتر نسبت 18 به 64 کوچکتر خواهد بود تا جایی که 99% صرفه‌جویی خواهیم داشت. در این مثال $\frac{2}{3}$ صرفه‌جویی کردیم

Subject:
Year. Month. Date. ( )

در این مثال حالت از 3L حالت قابل وصول هستند رفتن حالتی بدون خواندن حرف برای سه انجام می‌دهیم و در همین حین یک سری هم انجام می‌دهیم.

تمرین ← یک NDFA را بگیرید و DFA را تحویل دهید طبق همین الگوریتم.

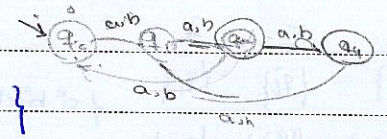
در دوری یک نمایی است که اطلاعات را به ما می‌دهد

	حالت اول	حالت دوم
q_0	0	1
⋮		

حالت نهایی

مثال: $L=\{w \mid |w| \bmod 3 = 0\}$

$L=\{w \mid (n_{(a)} - n_{(b)}) \bmod 3 = 0\}$



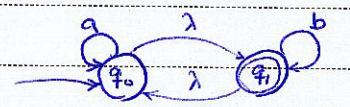
جلسه نهم 7,26

NDFA با حرکت λ

علاوه بر آرک‌های با برچسب‌های الفبای ترانزیت آرک‌هایی داشته باشیم که برچسب آنها λ باشد

آرکی با برچسب λ ← حرکتی مستقل از ورودی یعنی بدون اینکه چیزی بخواند می‌تواند حرکت کند در واقع read head در این حرکت حرکتی به جلو نخواهد داشت

$L=\{a,b\}^*$



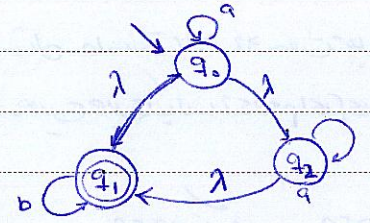
اینجا در q_0 تصمیم می‌گیرد a بخواند در q_0 بماند یا بدون اینکه چیزی بخواند به q_1 برود که این را انتخاب می‌کند که درست حدس بزند

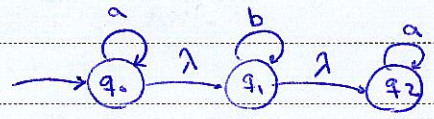
→ aaab ⇝ aaaλb ⇝ بذیرفته می‌شود

یعنی در واقع بین هر دو تا الفبای ترانزیت هر چی λ می‌خواهیم بگذاریم

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )





در q_0 و q_1 می‌توانند خود را به q_2 بفرستند

ولی در q_2 نمی‌توانند به جایی بروند که زبان زیر

رشته‌های همه $\{a^n b^m c^p \mid n, m, p \geq 0\}$

	a	b	c	λ
q_0	$\{q_0\}$	$\{\}$	$\{\}$	$\{q_1\}$
q_1	$\{\}$	$\{q_1\}$	$\{\}$	$\{q_2\}$
q_2	$\{\}$	$\{\}$	$\{q_2\}$	$\{\}$

یک تعریف:

مجموعه تمام حالاتی است که از q_0 قابل حصول است فقط توسط حرکات λ ← λ_closure(q_0)

هر حالتی بدون خواندن چیزی در حالت حلقه‌ای بشد λ (مثل اینجا q_0) داریم

بکمک حرکت λ

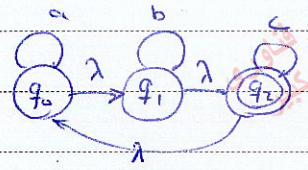
به کمک حرکت λ

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

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

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

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر



ولی در شکل زیر خواهیم داشت:

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

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

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

Month. Date. ( )

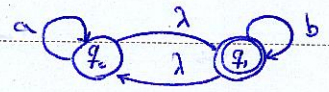
یک مجموعه

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

λ-closure را برای تک تک حالات حساب می‌کنیم و سپس هم می‌بریم

چگونه می‌توان یک NDFA λ را به یک NDFA معادل تبدیل کرد؟

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



δ:

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

output: NDFA $= (Q', \Sigma', \delta', q_0', F')$ accepting L → حالت بدون λ می‌باشد

$Q' = Q$ ⇝ مجموعه حالات هیچ فرقی نمی‌کند

$q_0' = q_0$ ⇝ حالت اولیه

$$F' = \begin{cases} F \cup \{q\} & \text{if } \lambda\text{-closure}(q) \text{ contains a state in } F \\ F & \text{otherwise} \end{cases}$$

$= \{q_0, q_1\}$ ⇝ اگر λ-closure هر یک از حالات جدید یکی از حالات نهایی باشد پس آن حالت را هم به عنوان حالت نهایی در نظر می‌گیریم.

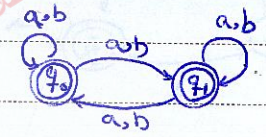
$\delta'(q, a) = \lambda\text{-closure}(\delta(\lambda\text{-closure}(q), a))$

$\delta'(q_0, a) = \lambda\text{-closure}(\underbrace{\delta(\lambda\text{-closure}(q_0), a)}_{\{q_0, q_1\}})$

δ':

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

$q_0 \Rightarrow$



$\lambda\text{-closure}(\delta(\underbrace{\lambda\text{-closure}(q_0)}_{\{q_0, q_1\}}, a))$

$\{q_0\} \cup \{\} = \{q_0\}$

q_0

q_0 ⇝ a

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

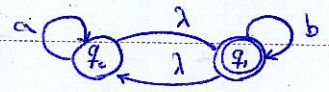
Subject:
Year. Month. Date. ( )

$\delta\text{-closure}(\overset{\text{یک مجموعه}}{P}) = \bigcup_{q \in P} \lambda\text{-closure}(q)$

λ-closure را برای تک تک حالات حساب می‌کنیم و روی هم می‌ریزیم

چگونه می‌توان یک NDFA را بگیریم NDFA معادل تبدیل کرد؟

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



δ:

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

output: NDFA $= (Q', \Sigma', \delta', q_0', F')$ accepting L → همان زبان را می‌پذیرد

$Q' = Q$ ⇝ مجموعه حالات هیچ فرقی نمی‌کند

$q_0' = q_0$ ⇝ حالت اولیه

$$F' = \begin{cases} F \cup \{q\} & \text{if } \lambda\text{-closure}(q) \text{ contains a state in } F \\ F & \text{otherwise} \end{cases}$$

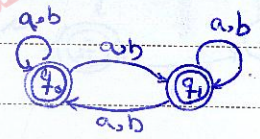
$= \{q_0, q_1\}$ ⇝ اگر λ-closure هر یک از حالات جدید یکی از حالات نهایی باشد پس آن حالت را هم به عنوان حالت نهایی در نظر می‌گیریم.

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

$\delta'(q_0, a) = \lambda\text{-closure}(\underbrace{\delta(\lambda\text{-closure}(q_0), a)}_{\{q_0, q_1\}})$

δ':

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



$\lambda\text{-closure}(\delta(\underbrace{\lambda\text{-closure}(q_0)}_{\{q_0, q_1\}}, a))$

$\{q_0\} \cup \{\} = \{q_0\}$

1

q_0 q_0 ⇝ a

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

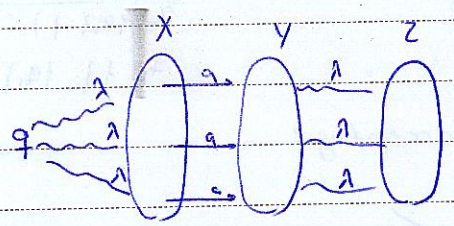
Subject:
Year. Month. Date. ( )

چرا تبدیل F به F' کاری کنند؟

اگر λ-closure شامل یکی از حالت های نهایی باشد می تواند خودش حالت نهایی باشد چون می توانیم بدون خواندن چیزی به حالت نهایی برویم.

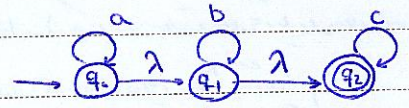
چرا تبدیل δ به δ' کاری کنند؟

برای اینکه ببینیم وقتی در q هستیم بعد از خواندن مثلاً a کجا خواهیم بود اول باید ببینیم که با استفاده از حرکات λ کجا خواهیم بود.



اگر پیشی نخواهد حرکات λ داشته باشد و بخواهد همان رفتار را نشان دهد باید بتواند با خواندن a به جای آخر برود. منظور اینست که اگر در q بودیم و پیشی داشتیم بدون حرکات λ و بخواهیم همان رفتار را داشته باشد باید کاری کنیم که بتوانیم از q با خواندن a به یکی از اعضای Z برویم.

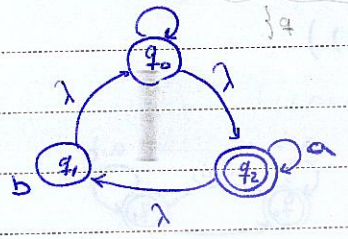
مثال) تبدیل NDFA :



$\Sigma = \{q_0, q_1, q_2\}$

$\delta'(q_0, a) = \lambda\text{-closure}(\delta(\lambda\text{-closure}(q_0), b))$

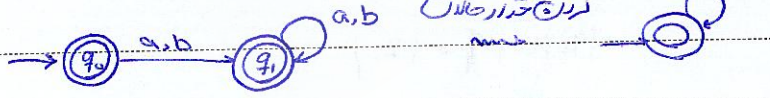
	a	b	c
q_0	$\{q_0, q_1, q_2\}$	$\{\}$	$\{\}$
q_1			
q_2			



دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

حال پیش NDFA بدست آمده در مثال حل شده را می خواهیم به DFA تبدیل کنیم

با اعمال روش minimize کردن تعداد حالات



Subject:
Year. Month. Date. ( )

عدم قطعیت از کجا بیاید؟ علتش چیست؟

* علت عدم قطعیت را بعداً می‌گوییم.

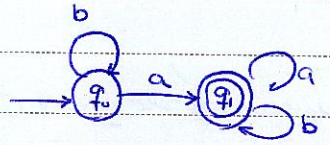
مثال) ماشینی برای این زبان طراحی کنید:

مجموعه تمام رشته‌هایی که هر کدام از آنها دارای حداقل یک a باشد.

حرکت کنیم اولین a را که دیدیم اعلام پذیرش کنیم

برای تبدیل به ماشین ← هنوز a را ندیدیم ← q_0

اولین a را دیدیم ← q_1



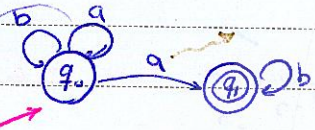
وقتی اولین a را دیدیم بدیهی خود را به انتهای رشته برسانیم

و در نهایت اعلام پذیرش کنیم

حرکت را از راست به چپ باشد و از انتهای رشته دنبال یک a گشتیم ← آخرین a سلک پذیرش می‌شود پس حالتی که باید درست کنیم ← هنوز آخرین a را ندیدیم q_0

آخرین a را دیدیم q_1

a های اولیه و میانی را می‌خوانیم

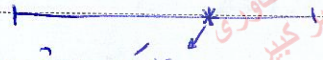


آخرین a باعث می‌شود به حالت اخیری برویم

وقتی به آخرین a رسیدیم پس تا آخر رشته بقیه‌اش b است.

عدم قطعیت به علت طراحی!

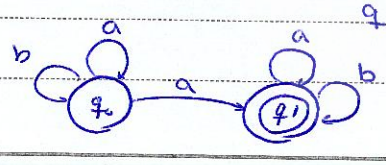
دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر



a ای که به چشم می‌خورد

دو حالت داریم ← هنوز a ای که باعث پذیرش شد را ندیدیم q_0

a ای را که ... دیدیم q_1



Subject:
Year. Month. Date. ( )

point: عدم قطعیت دست چینی خود ماشین خواهد بود و باعث این امر نحوه فکر ماست.

جلسات مهم

← NDFA معادل DFA است ⇐ برای هر ماشین غیرقطعی یک ماشین قطعی وجود دارد.

← $NDFA_\lambda$ " NDFA " ⇐ ماشینی با حرکات λ را می‌توان به ماشینی بدون حرکات λ تبدیل کرد

← $NDFA_\lambda$ " DFA " ⇐ $NDFA_\lambda$ ← NDFA ← DFA

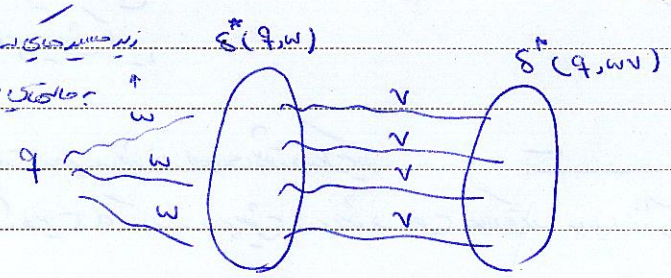
معادل بودن: دو ماشین M_1 و M_2 معادل هستند اگر $L(M_1) = L(M_2)$

↳ زبان‌های یکسانی را بپذیرند.

17. $\delta^*(q, wv) = \bigcup_{p \in \delta^*(q,w)} \delta^*(p, v)$

از حالت q روی ورودی wv به کجا می‌رویم.

زیرمجموعه‌هایی که به جیب‌ها دارند به حالت‌های نهایی می‌رسد



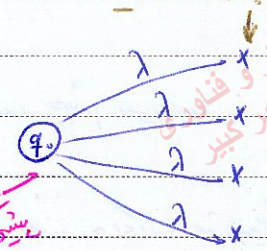
مسئله 9 ص 56

سئوال قبل از جلسه 9 : 2-4 ، 5,6,7

← فرض می‌کنیم ماشین‌ها دارای یک حالت اولیه یک حالت نهایی داشته باشند

تبدیل ماشین با چند حالت اولیه به ماشین با یک حالت اولیه

چهار تا حالت اولیه



ماشینی که دارای یک حالت اولیه است

NDFA → DFA

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

جلسه دهم 7,29

ادامه بحث درباره تئوری زبان های نوع سوم:

فصل سوم: عبارات منظم و گرامرهای منظم:

عبارت منظم (Regular Exp.)

- زاینده زبان های نوع سوم هستند
- دلالت بر زبان های نوع سوم دارند
- جزء نمادهای نوع سوم از پایه ای شوند.

در تعریف:
- عبارات منظم Syntax
- " " Semantic

فرض می کنیم عبارات منظم روی Σ تعریف شده است.

Syntax عبارت منظم:

بدیهی تعریف:
1. $\emptyset$ یک عبارت منظم است.
2. " " " " λ .
3. " " " " $a \in \Sigma$ هر چیزی عضو Σ

4. اگر R_1, R_2 عبارات منظم باشند در این صورت R_1+R_2 ، $R_1 R_2$ ، R_1^* ، (R_1) عبارات منظم هستند

قوت بازگشتی تعریف

مثال:

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

$R = (0+1)^*(10+1)(0+1)^*$ ← ابتدا از مبادی به جزء شنا ی رسیم عبارت بدیهی منظم هستند

$0 \in \Sigma$ ← منظم است

$1 \in \Sigma$ ← " "

$\Rightarrow 0+1 \xRightarrow{\text{منظم}} (0+1) \xRightarrow{\text{منظم}} (0+1)^*$ منظم

$R_1 R_2 \Rightarrow 10 \xRightarrow{\text{عبارت منظم}} 10+1 \xRightarrow{\text{منظم}} (10+1) \xRightarrow{\text{منظم}}$

$(0+1)^* \Rightarrow$ منظم $\Rightarrow$

$R_1 R_2 \Rightarrow \underbrace{(0+1)^*(10+1)}_{R_1} \xRightarrow{\text{منظم}} R_1 R_2 \Rightarrow$ منظم است. $(0+1)^*$

دانشکده مهندسی کامپیوتر و علوم
اطلاعات دانشگاه صنعتی امیر کبیر

Subject:
Year. Month. Date. ( )

Semantic عبارات منظم ← عبارات منظم را چگونه ارزیابی کنیم

پایه تعریف:
1. $\emptyset$ دلالت بر $\{\}$ دارد
2. λ " " $\{\lambda\}$ "
3. $a \in \Sigma$ " " $\{a\}$ "

قسمت بازگشتی تعریف ← 4. فرض کنید R_1 و R_2 به ترتیب دلالت بر $L(R_1)$ و $L(R_2)$ داشته باشند در این صورت

- $L(R_1+R_2) = L(R_1) \cup L(R_2)$
- $L(R_1^*) = L(R_1)^*$
- $L(R_1R_2) = L(R_1)L(R_2)$
- $L((R_1)) = L(R_1)$

مثال) $R = (0+1)^*(00+11)(0+1)^*$

ابتدا با استفاده از موارد 1 و 2 و 3 اجزای متشکله را ارزیابی می‌کنیم

$0 : \{0\}$

$1 : \{1\}$

$0+1 : \{0\} \cup \{1\} = \{0,1\} \Rightarrow (0+1) = \{0,1\}$

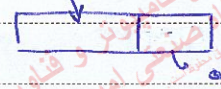
$(0+1)^* : \{0,1\}^* \rightsquigarrow$ اجتماع نیست

$R_1 \quad 00 : \{0\}\{0\} = \{0,0\}$

$R_2 \quad 11 : \{1\}\{1\} = \{1,1\}$

$(R_1+R_2) = (00+11) : \{00,11\}$

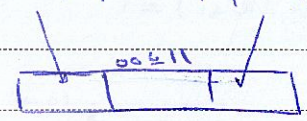
$(0+1)^*(00+11) : \{0,1\}^*\{00,11\} \rightsquigarrow$ هر عضوی از اولی با هر عضوی از دومی concate کنیم



رشته‌ای که متشکل از 00 و است و به 00 یا 11 منتهی می‌شود.

$(0+1)^*(00+11)(0+1)^* = \{0,1\}^*\{00,11\}\{0,1\}^*$

جمیع عناصر رشته‌های درون ها حداقل یک 00 و یک 11 باشد.



دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

اگر یک عبارت منظمی به ما دادند باید بتوانیم تجزیه در قالب کلمات تولید کنیم. یعنی بعد بتوانیم با دیدن نام اعضای آن را بشناسیم.

اگر بخواهیم یک ماشین اتوماتیک به یک رشته یا ترین رشته باشیم موارد زیر را باید رعایت کنیم

+ : یا

تکرار : *

$(0+1)^*$ ⇒ تکرار کن انتخاب های بین 0 را و این انتخاب ها را پشت هم قرار می دهیم

$(0+1)^* = \{0,1\}^*$ هر رشته ای که حاصل می شود در این مجموعه است.

```
while (needed) do
    Select 0 or 1
```

$(0+1)^*(00+11)$ ← یکی از این انتخاب کن ← یکی از این انتخاب کن ← به هم concat کن

```
while (needed) do
    Select 0 or 1
Select.
    Select 0
        " 0
or.
    select 1
        " 1
```

$(0+1)^*(00+11)(0+1)^*$

```
while (needed) do
    Select 0 or 1
    Select 00
    or 11
    while (needed) do
        Select 0 or 1
```

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

این رشته را می‌خواهیم تولید کنیم

00 11010101011

روش مشخصی برای تولید وجود دارد

1. 1 را تکرار می‌کنیم
2. در دومی 00 را انتخاب می‌کنیم
3. در سومی 11 را تکرار می‌کنیم

00 11010101011

1. در بار دوم انتخاب 0
2. انتخاب 11
3. 9 را تکرار می‌کنیم

مثال)

فرض کنید $L_1=\{a,b\}$ ، $L_2=\{aa,ab,ba,bb\}$

$$L_1^* L_2 \overset{?}{=} L_2 L_1^*$$

می‌توان گفت: L_1^* همه رشته

L_2 رشته با طول 2 ⇐ مجموعه تمام رشته‌هایی که دارای حداقل طول 2 هستند

$$L_1 L_2^* \overset{?}{=} L_2^* L_1 \Rightarrow$$

$1+2k$ $2k+1$

صفت مشخصه هر دو زوج بودن طول آنها

طولش زوج است

$$L_1 L_2^* \cup L_1 \overset{?}{\neq} L_2^* L_1 \cup L_2$$

نیستند

طول زوج تضمین در $L_1 L_2^*$ است پس $L_1 L_2^*$ می‌شود

اینجا طول دهم داریم یعنی طول زوج

پس این دو با هم برابر نیستند

دانشکده مهندسی کامپیوتر
اطلاعات دانشگاه صنعتی امیرکبیر

PAPCO

Subject:
Year. Month. Date. ( )

یک مسئله) برای این می‌خواهیم یک عبارت منظم بنویسیم $L = \{a^n b^m \mid n \neq m\}$

اما قبل از این کار یک بری مسایل کوچکتر حل می‌کنیم

$a^* : \{a\}^*$ ⇝ مجموعه تمام رشته‌های متشکل از a با طول صفر یا بیشتر

$aa^* : \{a\}\{a\}^*$ ⇝ مجموعه تمام رشته‌هایی که دارای حداقل طول یک باشند

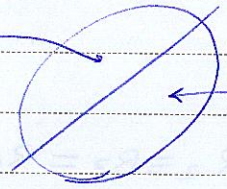
$(aa)^* : \{aa\}^*$ ⇝ متشکل از a و دارای طول زوج

$a(aa)^* : \{a\}\{aa\}^*$ ⇝ فرد " " " " "

حالا به مسئله خودمان برمی‌گردیم.

اگر چیزی در ابتدا به ذهنمان نرسد بهتر مسایل را به مسایل کوچکتر بشکنیم

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



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

$L = L_1 \cup L_2$ ⇐ هرکدام از این زبان‌ها برقرار باشد در نتیجه L هم برقرار است.

← این زبان اصلاً نوع سوم نیست پس نمی‌توان برای آن عبارت منظم بدست آورد.

اگر زبان نوع سوم نباشد نمی‌شود برایش عبارت منظم بدست آورد

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

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

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

$\Rightarrow L = L_1 \cup L_2$

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

مستقلاً برای L_1 و L_2 عبارت منظم می‌نویسیم

$L_2 = \underbrace{a(aa)^*}_{\text{تعداد فرد } a}\ \underbrace{b(bb)^*}_{\text{تعداد فرد } b}$

تعداد فرد a که به دنبالش تعداد فرد b می‌آید

$L_1 = \underbrace{(aa)^*}_{\text{تعداد زوج } a}\underbrace{(bb)^*}_{\text{تعداد زوج } b}$ ⇝ تعداد زوج a که به دنبالش تعداد زوج b می‌آید

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

یا از R_1 آمده یا از R_2

$R_3 = (aa)^* ab(bb^*)$ ⇝ R_2 معادل R_3 است

زیرا هر دو یک زبان واحد را می‌گیرند

$R_4 = (aa)^* a(bb^*)b$ ⇝ باز هم همان R_2 است.

$R_5 = a(aa)^*(bb)^* b$

$R_2 \equiv R_3 \equiv R_4 \equiv R_5$

مسئله مشابه، مسئله 9، صفحه 78

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

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

این زیرزبان‌ها اشتراک غیرمشترک نیستند، یعنی رشته اعضای مشترکی داشته باشند.

$R = \underbrace{aa^* bbbb^*}_{\text{I}} + \underbrace{aaaa^* bb^*}_{\text{II}} + \underbrace{aaa^* bbb^*}_{\text{III}}$

I: حداقل طول 3 — حداقل طول یک

III: حداقل طول 2

select I or II or III

اینجا بخاطر عبارت منظم می‌توان برسر آمد. طبیعتاً آن که خلاصه‌تر است بهتر است.

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

جلسه یازدهم

عبارات منظم

... و دلالت برای عبارات منظم

فرض کنید α و β و γ عبارات منظم باشند در این صورت

خاصیت جابه‌جایی $\alpha + \beta = \beta + \alpha$ .1

پخش پذیری $\alpha(\beta+\gamma) = \alpha\beta + \alpha\gamma$ .2

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

مثال) یک عبارت منظم برای مجموعه تمام رشته‌هایی که دارای طولی که مضربی از 3 باشد بدست آورید

$L = \{w \mid w \in \{a,b\}^*, |w| \bmod 3 = 0\}$

یک از آنها

می‌توانند رشته‌ها تبدیل شوند که هر یک دارای طول سه هستند و هر یک از رشته‌ها عضو مجموعه زیر است

$\{aab, bab, baa, bba, bbb, aba, aaa, abb\}$

پس تقریباً مثل همان تمرین closure است یعنی عبارت منظم بصورت زیر است

$R = (aab + bab + baa + bba + bbb + aba + aaa + abb)^*$

اگر بخواهیم در صورت انتخاب یکی از R را انتخاب کنیم نیاز به انتخابات زیادی خواهیم بود

می‌خواهیم ساده‌سازی کنیم با استفاده از 1 و 2

$R = (aab + bab + baa + bba + bbb + aba + aaa + abb)^*$

با فاکتورگیری $\rightarrow$ $R = (aa(b+a) + ba(a+b) + bb(a+b) + ab(a+b))^*$

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

تمرین: تعداد کلی از این 9 تا که یک a و یا b به دنبالش می‌آید

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

به تعداد تبدیل می‌شود که هر کدام از این تعداد به سمت تبدیل می‌شود به a است یا b

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

تعدادی که به دنبالش تعدادی b می‌آید که یا تعداد زوج و یا فرد است.

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

$= (aa)^*(ab + \lambda)(bb)^*$ $\rightarrow$ تمام رشته‌هایی که تعدادی زوج a شروع شده و به تعدادی زوج b ختم می‌شوند

پس این دو با هیچی می‌آید و اگر بیاید فقط ab است.

دانشکده مهندسی کامپیوتر و فناوری اطلاعات پژوهشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

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

تفاوت بین عدد λ است

$L(\alpha^*) = \{\lambda, \alpha, \alpha\alpha, \alpha\alpha\alpha, \dots\}$

$L(\alpha\alpha^*) = \{\alpha, \alpha\alpha, \alpha\alpha\alpha, \dots\}$

$(\alpha^*)^* = \alpha^*$ .4 ← به قطعه‌ی تقسیم‌ی می‌گوییم که خود قطعه‌ی استیلی می‌شود که همان α است.

← به قطعه‌ی تقسیم‌ی می‌گوییم که هر کدام α است

$\alpha^{*\cdots*} = \alpha^*$

$(\alpha+\beta)^* = (\alpha^*+\beta)^*$ ← هر چی این تولید می‌کند، در یک جم تولید می‌کند چون تکرار زیاد را می‌توانیم ۱ بگیریم

while needed do
Select α or β

while needed do
select
while needed do
select α
or
select β

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

while needed do
select
while needed do
select α
or
while needed do
select β

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

← اصطلاح تولید یک تولید شود

$(\alpha+\beta)^*$ را block α را اینجا قرار دادیم برای اینکه بتوانند توسط

$\alpha^*(\alpha+\beta)^*$ ⇒ while nede do
Select α
while needed do
select
select α
or
Select β

while needed do
while needed do
select α
while needed do
select β

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

$\alpha\alpha\beta\alpha\beta\beta\alpha\alpha\beta \quad \alpha\beta \quad \alpha$

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

$$(\alpha+\beta)^* = \underbrace{((\alpha+\beta)^*)^*}_{(\alpha+\beta)^*}\alpha^*\underbrace{(\alpha^*\beta^*)^*}_{(\alpha+\beta)^*}\alpha^* \checkmark$$

$$\alpha^*\alpha^* = \alpha^* \Leftarrow (\alpha+\beta)^* \quad (\alpha+\beta)^* = (\alpha+\beta)^*$$

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

$\beta\alpha\beta\alpha\beta\alpha\beta\alpha\beta$

$\beta(\alpha\beta+\alpha)^* \neq (\beta\alpha+\alpha)^*\beta$ ✓ ← رشته‌هایی که خروجشان α و یا β است و موضع β نیست

همه رشته‌هایی که تولید می‌کنند خروجشان β است

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

$\alpha\beta\alpha\beta\alpha\beta\alpha\ \alpha\alpha\alpha$

مثال) چند گرامر خطی از سمت چپ معادل پیدا کنید.
چند گرامر دیگر خطی از سمت راست پیدا کنید.

$S \to abA$
$A \to baB$
$B \to aA \mid bb$

} خطی از سمت راست

abbaabbaabbaabbaabbaabbabb

تعداد تکرار شونده (aba)

$S \to abbA$
$A \to aaB \mid abb$
$B \to bA$

$S \to abbaA$
$A \to abB \mid bb$
$B \to aA$

عبارت منظم: $abb(aab)^*abb = ab(baa)^*bb$

حالا گرامر خطی از سمت چپ برایش می‌آوریم.

گرامر خطی سمت راست رشته → تولید می‌شود

← ... چپ قسمت‌های انتهایی تولید می‌شود

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

$S \rightarrow Abb$

$A \rightarrow Bba \mid abba$

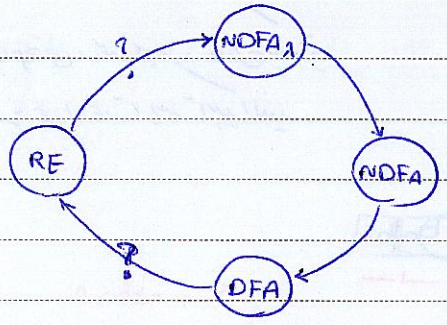
$B \rightarrow Aa$

برای حذفی از سمت چپ:

باید توجه داشت که در رشته اول قسمت های انتهایی تشکیل می شود.

تئوری زبان های نوع سوم: اگر نوع سوم نباشد هیچ کدام از موارد زیر را نمی توان برایش تعیین کرد.

- پذیرنده:
  - DFA
  - NDFA ← بدون حرکات λ
  - NDFA ← با حرکات λ
- عبارات منظم



→ این شکل می گوید اگر یکی از این ها را داشته باشیم بدست آوردن بقیه را بدست آورده ایم

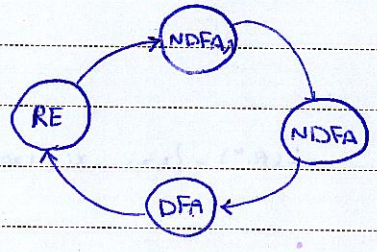
مثال 3.5 - 3.4 3.3

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیر کبیر

Subject:
Year. Month. Date. ( )

جلسه دوازدهم:

۱) اثبات معادل تقریبی زبان‌های نوع سوم (منظم)



RE $\longrightarrow$ NDFA λ

رویه تبدیل RE به NDFA$_\lambda$

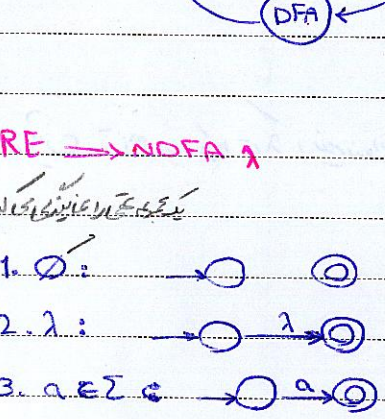
یک عبارت را عایق می‌کند

1. $\emptyset$:

2. λ :

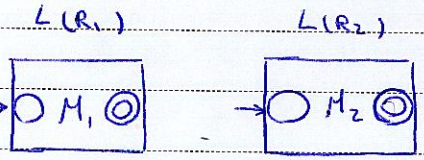
3. $a \in \Sigma$:

4. 1

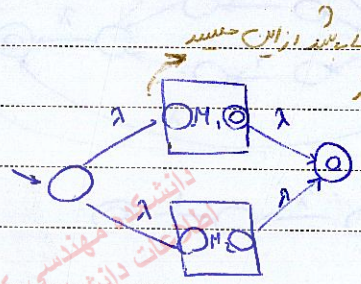


هیچ مسیری از حالت اولیه به حالت نهایی ندارد

ماشینی که تنها یک عضو دارد و آن هم λ است.



$L(R_1) \cup L(R_2) \Rightarrow$



اگر رشته‌ای $L(R_1)$ باشد از این مسیر اعلام پذیرش می‌کند

4.2

از ترکیب این دو ماشین می‌سازیم که $L(R_1)L(R_2)$ را بپذیرد

$L(R_1)L(R_2) = \{xy \mid x \in L(R_1), y \in L(R_2)\}$

در انتهای x باشیم اینجا هست

در انتهای y اینجا می‌رسد

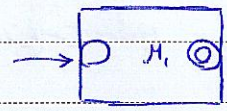
دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:

Year. Month. Date. ( )

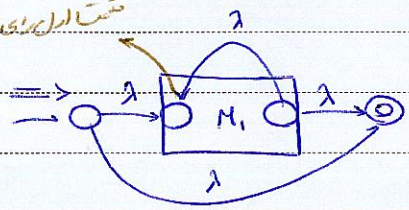
9.3 $L(R_1)$

می‌توانیم ماشینی بسازیم که $L(R_1^*)$ را بپذیرد



قسمت اول برای خواندن بی‌شمار دفعه و قسمت دوم را پذیرش می‌کند

$L(R_1^*)$ ⟹

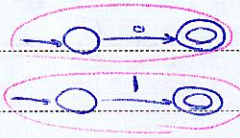


$L(R_1^*) = \{x_1 x_2 \ldots x_i \mid i \geq 0,\ x_i \in L(R_1)\}$

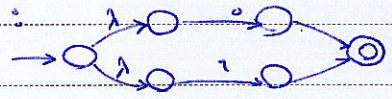
مثال: ماشین متناهی با حرکات λ که پذیرنده این عبارت را بسازید:

$R = (0+1)^*(11+0)(0+1)^*$

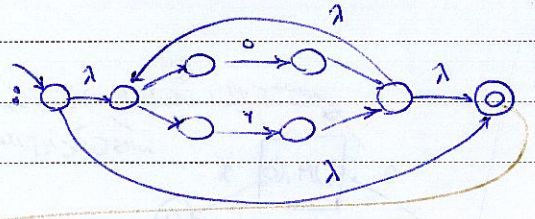
0 :



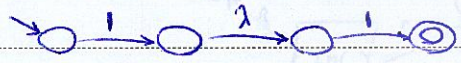
(0+1) :



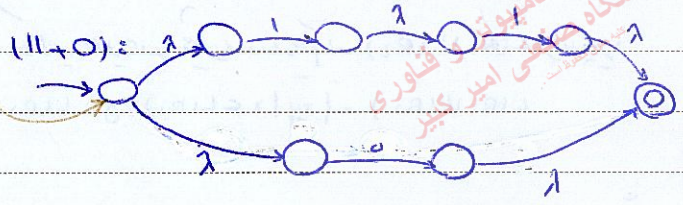
(0+1)* :



11 :



(11+0) :



دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

باید بدست آوریم NDFA و می‌توانیم با روش‌های قبلی NDFA را مینیمم کنیم.

FA ⟹ RE

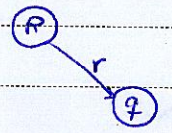
ساخت FA از روی RE

Generalized Transition Graph گراف متعمم یافته

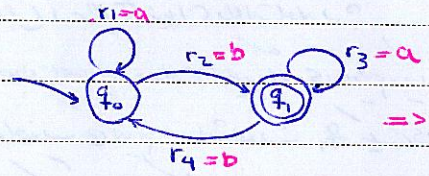
← در node یک arc هم وصل می‌شوند که هر arc یک label داشت (در Transition Graph)

← در این نوع گراف برچسب انتقالات عبارات منظم هستند

اگر ماشین در q بود و رشته‌ای آمد که r بر آن دلالت دارد به حالت q خواهد رفت (در حقیقت یک مجموعه وابسته می‌کنیم به arc)



مثال) یک عبارت منظمی پیدا کنید که برای کلمه این ماشین می‌پذیرد دلالت داشته باشد.



$\Rightarrow R = a^*b(ba^*b+a)^*$

$R = ?$ ⟹ مجموعه برچسب‌های عبارات منظم را به ما خواهد داد

مسیرهایی که در این گراف ما را از q_0 به q_1 می‌رساند

چندین بار در q_0 loop می‌زند

$q_0 \xrightarrow{r_1^*} q_0 \xrightarrow{r_2} q_1 \rightsquigarrow q_1 \rightsquigarrow q_1 \rightsquigarrow q$

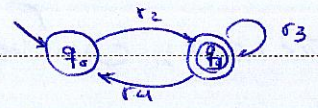
برای خروج و ورود به q_0 دو راه داریم ← $r_4 r_1^* r_2$ ، r_3^*

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

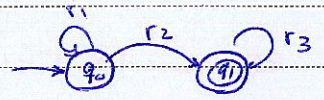
سایه دارد q_1 می‌کند

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

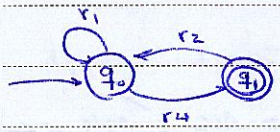
Subject:
Year. Month. Date. ( )



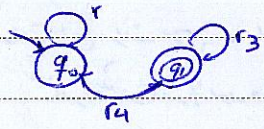
$R = r_2 (r_4 r_2 + r_3)^*$



$\Rightarrow R = r_1^* r_2 r_3^*$

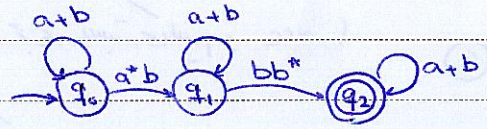


$R = r_1^* r_2 (r_4 r_1^* r_2)^*$



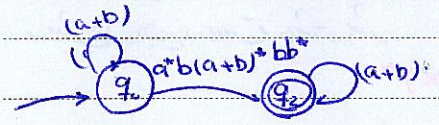
$\Rightarrow R = \emptyset$

مثال)



عبارت منظمی که بر این زبان دلالت دارد؟

- حالت نهایی و حالت اولیه را نگه می‌داریم و دیگر حالت را حذف می‌کنیم در اثر حذف q_1 یک مسیر جدید از بین می‌رود که آن را به می‌گیریم

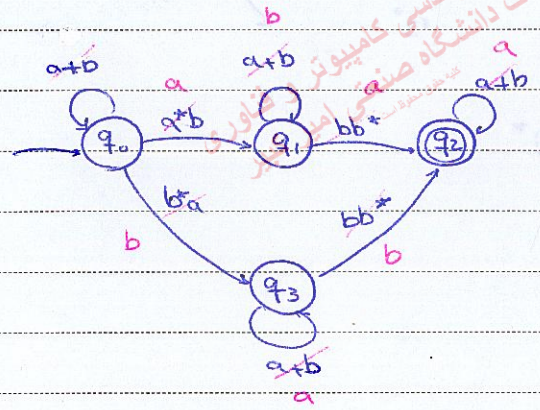


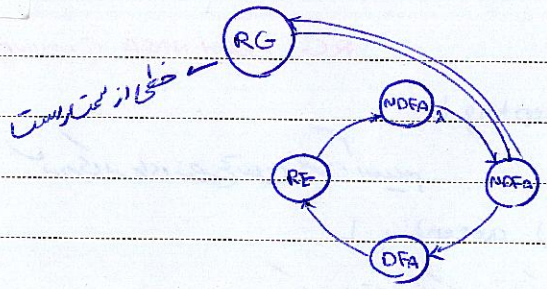
مسیرهایی که مستقیماً از q_0 به q_2 می‌رسند از q_1 می‌گذرند

$R = (a+b)^* \, a^*b(a+b)^* \, bb^* \, (a+b)^*$

اگر 4 تا حالت داشت node های میانی را یک به یک حذف می‌کنیم و به جایش جدید می‌گذاریم و سه تا node می‌ماند و بعد یک node میانی حذف می‌کنیم.

دانشکده مهندسی کامپیوتر
انجمن علمی دانشجویی
دانشگاه صنعتی امیرکبیر



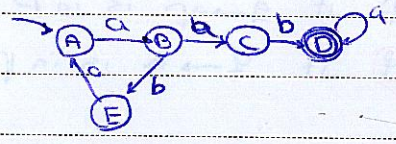


خطی از لحاظ است

اگر شکل Regular Grammer را اضافه کنیم

NDFA => RG (NDFA از روی RG پیدا می‌شود) RG ← NDFA تبدیل

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



Find : RG = (V, T, P, S) accepting L

برای این مثال

$V = Q \Rightarrow Q = \{A, B, C, D, E\}$

$T = \Sigma = \{a, b\}$ $\quad p = \delta(p, a) = q =$

علامت‌هایی که رشته برای ساختن

Starting symbol $= q_0 \Rightarrow S = q_0$ برای این مثال $S = q_0 = A$

مجموعه قواعد → $P = ?$

مجموعه قواعد از روی δ یا transition پیدا می‌شود.

- Add $p \to aq$ to P if $\delta(p, a) = q$ که $q \in F$
- " $p \to aq$ and $p \to a$ to P if $\delta(p, a) = q$ که $q \in F$

$P = \{A \to aB,$
$B \to bE,$
$E \to aA,$
$B \to aC,$
$C \to bD, C \to b,$
$D \to aD, D \to a\}$

هر رشته‌ای که توسط این ماشین پذیرفته شود در این گرامر هم کار می‌کند.

هر متغیر چیزی دارد که یک رشته است و این رشته در این گرامر یک اشتقاق دارد.

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

RG ⟹ NDFA — روشی برای ساخت NDFA از روی RG

Given : RG = (V, T, P, S) accepting L

شرایطی که برای موضوع قبل بدست آوردیم

Find : NDFA = (Q, Σ, δ, q₀, F) accepting L

Q = V ∪ {f} — یک حالت نهایی که باید جدید خودمان اضافه کنیم

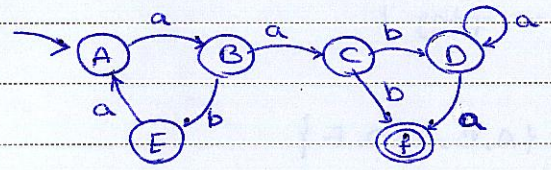
Q = {A, B, C, D, E, f}

q₀ = S = A

δ : — از روی قواعد گرامر دو روی پیدا خواهیم کرد

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

همان حالت نهایی است



با کمی دقت می‌توان دید که می‌توانیم f را حذف کنیم و D را حالت نهایی قرار داد.

اگر در C بودیم و b ببینیم این آخرین b باشد به حالت نهایی می‌رویم ← D

RE ⟹ RG — ترکیب‌های کلیدی به production تبدیل می‌شوند

R = (a+b) — آخرین نمادی که می‌خوانیم یا a یا b است.

S → a | b

R = aa*b^A

S → aA

A → aA | b

A منتهی می‌شود به b

R = aa*(b+a)

S → aA | a ?

A → aA | b | a

aA

aA

A → aA | b | a

a

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

قضیه:

Given : $RG = (N, T, P, S)$ accepting L

Find : $RG' = (N', T, P', S')$ " L^R زبان معکوس

Given: $S \to aA$, $A \to bB$

$B \to aB$, $B \to b$

Find: $S \to Aa$, $A \to Bb$

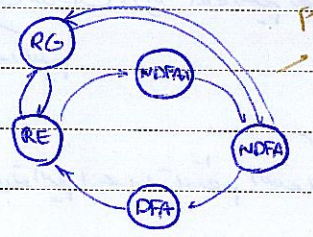
$B \to Ba$, $B \to b$

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

جلسه سیزدهم: 13, 8,

ادامه بحث در مورد تئوری زبان های نوع سوم:

فصل چهارم



فصل چهارم:

خواصیات زبان های منظم (نوع سوم)

closure properties ← خواصیات بستاری زبان های منظم

- مسائل قابل تصمیم گیری برای زبان های منظم

- قضیه pumping برای زبان های منظم

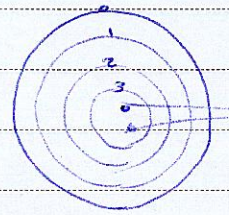
دانشکده مهندسی کامپیوتر
اطلاعات و فناوری
دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

خصوصیات بستاری زبان های منظم (closure properties)

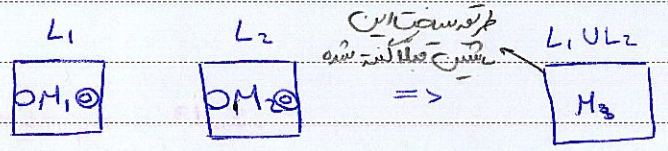
قضیه ۱: خانواده زبان های منظم تحت عملگرهای $\cup$ ، concatenation ، * بسته است

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



concate اگر دو زبان از این خانواده باشند اجتماع، closure اشتراک هم در این خانواده است

اثبات قضیه: L_1 و L_2 زبان های نوع سوم هستند : فرض ها



ماشینی می سازیم که این زبان را می پذیرد ⇐ اگر نوع سوم نبودن می توانستیم ماشین بسازیم

Constructive proof → از شواهد و ارائه برای اثبات استفاده می کنیم بلکه از ساخت برای اثبات استفاده می کنیم

برای اثبات الفا می توانیم از عبارات منظم استفاده کنیم ← با استفاده از تجاریا عبارات منظم

برای موارد دیگر در این قضیه از همین روش استفاده می کنیم

قضیه ۲ ← اگر L نوع سوم باشد در این صورت $\bar{L}$ نوع سوم است
خانواده زبان های نوع سوم تحت متمم بسته است

اثبات: از طریق constructive proof یعنی M_1 که L را می پذیرد داریم ماشینی می سازیم که $\bar{L}$ را بپذیرد

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

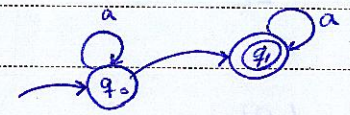
Find: $\overline{FA} = (\bar{Q}, \bar{\Sigma}, \bar{\delta}, \bar{q}_0, \bar{F})$ " $\bar{L}$

دانشکده مهندسی کامپیوتر
دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

- به شرح و ساخت ماشین را می‌گوییم

مثال: ماشین FA که L را می‌پذیرد



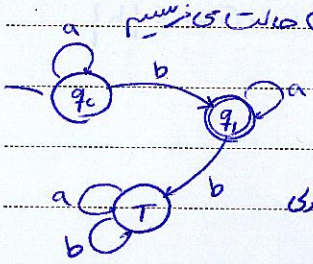
$\bar{Q} = Q$

$\bar{\delta} = \delta$

$\bar{q}_0 = q$

$\bar{F} = Q - F$

یک حالت جدید به نام Trap ایجاد می‌کنیم و در جاهایی که Transition وجود ندارد به این حالت می‌فرستیم



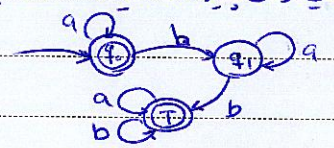
اگر وارد شویم دیگر نمی‌توانیم خارج شویم

همیشه ماشین جدید خودش را به انتهای رشته می‌رساند و جداً اعلام پذیرش یا عدم پذیرش می‌کند ولی

در ماشین قبلی ممکن است در وسط رشته اعلام عدم پذیرش کند

هر جا که ماشین قبلی می‌پذیرفت این نمی‌پذیرفت ← چون در ماشین قبلی q_1 حالت نهایی بود و هر چه در نهایت به q_1

برسد پذیرفته می‌شد ولی در این جا T و q_0 حالت نهایی است و زبان‌هایی را می‌پذیرد که در این حالت باشند



ماشینی که متمم را می‌پذیرد

قضیه 3 ← خانواده زبان‌های نوع سوم تحت اشتراک بسته است. اشتراک دو زبان نوع سوم، نوع سوم است

دو تا اثبات دارد ← De Morgan

constructive proof

اثبات: ① $\overline{L_1 \cap L_2} = \overline{L_1} \cup \overline{L_2}$

زبان‌های نوع سوم تحت متمم و اجتماع بسته است

تحت راست نوع سوم است پس طرف چپ هم نوع سوم است

می‌توان طی 9 مرحله با استفاده از درس‌های قبلی ماشینی ساخت که اشتراک را بپذیرد

دانشکده مهندسی کامپیوتر
اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

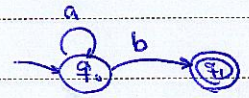
اثبات ج :

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

$FA_2 : (Q_2, \Sigma_2, \delta_2, q_0^2, F_2)$ " L_2

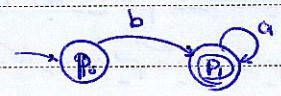
Find : $FA : (Q, \Sigma, \delta, q_0, F)$ accepting $L_1 \cap L_2$

مثال)

$FA_1 \rightsquigarrow$ 

$: \{a^n b \mid n \geq 0\}$

$\Rightarrow L_1 \cap L_2 = \{b\}$

$FA_2 \rightsquigarrow$ 

$: \{b a^n \mid n \geq 0\}$

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

$q_0 = [q_0, P_0]$

↳ این حالت حالتی که عضو اول عضو شروع کننده ماشین اول و عضو دوم عضو شروع کننده ماشین دوم باشد

$F = \{ [r, s] \mid r \in F_1 , s \in F_2 \} = \{ [q_1, P_1] \}$

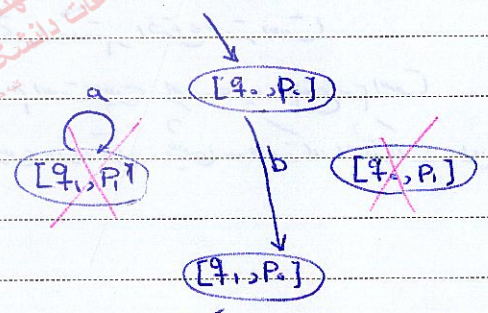
حالت‌های FA_1 ↲ ↳ FA_2 حالت‌های

برای تعریف شدن Transition هر دوی اینها باید تعریف شده باشند

$\delta([r,s], a) = [r', s']$ iff $\delta_1(r, a) = r'$, $\delta_2(s, a) = s'$

ماشین اول ماشین دوم

	a	b
$[q_0, P_0]$	—	$[q_1, P_1]$
$[q_0, P_1]$	$[q_0, P_1]$	—
$[q_1, P_0]$	—	—
$[q_1, P_1]$	—	—

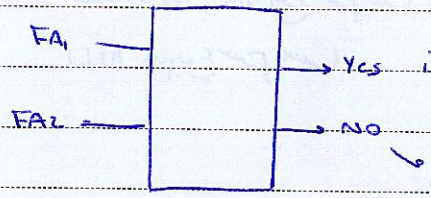


این دو را نمی‌توانیم حذف کنیم از حالت اولیه حالت‌هایی که هیچ مسیری وجود ندارند

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

مثال) ماشینی طراحی کنید:



در غیر این صورت

دوراه ← رشته را تک تک باید چک کنیم اگر هر دو پذیرفتند

رشته را همزمان به آنها بدهیم و رفتار ماشین ها را در خواندن رشته کنترل کنیم یعنی علامت علامت

راجع اینکه اگر در انتهای رشته هر دو در حالت های باشند پذیرش رشته می کنیم

تمرین) برای همین مثال با استفاده از قانون دمورگان ماشین را درست بیاورید

Difference

تفسیر ۹ ← خانواده زبان های نوع سوم تحت عملگر تفریق بسته است. اگر L_1 و L_2 زبان های نوع سوم باشند

در این صورت $L_1 - L_2$ هم نوع سوم است

اصلاً تفریق هم باید بسته باشد

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

طی دو مرحله ماشینی بسازیم که ← رای پذیرد ← اول L_2

دوم اشتراک

Homomorphism : هومومورفیزم

$h: \Sigma_1 \longrightarrow \Sigma_2^*$

(mapping) یک نگاشت

سوالات Σ_1, Σ_2

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

یک نگاشت

$h(a) = 00$ or $h(a) = 0$

$h(b) = 10$ or $h(b) = 1$

این نگاشت

$L = \{abb, ba, bb\}$

$h(aab) = h(a) \cdot h(a) \cdot h(b) = 000010$

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

$= \{001010, 1000, 1010\}$

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

5 ← خانواده زبان های نوع سوم تحت همومورفیزم بسته است. اگر L نوع سوم باشد در این صورت h(L) هم نوع سوم است.

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

$h(L_1) = \{(aa)^n (ba)^m \mid n,m \geq 0\}$

اثبات: روش های مختلفی برای اثبات وجود دارد ← از گرامر یا ماشین و عبارات منظم می توانیم این کار را انجام دهیم

اثبات از طریق گرامر:

L_1: $S \to aA$
$A \to bbA$
$A \to ab$

$h(L_1)$: $S \to h(a)A$
$A \to h(b)h(b)A$
$A \to h(a)h(b)$

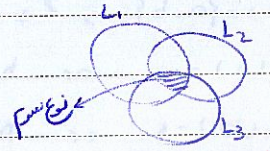
رشته های در L_1 ← تعدادی از یک علامت که بیشتر از تعدادی دیگر علامت دوم بیشتر از آن می آید

!! هم نیست علامت چه باشد یک رشته با علامت.

همه خصوصیاتی که در زبان های نوع سوم است توسط گرامر نوع سوم بیان شود. نمی توان

مسئله 5: خانواده ی زبان های نوع سوم تحت اجتماع محدود و اشتراک محدود بسته است.

اگر $L_1, L_2, \dots, L_n$ نوع سوم باشند در این صورت $\bigcup_{i=1}^{n} L_i$ و $\bigcap_{i=1}^{n} L_i$ نوع سوم هستند



اگر تعداد n محدود باشد هیچ مشکلی نمی توانیم اثبات کنیم چون با همین رویه ای که برای زبان n محدود مراحل n محدود این کار را انجام دهیم

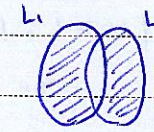
روش اثبات اینست که دو تا دو تا با هم بگیریم و نتیجه را با بقیه بگیریم.

دانشکده مهندسی کامپیوتر و فناوری
اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

مسئله 6) Symetric Difference

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



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

از تحت تمام این عملگرها بسته است پس تحت $\ominus$ هم بسته است

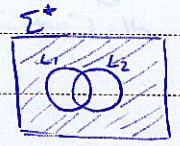
خط مراحل پیشنی بسازید که $\ominus$ را بدهد.

مسئله 7) L_1 و L_2 دو زبان Σ تعریف شده

$nor(L_1, L_2) = \{ x \mid x \in L_1 \text{ and } x \in L_2 \}$

$\Sigma^* - (L_1 \cup L_2)$

خروجی نوع سوم است



جلسه چهاردهم 8,18,

خصوصیات زبان های نوع سوم:

- خصوصیات بسته ای

- خانواده زبان های نوع سوم تحت $\cup, \cap, *, \cdot$ بسته است

مسئله 8 ص 112. کدام یک از روابط زیر صحیح است؟

$h(L_1 \cup L_2) = h(L_1) \cup h(L_2)$

$h(L_1 \cap L_2) = h(L_1) \cap h(L_2)$

$h(h_1 \cdot h_2) = h(L_1) \cdot h(L_2)$

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

12 ، 112 ← اگر L یک زبان منظم است نشان دهید که منظم است

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

$L_1 = \{UV \mid U \in L ,\ |V| = 2\}$

نوع سوم

$L_1 = L \cdot \{aa, ab, ba, bb\}$

از L باقی آید

یکی از این 4 تا است

هر زبان متناهی نوع سوم است ← چون اینکه یک عبارت منظم برایش بنویسیم

concatenation در زبان نوع سوم ← نتیجه نوع سوم خواهد بود.

← خانواده زبان های نوع سوم تحت عملگر معکوس بسته است. اگر L نوع سوم باشد در این صورت L^R هم نوع سوم است

اثبات با استفاده از ماشین های متناهی:

ماشینی داریم که L را می پذیرد ماشینی بسازیم که L^R را بپذیرد ← جای حالت اولیه و نهایی را عوض می کنیم و جهت فلش ها را برعکس می کنیم

مسئله 13 ـ 112) اگر L یک زبان منظم است نشان دهید که $L_1 = \{UV \mid U \in L ,\ V \in L^R\}$ هم منظم است (نوع سوم است)

$L_1 = L \cdot L^R$

L نوع سوم ← طبق قضیه قبل نوع سوم است ← در نتیجه L_1 هم نوع سوم است.

مسئله 16 ـ 113) $head(L) = \{x \mid xy \in L \text{ for some } y \in \Sigma^*\}$

$L = \{abb, ba, bb\}$

← تمام رشته هایی که اگر رشته ای مانند $y \in \Sigma^*$ را به دنبال آن بگذاریم

یعنی از L می شود

تمام prefix ها

$(head)\ L = \{\lambda, \lambda, \lambda\}$ ⇒

L: a, b, b

تمام prefix های موجود در L: ab, ba, bb

abb

⇒ $L \subset head(L)$

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

اگر L نوع سوم باشد ← head(L) هم نوع سوم است

← ماشینی داریم که L را می‌پذیرد ← تمام حالت‌ها را به حالت‌های تبدیل کنیم چون head(L) تمام زیرمجموعه‌ها را می‌پذیرد

مسائل قابل تصمیم‌گیری برای زبان‌های نوع سوم:

← مسأله عضویت برای زبان‌های نوع سوم قابل تصمیم‌گیری است

deciedable problems

عضویت

یک رشته و یک زبان بدهند ← ببینیم که رشته عضو زبان است یا نه

decidable ← الگوریتم و روشی وجود دارد که بتوانیم بفهمیم در این جا عضویت را درک هک کند

در این جا ماشین تصمیم‌گیری همان الگوریتم عضویت است

← این مسئله که یک زبان نوع سوم تهی می‌باشد یا نه قابل تصمیم‌گیری است

← " " "

Subject:
Year. Month. Date. ( )

مسأله ناتمامی بودن یک زبان نوع سوم

تعداد رشته‌هایی که می‌پذیرد متناهی است یا نامتناهی؟

مسأله تساوی دو زبان نوع سوم

هر دو متناهی باشند بررسی کنیم آیا این شرایط برقرار است یا نه

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

$L_1 \subset L_2$ ، $L_2 \subseteq L_1$

$x \in L_1 \Rightarrow x \in L_2$ ، $x \in L_2 \Rightarrow x \in L_1$

← در زبان متناهی این کار انجام می‌شود

اگر نامتناهی باشند این روش قابل انجام نیست ← ممکن است چک کردن یک عضو بی‌نهایت طول بکشد چون نامتناهی است و بی‌نهایت عضو دارد ← فقط نشان می‌دهیم که قابل تصمیم‌گیری است

الگوریتمی برای تصمیم‌گیری ← L_3 را می‌سازد

$L_3 = (L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2)$

از روی ماشین‌های L_1 و L_2 ← ماشین L_3 را می‌سازد

در ۵ مرحله این ماشین را می‌سازیم

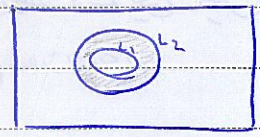
الگوریتمی برای چک کردن این داریم

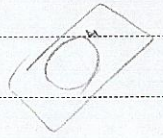
if $L_3 = \emptyset$ then $L_1 = L_2$

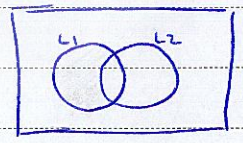
else $L_1 \neq L_2$

این الگوریتم در زمان متناهی انجام می‌شود و ساخت ماشین در مدت زمان متناهی پایان می‌پذیرد ← در زمان متناهی می‌دانیم پاسخ برای این سوال باشیم

از طریق ماشین روبرو L_3 را می‌سازیم







دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

پس تجزیه پذیری یک زبان نوع سوم برای زبان نوع سوم کلاسیک

روش کلاسیک امکان پذیر نیست

$L_1 \overset{?}{\subset} L_2 \Rightarrow L_3 = L_1 \cap L_2$

if $L_3 = L_1 \Rightarrow L_1 \subset L_2$

if $L_3 = L_2 \Rightarrow L_2 \subset L_1$

ساخت زبان در زبان متناهی امکان پذیر است ← جواب به این سوال در زبان متناهی قابل بررسی است.

این مسئله که آیا عضوی از یک زبان نوع سوم می باشد یا نه قابل تصمیم گیری است.

فرض می کنیم DFA داده شده باشد چون اگر NDFA باشد باید ابتدا به DFA تبدیل کنیم سپس تصمیم گیری کنیم. جای حالت اولیه دلخواهی را عوض می کنیم

← یک زبان پالیندروم palindrome است اگر $L = L^R$

اگر تک تک رشته های یک زبان پالیندروم باشد ← زبان پالیندروم است.

← برای این کار ماشین مکعبی را می سازیم

لم تزریقی (تطبیقی) Pumping Lemma برای زبان های نوع سوم

فرض کنید L یک زبان متناهی نوع سوم باشد (منظم) در این صورت یک عدد مثبت صحیح m وجود دارد

به طوری که برای هر $w \in L$ و $|w| \geq m$ باشد آنرا می توان به صورت xyz تجزیه کرد

به قسمت شکنند

شرایط تجزیه: $|y| \geq 1$ ، $|xy| \leq m$ به طوری که $xy^iz \in L$ برای تمام $i = 0, 1, 2, \ldots$

مقدار $|y|$ را حداکثر می خواهیم اگر کوچک کنیم به رجم عضوی از L باشد

$i = 0 \Rightarrow xy$

$i = 1 \Rightarrow xyz$

$i = 2 \Rightarrow xyyz$

دانشکده مهندسی کامپیوتر
اطلاعات دانشگاه صنعتی امیرکبیر

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

بررسی بودن یک زبان نوع سوم برای زبان نوع سوم دیگر

$L_1 \overset{?}{\subset} L_2 \Rightarrow L_3 = L_1 \cap L_2$ — بررسی کلاسیک امکان پذیر نیست

if $L_3 = L_1 \Rightarrow L_1 \subset L_2$

if $L_3 = L_2 \Rightarrow L_2 \subset L_1$

ساخت زبان در زمان متناهی امکان پذیر است ← جواب به این سوال در زمان متناهی قابل پیگیری است.

این مسئله که λ عضوی از یک زبان نوع سوم می باشد یا نه قابل تصمیم گیری است.

فرض می کنیم DFA داده شده باشد چون اگر NDFA باشد باید ابتدا به DFA تبدیل کنیم و سپس تصمیم گیری کنیم. جای حالت اولیه و نهایی را عوض می کنیم

← یک زبان پالیندروم است اگر $L = L^R$ palindrome

اگر تک تک رشته های یک زبان پالیندروم باشد ← زبان پالیندروم است.

← برای این کار ماشین تورینگ را می سازیم

لم تزریقی (تلمبه ای) Pumping Lemma برای زبان های نوع سوم

فرض کنید L یک زبان متناهی نوع سوم باشد (منظم) در این صورت یک عدد مثبت صحیح m وجود دارد به طوری که برای هر $w \in L$ و $|w| \geq m$ باشد، w را می توان به صورت xyz تجزیه کرد

به قسمت شکاف

۱) شرایط تجزیه: $|y| \geq 1$ ، $|xy| \leq m$ به طوری که $xy^iz \in L$ برای تمام i ها $i = 0, 1, 2, \ldots$

مشاهده شد را مشاهده می کنیم که اگر کلمه ای از L باشد

$i = 0 \Rightarrow xy$

$i = 1 \Rightarrow xyz$

$i = 2 \Rightarrow xyyz$

دانشکده مهندسی کامپیوتر
اطلاعات دانشگاه صنعتی امیرکبیر

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

جلسه پانزدهم 87/8/20

قضیه pumping برای زبان های نوع سوم:

- فرض کنید L یک زبان منظم نامتناهی باشد در این صورت یک عدد صحیح مثبت m وجود دارد به طوری که برای هر $w \in L$ ، $|w| \geq m$ آن را به صورت xyz تجزیه کرد (شروط تجزیه: $|y| \geq 1$ ، $|xy| \leq m$)

به طوری که $xy^iz \in L$ برای هر $i = 0, 1, \ldots$

کاربرد:

- استفاده می شود برای اینکه نشان دهیم یک زبان نوع سوم نیست ← رشته ای پیدا کنیم که به هیچ وجه دارای این خاصیت نیست (برای آن شکستگی وجود ندارد)

- از این قضیه برای نشان دادن اینکه زبان نوع سوم است نمی توان استفاده کرد چون باید برای تمام رشته ها تمام آنها این خاصیت را نشان دهیم که در این صورت بی نهایت زمان نیاز داریم.

مثال) این زبان نوع سوم نیست $L = \{a^n b^n \mid n \geq 0\}$ ⟹

نمی شود مثل گرامر نوع عبارت منظم برای این می توانیم استفاده کنیم

اثبات اینکه نیست

من ← تیپ ← اول من جلو بیاری کنند

$a^m b^m$ ② m ①

اولین حرکت

من در اولین حرکت یک رشته مناسب انتخاب کنند

$\underbrace{aa\ldots a}_{m}\underbrace{bb\ldots b}_{m}$ ③

در بعد رقیب این رشته را می شکاند و به من می دهد

xyz ④

$xy^iz \notin L$ ⑤

⑥ من برنده می شود.

در مثال حرکتی که رقیب کامل است انجام دهد برنده داشته باشیم

برای تمام شکاندن های رقیب باید یک برنده داشته باشیم.

در مرحله ④ باید تابعی پیدا کنیم که بتوانیم حرکات رقیب را درک میان کنیم

جلو می آوریم $|xy| \leq m$ ← در مثال اول (a^n) قرار می گیرند ←

$\overset{x}{a^k}\ \overset{y}{a^l}\ \overset{z}{a^k b^m}$

دانشکده مهندسی کامپیوتر خاوری
اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

برای مقادیر مختلف k و k' می‌توانیم نقش‌های رقیب را ببینیم اگر نشان دهیم برای هر مقدار از k، iای وجود دارد که برنده شدیم

$$a^k (a^k)^i a^{k'} b^m \notin L$$

$\Rightarrow$ $i=0$ ... اگر عضو L نخواهد بود و تعداد aها بالا ... می‌ماند

مثال دیگر

$L = \{ ww^R \mid w \in \{a,b\}^* \}$ $\Rightarrow$ پالیندروم‌های با طول زوج

$(ww^R)^R = w^R w^R = ww^R$

من

$a^m b^m b^m a^m$

رقیب

m

$\underbrace{aa\ldots a}_{m}\underbrace{bb\ldots b}_{m}\underbrace{bb\ldots b}_{m}\underbrace{aa\ldots a}_{m}$

$|xy| \le m$ لا در این منطقه‌ی اقتدر جدل

$x \quad y \quad z$

$a^{k'} \quad a^k \quad a^{k'} b^m b^m a^m$

← اتحاد رشته توسط من تعیین کننده است ← اثبات را کلن دیا بر بحث کند (تعداد حرکات رقیب را محدود کند)
هر چه تعداد حرکات رقیب کمتر باشد باید در مقابل حرکات بیشتری داشته باشیم و انرژی بیشتری باید مصرف کنیم

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

اگر i را صفر بگیریم همیشه برنده‌ام چون تعداد aها در قسمت اول تغییری نکرده برای اینکه دوباره عضو L باشد باید از کمت راست هم تعداد aها بیفتد ولی با bها دستکاری نداریم

انشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

مثال سوم) $L = \{a^{n!} \mid n \geq 0\}$ ⟶ زبان نوع سوم نیست

رشته	رشته
$a^{m!}$ ← با طول $\geq m$	m
↳	$\underbrace{aa\ldots a}_{m!}$

$a^{k'}(a^{k})^{i}a^{k''}$ ⟵ $\overset{x}{a^{k'}}\,\overset{y}{a^{k}}\,\overset{z}{a^{k''}}$

$i=0 \rightarrow a^{m!-k}$

$1 \leq k \leq m$ ⟶ $k=m$ باشد $x=\lambda$ برای است.

به نشان دهیم فاکتوریل هیچ عددی نیست

$(m-1)! < m! - k < m!$ ⟹ $m! - k$ بین دو فاکتوریل متوالی است پس خودش نمی‌تواند فاکتوریل عددی باشد پس

$1 \leq k \leq m$

$a^{m!-k} \notin L$

مثال چهارم)

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

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

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

مسئله 125) پیدا کردن یک زیر رشته pumping loop نزدیک‌تر به سمت چپ

پیدا کردن جایی اولین حالتی که تکرار شده را از سمت راست پیدا کنیم

دانشکده مهندسی کامپیوتر
اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

نشان دهید که $L=\{a^n b^k c^{n+k} \mid n,k \geq 0\}$ نوع سوم نیست ،

به طور مستقیم با استفاده از قضیه pumping می‌توان نشان داد.

ولی برای اینکه به طور غیرمستقیم نشان دهیم یک همومورفیزم تعریف می‌کنیم

$h(a)=a$

$h(b)=a$

$h(c)=c$

$h(L)=\{a^n b^k c^{n+k} \mid n,k \geq 0\}$

$=\{a^{n+k} c^{n+k} \mid n,k \geq 0\}$

$=\{a^m b^m \mid m \geq 0\} \Rightarrow$ تحت همومورفیزم بسته است چون

نشان دادیم که $a^n b^n$ نوع سوم نیست پس این هم نیست در واقع از برهان خلف استفاده کردیم.

برای زبان‌های نوع سوم (Context Free Language)

مثال‌های 5 ← مشتقات

6 ← ساده‌سازی‌ها برای زبان‌های نوع دوم

7 ← پذیرنده برای زبان‌های نوع دوم

8 ← خصوصیات بستاری زبان‌های نوع دوم

گرامرهای نوع دوم:

$A \rightarrow \alpha$

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

$A \in V$

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

اشتقاق از سمت چپ: Left most derivation

" " " راست: Right "

درخت اشتقاق: derivation Tree (parse Tree)

ابهام: (ambiguity)

Subject:
Year. Month. Date. ( )

$S \to AB$
$A \to aA$
$B \to bB$
$A \to a$
$B \to b$

$S \Rightarrow AB \Rightarrow aAB$ ← LMD
$\Rightarrow aaAB$ متغیر سمت چپ را جایگزین
$\Rightarrow aaaB$ می‌کنیم
$\Rightarrow aaabB$
$\Rightarrow aaabb$

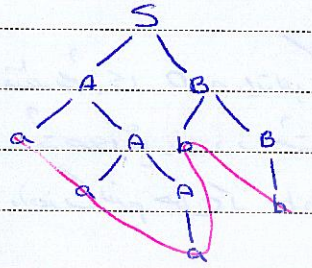
$S \Rightarrow AB \Rightarrow AbB$ RMD
$\Rightarrow Abb$
$\Rightarrow aAbb$
$\Rightarrow aaAbb$
$\Rightarrow aaabb$

ممکن است قسمت‌ها را جدا از درخت نشان دهیم یعنی برای هر قاعده یک درخت

سمت چپ قاعده

P = { S(A, B), A(a, A), B(b, B), A(a), B(b) }

اگر از این‌ها برای استفاده کنیم جواب حاصل یک درخت خواهد شد.

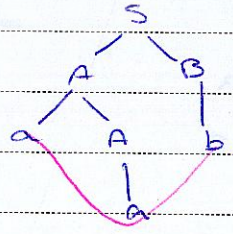


→ aaabb

کاراکترهایی که برگ‌ها علامت‌های پایانی باشند و متغیر نباشند که نشان جایگزینی کرد.

برای بدست آوردن رشته برگ‌ها را از چپ به راست می‌خوانیم

ab



⇒ aab

دانشکده مهندسی کامپیوتر و فناوری
اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

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

مسئله (5,165) نشان دهید گرامر مبهم است؟

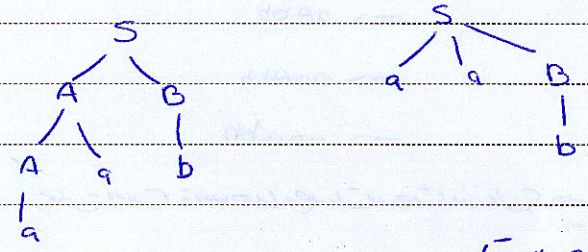
$S \to AB \mid aaB$

$A \to a \mid Aa$

$B \to b$

باید به دنبال رشته باشیم که دارای دو درخت اشتقاق یا دو LMD یا دو RMD باشد

aab ⇐ تنها رشته‌ای در این گرامر که دارای این شکل است



Point: الگوریتمی وجود ندارد که با گرفتن گرامر به ما بگوید که مبهم است

← رشته‌ای مبهم بودن را باید رفع ابهام کنیم (Remove ambiguity)

یا باید علت ابهام را پیدا کنیم و بعد آن را حذف کنیم

مثلاً رشته‌ی aab را تولید کنیم تقشی در تولید $S \to aaB \Rightarrow$

رشته‌های دیگر ندارد و رشته aab توسط $S \to AB$ هم ایجاد می‌شود پس می‌توانیم $S \to aaB$

را حذف کنیم چون که aab توسط $S \to AB$ هم تولید می‌شود

متفاوت نیز در پس هم LMD و RMD است

$S \Rightarrow AB$
$\Rightarrow AaB$
$\Rightarrow aaB$
$\Rightarrow aaB$

$S \Rightarrow aaB \to$
$= aab$

انجمن علمی مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

جلسه شانزدهم 87/8/25

ابهام (ambiguity)

یک گرامر نوع دوم مبهم است اگر یک جمله تولید کند که دارای حداقل دو درخت اشتقاق باشد

مثال) گرامر عبارات جبری ساده

یک تعریف بازگشتی برای عبارات جبری ساده

expression — identifier

$E \rightarrow I$

$E \rightarrow E+E$ (یعنی دو expression)

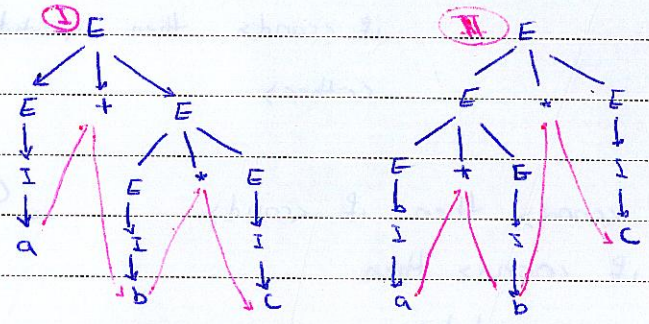
$E \rightarrow E*E$

$E \rightarrow (E)$

$I \rightarrow a|b|c$

این گرامر مبهم است. باید جمله پیدا کنیم که دارای دو درخت اشتقاق باشد.

یک نمونه $a+b*c$



به همین دلیل این گرامر مبهم است

ابهام چرا مهم است؟

هر جمله دارای معنا است که از طریق درخت اشتقاق مشخص می‌شود. مهم بودن ← جمله دارای دو معنای متفاوت است در این جا جمله دارای دو تفسیر از نظر عبارات است و این که کدام عملگر اول ارزیابی شود ← جمله دارای ارزش‌های مختلفی می‌شود

ارزش ← compiler درخت اشتقاق می‌سازد و ارزش را پیدا می‌کند

۱- ابتدا ارزش گره‌های برگ را پیدا می‌کند که در قسمت پایین هستند و به بالای رود و در واقع سعی می‌کند که برای درخت‌های زیرین پیدا می‌کند و به بالا حرکت می‌کند

دانشکده مهندسی کامپیوتر و فناوری
آزمایشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

کدی که عملی که برای این درخت تولید کند

(I)
load b = 2
Mul c 3 =>
Add a = 1 => AC = 7

(II)
load A
Add b
Mul c => AC = 9

اگر معلوم نباشد کدام درخت برای ارزیابی استفاده می‌شود نمی‌دانیم کدام ارزش صحیح است.

با اولویت‌بندی این مسئله حل می‌شود. اگر اولویت با ضرب (I)

جمع ... (II)

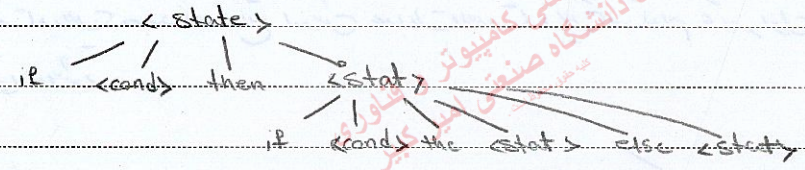
مثال‌های دیگر: جملات شرطی در زبان‌های برنامه‌سازی

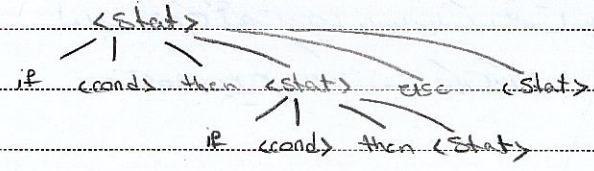
<statement> → if <cond> then <statement> |
if <cond> then <statement> else <statement> |
<other>

if <cond> then
if <cond> then
<stat>
else
<stat>

→ این جمله دارای دو درخت اشتقاق است

→ مشخص نیست کدام if وابسته است





دانشکده مهندسی کامپیوتر
اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

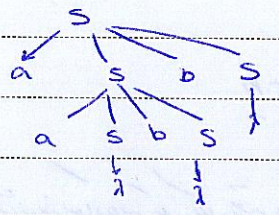
مثال) 12-150 نشان دهید که مبهم است

$S \to aSbS \mid bSaS \mid \lambda$ → تعداد a ها = تعداد b ها

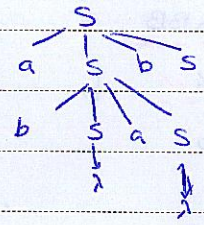
در بین کوتاه‌ترین رشته‌ها دنبال رشته مورد نظری می‌گردیم اگر موفق نشدیم به دنبال دومین کوتاه‌ترین رشته‌ها می‌رویم

ab ✓
ba ✓ ⟹ هیچ راه وجود دارد

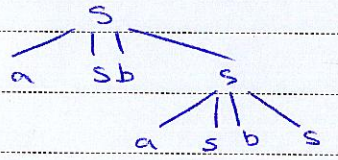
الگوریتمی پیدا می‌کنیم که رشته‌های به طول 4 به دست دهد



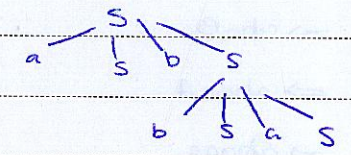
aabb



abab



abab



abba

می‌توانیم وقتی به اولین درخت اشتقاق رسیدیم متوقف شویم

راه دیگر پیشنهاد از شما بهتر است و نشان بدهیم چنین رشته وجود دارد و سپس خود رشته را پیدا کنیم باید اطلاعاتی در مورد زبانی که این تولید می‌کند و رشته بسازیم

تعداد رشته‌های به طول 4 که این رشته تولید می‌کند $= \frac{4!}{2!\,2!} = 6$

تعداد درخت‌های اشتقاقی که داریم و تجزیه به طول 4 می‌شود = 6 تا

طبیعتاً بیش از یک درخت در این مجموعه‌ها متعلق به یک رشته واحد است.

بعضی از جملات بیش از یک درخت اشتقاق داشته باشند

دانشکده مهندسی کامپیوتر
اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

یکی از انواع گرامرهای نوع دوم

گرامرهای ساده (Simple Grammer)

یک گرامر نوع دوم ساده می‌باشد اگر

1. قواعد به فرم زیر باشند

$S \to aX \quad , \quad X \in V^*$ بیشتری از نخیره

2. هیچ دو قاعده‌ای که سمت چپشان یکسان است سمت راستشان با یک علامت یکسان شروع نمی‌شود.

مثال)

$S \to aAB \mid bBB$

$A \to bA \mid a \mid \varnothing$ پس این حالت ساده نبود.

$B \to aA \mid b$

به این چیزی می‌گویند ساده که یک نکته یا یک عضو ساده وجود دارد.

نشان دهید این رشته عضو زبان است

$S \Rightarrow aAB$

$\Rightarrow abAB$

$\Rightarrow abaB$

$\Rightarrow abaaA$

$\Rightarrow abaaa$

مثال: دو قاعده‌ای که سمت چپ S سمت راست با a شروع شده باشد.

$S \Rightarrow aAB$

$\Rightarrow abAB$

$\Rightarrow abaB$

$\Rightarrow abaaA$

$\Rightarrow abaaa$

اگر رشته مثال طول باشد می‌توانیم با تعداد مراحلی که برابر است با طول رشته می‌توانیم به آن برسیم

جدا کنید ... می‌توانیم $O(|w|)$

اعلام نظر کنیم

دانشکده مهندسی کامپیوتر و فناوری
دانشگاه صنعتی امیرکبیر

PAPCO

Subject:
Year. Month. Date. ( )

مثال) نشان دهید که هر S-grammer غیرمبهم است

اگر برای S بیش از یک نمایش بتوانیم بدهیم غیرمبهم است.

اگر بشود رشته تولید شده یک LMD خاص داشت و در هر مرحله دو گزینه نداریم برای جایگزینی و دقیقاً می‌دانیم چی باید جایگزین کنیم ← طبق تعریف

← زبان ذاتاً مبهم

یک زبان ذاتاً مبهم است اگر هیچ گرامر غیرمبهمی نتوان برای آن بدست آورد

Inherently ambigous

مثال) نشان دهید یک زبان منظم نمی‌تواند ذاتاً مبهم باشد.

مثال) آیا این اشکال پذیر است که یک گرامر منظم مبهم باشد؟ درخت‌های برای تبدیل نمایش برای زبان نمایش،

اشتقاق دیگر درخت‌های منظم هم LMD ، RMD ← فقط یک شیوه ظاهر می‌شود. S => aA => abB

=> abbB

=> abba

یعنی اینکه برای رسیدن به یک رشته دو راه و دو طریق داشته باشیم حساسیت دو مسیر وجود داشته باشد که ما از حالت اولیه به حالت نهایی برسانند ← DFA غیرمبهم خواهد بود ← چون یک مسیر وجود دارد

NDFA ← دو مسیری که منتهی به یک راه می‌شود دو اشتقاق تبدیل می‌شود ← اشکال پذیر است اگر بتوانیم یک DFA از آن بسازیم پس نمی‌تواند ذاتاً مبهم باشد.

دانشکده مهندسی کامپیوتر و فناوری
اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

جلسه هفدهم 87/8/27

- ساده سازی برای برهای نوع دوم (context Free) فصل ششم
- فرم های نرمال (Normal Forms) برای برهای نوع دوم

ساده سازی برای برهای نوع دوم:

- حذف قواعد بازگشتی از سمت چپ — Left Recursive Production
- حذف قواعد بازگشتی از سمت چپ — Right Recursive Production
- حذف قواعد بدون استفاده — Useless Production
- حذف قواعد واحد — unite Production
- حذف قواعد λ — λ Production

A

قواعد بازگشتی از سمت چپ (مثال 21,170)

یک قاعده بازگشتی روی A دارای فرم زیر می باشد.

$A \rightarrow Ax$ — بازگشتی از سمت راست A در سمت الیه راست قرار گیرد.

$x \in (N \cup T)^*$

- روشی برای حذف قواعد بازگشتی: مثال 21,170

- حذف قواعد بازگشتی از سمت چپ برای A:

قواعدی که سمت چپ آنها A است جدا می کنیم و به دو گروه تقسیم می کنیم:

1. آنهایی که بازگشتی از سمت چپ هستند $A \rightarrow Ax_1 | Ax_2 | \dots | Ax_n$
2. بقیه روابطی (هیچ کدام با A شروع نمی شوند) $A \rightarrow y_1 | y_2 | \dots | y_m$

$$\begin{cases} A \rightarrow y_i \mid y_i Z & 1 \le i \le m \\ Z \rightarrow x_i \mid x_i Z & 1 \le i \le n \end{cases}$$

Z متغیر جدید

y_i ها A هیچ وقت شروع نمی شوند چون ضرورتاً جدا کردیم

PAPCO

Subject:
Year. Month. Date. ( )

مثال) متغیر بازگشتی از سمت چپ روی A داریم.

$S \to ABa \mid BBa$

$A \to Aab \mid AbB \mid Ba \mid BA$

$A \to a$

$B \to b$

قواعد بازگشتی روی A را جدا می‌کنیم

$$\begin{cases} A \to \underbrace{Aab}_{x_1} \mid \underbrace{AbB}_{x_2} \\ A \to \underbrace{Ba}_{y_1} \mid \underbrace{BA}_{y_2} \mid \underbrace{a}_{y_3} \end{cases}$$

$$\begin{cases} A \to Ba \mid BaZ \mid BA \mid BAZ \mid a \mid aZ \\ Z \to ab \mid abZ \mid bB \mid bBZ \end{cases}$$

چرا قواعد بازگشتی از سمت چپ را حذف می‌کنیم

وقتی بخواهیم با الگوریتم بعضیت parser تولید کنیم ایجاد مشکل می‌کند چون ممکن است وارد loop نامتناهی شود

در درس Compiler

چرا این روش کار می‌کند؟

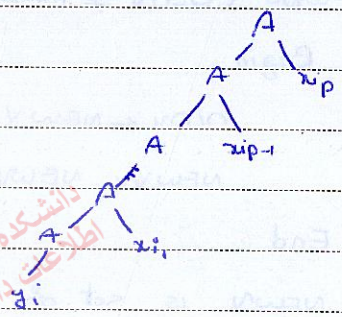
$$\begin{cases} A \to Ax_1 \mid Ax_2 \mid \dots \mid Ax_n \\ A \to y_1 \mid y_2 \mid \dots \mid y_m \end{cases}$$

پشت سر هم تولید می‌شود

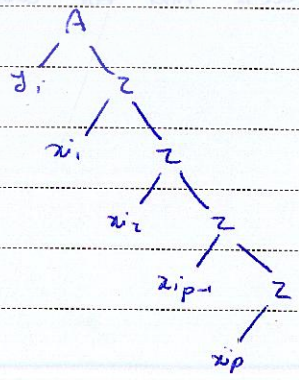
$\Rightarrow$ $y_i x_1 x_2 \dots x_p$

$A \to y_1 \mid y_i Z$

$Z \to x_i \mid x_i Z$



$$\begin{cases} A \to y_i \mid y_i Z \\ Z \to x_i \mid x_i Z \end{cases} \Longrightarrow$$



دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

حذف قواعد بیمورد استفاده

- متغیرهای قابل استفاده (Useful Variables) متغیر X قابل استفاده است اگر دو چیز باشد:

جنبه اول ① $X \xrightarrow{*} w$, $w \in T^*$ — طی مراحلی بتوانیم به یک رشته پایانی برسیم

جنبه دوم ② $S \xrightarrow{*} \alpha X \beta$ — دارای فرم جمله‌ای که X در آن باشد

- قواعد غیرقابل استفاده

حداقل یک متغیر غیرقابل استفاده در آن باشد.

رویکرد اول

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

Output : S.t of all variables $X \xrightarrow{*} w$, $w \in T^*$

```
Begin
    OLDV := {}
    NEWV := {A | A → x , x ∈ T*}
    while (OLDV ≠ NEWV) do
    Begin
        OLDV ← NEWV
        NEWV ← NEWV ∪ {A | A → α , α ∈ (T ∪ OLDV)*}
    End
    NEWV is set of variables X
    such that for every y ∈ X , y ⇒* w
```

تمام متغیرهایی که دست‌کم یک قاعده دارند که به حالت پایانی است.

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:

Year. Month. Date. ( )

مثال 6 ـ 169

	OLDV	NEWV
$S \to a \mid aA \mid B \mid C$		
$A \to aB \mid \lambda$	$\{\}$	$\{S, A, D\}$
$B \to Aa$ (OLDV, $\in T$)	$\{S, A, D\}$	$\{S, A, D\} \cup \{B\}$
$C \to cCD$ ← چون بازگشتی	$\{S, A, D, B\}$	$\{S, A, D, B\} \cup \{\}$
$D \to ddd$		

مفید از جنس اول

رویه دوم:

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

Output: Set of all variables X such that for every $Y \in X$ we have $S \overset{*}{\Rightarrow} xYB$

← مفید بودن از جنس دوم

```
Begin
    OLDV ← {}
    NEWV ← {S}
    while (OLDV ≠ NEWV) do
    Begin
        OLDV ← NEWV
        NEWV ← NEWV ∪ {X | A → α, A ∈ OLDV
                                and X appears in α}
    end
    'NEWV'
```

تنها متغیری که می‌دانیم از جنس دوم مفید است

select از S

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

مثال 6 ـ 169

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

خروجی ← مفید از جنس دوم

Subject:
Year. Month. Date. ( )

چگونه با استفاده از این دو مرحله قواعد را حذف کنیم؟

1. جدا نمودن متغیرهای غیر از جنبه اول. قواعد این دو را نگاه می‌کنند و به عنوان خروجی قواعدی را می‌دهند که از جنبه اول خنثی هستند و به عنوان ورودی به مرحله دوم می‌رسیم

2. مرحله دوم اعمال شدن مرحله مرحله اول

برای مثال (196-6)

$\{S, A, B, D\}$ اعمال مرحله اول $\longrightarrow$

$$\begin{cases} S \to a \mid aA \mid B \\ A \to aB \mid \lambda \\ B \to Aa \\ D \to ddd \end{cases}$$

از S قابل وصول نیست

به عنوان ورودی به مرحله دوم می‌دهیم $\longleftarrow$ $\{S, A, B\}$

$$\begin{cases} S \to aA \mid a \mid B \\ A \to aB \mid \lambda \\ B \to Aa \end{cases}$$

اول مرحله دوم

$$\begin{cases} S \to a \mid aA \mid B \mid C \\ A \to aB \mid \lambda \\ B \to Aa \\ C \to cCD \\ D \to ddd \end{cases} \xrightarrow{\{S, A, B, D\}} \begin{cases} S \to a \mid aA \mid B \\ A \to aB \mid \lambda \\ B \to Aa \\ D \to ddd \end{cases}$$

دانشکده مهندسی کامپیوتر و فناوری
اطلاعات دانشگاه صنعتی امیر کبیر

در مرحله دوم متغیرهای که نتیجه گرفتیم کدام درست است چون در راه دوم قواعدی داریم که غیر قابل استفاده است ($D \to ddd$) پس راه اول صحیح است

قواعد واحد: $A \to B$

سمت چپ و راست متغیرهای باشند

قواعد λ: ترتیبی جدی λ نباشد برای درست آوردن گرامر نیازی به λ نمی‌شود

$A \to \lambda$ سمت راست λ باشد

Subject:
Year. Month. Date. ( )

جلسه هجدهم 87.9.4

ساده سازی گرامرهای نوع سوم:

- حذف قواعد بر اشتی از سمت چپ
- " " " " " راست
- " " " بدون استفاده
- " " " واحد
- " " " λ

$A \to B$ $A,B \in V$ سمت چپ و راست تغییر نکند

حذف قواعد واحد

مثالی از کتاب:

$S \to Aa \mid B$

$B \to A \mid bb$

$A \to a \mid bc \mid B$

قواعد واحد این گرامر $\to S \to B$

$B \to A$

$A \to B$

قواعد غیر واحد $\to S \to Aa \mid bb \mid bc \mid a$

$B \to bb \mid a \mid bc$

$A \to a \mid bc \mid bb$

یک گراف وابستگی از روی قواعد واحد می‌سازیم

$S \to B \leftrightarrows A$ → dependency Graph

از روی این گراف اثرات قواعد واحد را پیدا می‌کنیم

اثرات قواعد واحد

$S \xrightarrow{*} A$

$S \xrightarrow{*} B$

$B \xrightarrow{*} A$

$A \xrightarrow{*} B$

← هیچ کدام از S به A وجود دارد

با استفاده از قواعد واحد و اثرات می‌توانیم S را به B جایگزین کنیم

اگر بخواهیم این قواعد را حذف کنیم ولی گرامری معادل داشته باشیم باید این اثرات را داشته باشیم

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

اگر اجازه بدهیم که S مستقیماً به آن چیزهایی که B می‌رفته برود پس این اثر را داریم $S \to B$

برای حذف این قواعد باید اثرات آن ابتدا شود مثلاً برای $S \xrightarrow{*} A$ اگر اجازه دهیم S مستقیم به آن چیزهایی که A می‌رفته برود این اثر را مد نظر نیستیم

← برای حذف λ در نوع دومی می‌توانیم λ را بی‌مدار بدست آوریم که خالی از قواعد واحد باشد

برای حذف نوع دوم می‌توان λ را بی‌مدار بدست آورد که عاری از قواعد واحد باشد

حذف قواعد λ

طی دو مرحله انجام می‌شود

- پیدا کردن متغیرهای پوچ (null)

- حذف قواعد واحد ← با استفاده از متغیرهای پوچ

متغیرهای پوچ: متغیر x ، Nullable است اگر $X \xrightarrow{*} \lambda$

پیدا کردن متغیرهای پوچ

یک الگوریتم می‌نویسیم

Given : G(N, T, P, S)

Output: Set of Nullable variables

Begin

متغیرهایی که مستقیماً λ است

OLDNULL ← ∅ ; NEWNULL ← {A | A ← λ is in P}

while (OLDNULL ≠ NEWNULL) do

begin

OLDNULL ← NEWNULL

NEWNULL ← NEWNULL ∪ {A | A → α , all symbols in α are nullable} $\alpha \in OLDNULL^*$

end

مستقیماً فقط از متغیرهای NULL تشکیل شده‌اند

'NEWNULL' is the set of Nullable variables

end

دانشکده مهندسی کامپیوتر و علوم اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

چون Nullable نیستند S نداریم

مثال از کتاب) گرامری که به عنوان ورودی داده می‌شود

$D \Rightarrow S \to ABaC$

$A \to BC$

$B \to b \mid \lambda$

$C \to D \mid \lambda$

$D \to d$

	OLDNULL	NEW NULL
	$\{\}$	$\{B, C\}$
	$\{B, C\}$	$\{B, C\} \cup \{A\}$
	$\{A, B, C\}$	$\{A, B, C\} \cup \{\}$

حذف قواعد λ با استفاده از متغیرهای اچ

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

Find: $G' = (V, T, P', S)$, such that P' doesn't contain λ-production

چون قرار است P' را پیدا کنیم

- P' is constructed as follows

if $A \to X_1 X_2 \dots X_n$ is in P

then add $A \to \alpha_1 \alpha_2 \dots \alpha_n$ to P'

where - $\alpha_i = X_i$ if X_i is not Nullable

- $\alpha_i = \lambda$ or X_i if X_i is "

- Not all α_i's are Nullable

حل مثال برای مثال کتاب)

متغیرهای

$P - S \to ABaC$ — A، B، C هم می‌توانند λ بشوند هم خودشان

این حالات را خواهیم داشت

P'

$S \to a$ ← همگی λ

$S \to AaC \mid BaC \mid ABa$ ← یکی λ

$S \to Aa \mid Ba \mid aC$ ← دو تا λ

$S \to ABaC$ ← هیچ کدام λ نشوند

انجمن علمی مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

$P \to A \to BC$

$P' \to A \to BC \mid A \to B \mid A \to C$ → دقت λ ی تواند باشد طبق حدس آخر جدول دوباره قاعده λ ایجاد خواهد شد.

$P \to B \to b$

$P' \to B \to b$

$P \to C \to D$

$P' \to C \to D$

$P \to D \to d$

$P' \to D \to d$

مسئله ← اگر امری به ما داده شد که می‌خواهیم گرامری بدست آوریم که عاری از قواعد λ و واحدی استفاده باشد به چه ترتیب باید این الگوریتم‌ها را اعمال کنیم و یا استفاده کنیم که مجموع بدست آمده عاری از این سه باشد

- حذف قواعد λ
- " " واحد
- " " بدون استفاده

6 حالت می‌توانیم داشته باشیم کدام را باید اعمال کنیم

در کتاب مثال‌هایی هست که هر کدام از این‌ها را نشان می‌دهد غلط است

این جواب است

1	1	2	2	3	3
2	3	1	3	1	2
3	2	3	1	2	1

در هر مسئله زیر مرتبه باید 2 بار یک انجام گیرد

13 ـ 169 یک مثال بیاورید که حذف قواعد λ باعث ایجاد قواعد واحد جدید گردد.

مسائل 14 و 15 ← برای حذف دو ترتیب باقی‌مانده

دانشکده مهندسی کامپیوتر - شاهرودی
اطلاعات دانشگاه صنعتی امیر کبیر

Subject:
Year. Month. Date. ( )

فرم‌های نرمال برای گرامرهای نوع دوم:

Normal Form ← یک گرامر را می‌توان تبدیل به این نوع فرم باشد

- فرم نرمال چامسکی — Chamsky NF
- فرم نرمال گریباخ — Griebach NF

Chamsky NF

یک گرامر نوع دوم به فرم نرمال چامسکی می‌باشد اگر قواعد به فرم زیر باشد

$A \to BC$ ← یا دو متغیر در سمت راست است

$A \to a$ ← یا یک علامت پایانی

$A, B, C \in V$

$a \in T$

$S \to AA \mid BA, a$

$A \to AA \mid a$

$B \to BA \mid a$

→ یک مثال از Chamsky NF

چگونه اثبات کنیم؟

برای اثبات باید الگوریتمی تهیه کنیم که گرامرها را به این نوع تبدیل کند

مثال:

$S \to ABaCb \mid ABa \mid b$

$A \to aA \mid b$

$B \to bB \mid b$

$S \to ABaCb$

$$\left\{\begin{array}{l} S \to AB X_a C X_b \\ X_a \to a \\ X_b \to b \end{array}\right. \Rightarrow \left.\begin{array}{l} S \to AX_1 \\ X_1 \to BX_2 \\ X_2 \to X_a X_3 \\ X_3 \to C X_b \end{array}\right\} \Rightarrow$$

به صورت نرمال در آمد

تبدیل شده به نرمال فرم

مرکز تحقیقات دانشگاه صنعتی امیرکبیر
مهندسی کامپیوتر و فناوری اطلاعات

Subject:
Year. Month. Date. ( )

$S \to ABa$ $\quad S \to ABX_a$, $X_a \to a$ $\Rightarrow \begin{cases} S \to AX_4 \\ X_4 \to BX_a \end{cases}$

$A \to b$

$A \to aA$ $\quad A \to X_aA$, $X_a \to a$

$B \to bB$ $\quad B \to X_bB$, $X_b \to b$

$B \to b$

Coriebach NF

قواعد به فرم زیر می‌باشند

$A \to aX$

$a \in T$

$X \in N^*$

$S \to aABC \mid aAA \mid a$

$A \to aA \mid b$

$B \to bB \mid b$

مثال) 9-177

$S \to aSb \mid ab$

$S \to aSb$ $\quad \begin{cases} S \to aSB \\ B \to b \end{cases}$ $\quad S \to ab$ $\quad \begin{cases} S \to aB \\ B \to b \end{cases}$

$A \to y, yz$

$z \to x_i, x_iz$

$S \to SS \mid Sa \mid ab$

مثال

ابتدا باید قواعد بازگشتی را حذف کنیم (از سمت چپ)

$\begin{cases} S \to \underbrace{SS}_{x_1} \mid \underbrace{Sa}_{x_2} \\ S \to \underbrace{ab}_{y_1} \end{cases} \Rightarrow \begin{cases} S \to ab \mid abZ \\ Z \to SZ \mid S \mid a \mid aZ \end{cases}$

ممکن است قواعد واحد ایجاد شود که باید آنها را حذف کنیم

انجمن مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

$S \to ab \mid abZ$

$Z \to SZ \mid a \mid aZ \mid ab \mid abZ$

بعد از این مرحله باید قواعد تک تک بگیریم

$S \to ab \Rightarrow \begin{cases} S \to aB \\ B \to b \end{cases}$ $\quad S \to abZ \Rightarrow \begin{cases} S \to aBZ \\ B \to b \end{cases}$

قبل از اینکه به گره دوم برویم یک بحث ساده سازی انجام دهیم:

$$\begin{cases} A \to y_1 B y_2 & y_1, y_2 \in (V \cup T)^* \\ B \to x_1 \mid x_2 \mid \dots \mid x_n & x_i \in (V \cup T)^* \end{cases}$$

B می‌تواند جایگزین شود در A $\Rightarrow A \to y_1 x_1 y_2 \mid y_1 x_2 y_2 \mid \dots \mid y_1 x_n y_2$

$\begin{cases} Z \to \overset{B}{S}Z \\ S \to aB \mid aBZ \end{cases} \Rightarrow$ $\quad$ لا را نداریم

$\to Z \to aBZ \mid aBZZ$

مصرف این مثال داریم:

$Z \to SZ \Rightarrow Z \to aBZ \mid aBZZ$

$Z \to ab \Rightarrow \begin{cases} Z \to aB \\ B \to b \end{cases}$

$Z \to abZ \Rightarrow \begin{cases} Z \to aBZ \\ B \to b \end{cases}$

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

اگر اول قواعد برگشتی از جمله چپ را حذف نکنیم وقتی به ساده سازی ساده برسیم دوباره قواعدی مانند $Z \to SZ$ ایجاد خواهد شد.

بعد از ساده سازی به قواعد شکل دلخواه داریم یا یک ترفند ساده حل می‌شود

Subject:
Year. Month. Date. ( )

جلسه سیزدهم 87/9/9

تئوری زبان های نوع دوم:

- مقدمات
- ساده سازی گرامرهای نوع دوم
- فرم های نرمال ← فرم نرمال چامسکی
  فرم نرمال گریباخ

فرم نرمال چامسکی اشاره شده در صفحه 177 - 8

$A \to aB$
$A \to Ba$
$A \to a$

در زبان های نوع سوم می توانستیم هم خطی از سمت راست داشته باشیم هم از سمت چپ ← در نوع دوم می توانیم

برای اثبات این باید الگوریتمی پیدا کنیم که زبان را بگیرد و به این فرم تبدیل کند

مسئله 12 و 178 ← two standard NF

حالت خاصی از گریباخ است

$A \to aBC$
$A \to aB$
$A \to a$

برای λ در گریباخ ضرورت به وجود نداشت ولی
λ اندازه اش می تواند 0 یا 1 یا 2 باشد

فوائد NF ها:

وقتی فرم نرمال باشد فرم قواعد محدود هستند ← کار کردن با آنها راحت است و برای اثبات یک چیزی می توانیم از فرم نرمال آن استفاده کنیم

با استفاده از NF ها می توانیم الگوریتم های بهتری خیلی کارآمدتر بیاوریم

برای این مطلب مثال 6.3.178

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

پیچیدگی گرامر G

مسئله (19-170)

$Complexity(G) = \sum_{A \to v \in P} (1 + |v|)$ ← طول سمت راست هر قاعده را به یک جمع می‌کند

$S \to aAB \mid bB \mid \lambda$ $\quad (3+1)+(2+1)+(0+1)$

$A \to aA \mid b$ $\quad (2+1)+(1+1)+$

$B \to bB \mid a$ $\quad (2+1)+(1+1) = 18$

پیچیدگی گرامر زیاد است اگر تعداد قواعد زیاد باشد یا طول سمت راست بیشتر باشد

مثال: حذف قواعد بدون استفاده همیشه پیچیدگی را کاهش می‌دهد؟

در مورد حذف قواعد λ و حذف قواعد واحد چه می‌توان گفت؟

مسئله (20-170)

یک گرامر نوع دو G مینیمال (minimal) گفته می‌شود اگر داشته باشیم

$complexity(G) \leq complexity(\hat{G})$

برای هر $\hat{G}$ که زبان L را تولید می‌کند.

پیچیدگی گرامر دو مساوی هر گرامری باشد که زبان L را تولید می‌کند

آیا حذف قواعد بدون استفاده باعث تولید گرامر minimal می‌شود؟

نمی‌دانیم ← minimal بودن به طبیعت زبان مطلب بستگی دارد و نمی‌توان به چنین ساده‌سازی‌ها

برای آن رابطه ساده آمد ← هیچ ربطی به automata نیست و طبیعت زبان minimal باشد

باشگاه مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

فصل هشتم

پذیرنده برای زبان های نوع دوم

Push down Automata (PDA)

از سه جزء تشکیل شده که یک نوار شبیه به ماشین های متناهی

(read head) Finit Control یک

یک پشته

حالت جاری ماشین با اطلاعات ماشین در حال حاضر و علامت جاری، آنچه در stack است مشخص می شود

جدا از مشخص شدن حالت جاری stack هم عوض می شود

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

Q مجموعه حالت

Σ الفبای ما

T علامات مجاز که در پشته می توانند قرار گیرند

q_0 ، $q_0 \in Q$ ، حالت اولیه

Z ، $Z \in T$ ، علامت اولیه پشته و نقش شان دادن ته پشته است

F مجموعه علامت نهایی $F \subseteq Q$

δ Transition Function

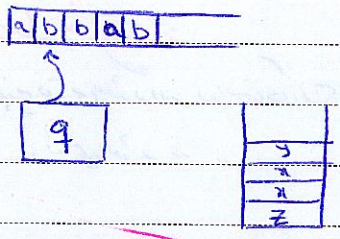
$\delta : Q \times (\Sigma \cup \{\lambda\}) \times T$

$\rightarrow$ a finite subset of $Q \times T^*$

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

configuration

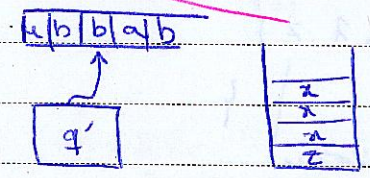


در پیش رو حالت q، سر b را بخوانیم pop stack می‌شود

$S(q,b,y) = \{(q',x)\}$

$\in T^*$

حالتی که در q است و چیزی است در stack x قرار می‌گیرد



سه configuration توسط یک سه‌تایی نمایش داده می‌شود.

(q, w, γ)

q ← حالتی که در آن قرار داریم

w ← قسمت خوانده نشده رشته روی tape

γ ← محتوای پشته

$(q, bbab, yzxz)$

$(q', bab, xxxz)$

$(q'', ab, xyxz)$

پذیرش (acceptance)

دو نوع پذیرش داریم:

دو تا پذیرش برای push down automata:

- پذیرش توسط حالت نهایی ✓ — acceptance by final state
- پذیرش توسط پشته خالی ✓ — acceptance by empty stack

دانشکده مهندسی کامپیوتر دکتر غفوری
اطلاعات دانشگاه صنعتی امیرکبیر

توسط حالت نهایی

اگر یک رشته پذیرفته می‌شود اگر در انتهای رشته ماشین در یکی از حالت‌های نهایی خود قرار گیرد.

$L(PDA) = \{ x \mid (q_0, x, Z) \overset{*}{\vdash} (q_f, \lambda, \gamma)$ for some $\gamma \in T^*, q_f \in F \}$

سه‌تایی اولیه

پیش رو حالت اولیه است

رشته خروجی می‌خواهیم باشد

Subject:
Year. Month. Date. ( )

پذیرش توسط پشته خالی:

- یک رشته پذیرفته می‌شود اگر در انتهای آن رشته پشته خالی باشد
- یک رشته " نمی‌شود " " " " " " خالی نباشد و یا قبل از رسیدن به انتهای رشته ماشین متوقف شود.

$$L(PDA) = \{ (q_0, x, Z) \vdash^* (q, \lambda, Z) \}$$

$Z = \lambda$

for some $q \in Q$

در انتهای رشته پشته خالی باشد و اینکه در چه حالتی باشد مهم نیست.

مثال:

$L = \{ a^n b^n \mid n \geq 0 \}$ => یک زبان نوع دوم است

a..a..a b..b..b

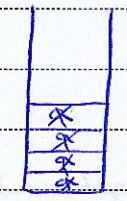
سعی می‌کنیم یک الگوریتم توسط پشته پیدا کنیم که این حسنه را حل کند و بعد آن را به این قالب ترجمه کنیم

aaaabbbb$

1 1 1 1

استفاده از پشته چک کنیم ← b به اندازه a

تعداد a = تعداد b



a را در stack می‌گذاریم تا به b برسیم

بعد پشته را چک می‌کنیم اگر a بود آن را پاک می‌کنیم

برای پذیرش رشته λ $\delta(q_0, \$, Z) = \{(q_f, \lambda)\}$ ← ته پشته بر پشته

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

$\delta(q_1, a, a) = \{(q_1, aa)\}$ → push a to top

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

$\delta(q_2, b, a) = \{(q_2, \lambda)\}$ → این کار را ادامه می‌دهیم تا به علامت بعدی $ برسیم

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

اگر n=0 باشد ← روی tape λ داریم باید بتوانیم آخر بپذیریم

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

در کتاب به عنوان علامت خنثی رشته را می‌گذارد به جای آن λ می‌گذارند

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

چون این جزء رشته توسط علامت خنثی است توی stack هم نیست می‌توانیم Z را برداریم یا برنداریم

در این صورت هم نیست می‌گذاریم نمی‌زنیم برای این شرایط مناسب است.

$(q_0, aaaabbbb, Z) \vdash$

$(q_1, aaabbbb, aZ) \vdash$

$(q_1, aabbbb, aaZ) \vdash$

$(q_1, abbbb, aaaZ) \vdash$

$(q_1, bbbb, aaaaZ) \vdash$

$(q_2, bbb, aaaZ) \vdash$

$(q_2, bb, aaZ) \vdash$

$(q_2, b, aZ) \vdash$

$(q_2, \lambda, Z) \vdash (q_f, \lambda, \lambda)$

رشته تمام شد

$(q_0, aabbb, Z) \vdash$

$(q_1, abbb, aZ) \vdash$

$(q_1, bbb, aaZ) \vdash$

$(q_2, bb, aZ) \vdash$

$(q_2, b, Z) \vdash$ دنبال حرف a برای این می‌گردیم وجود ندارد ← عدم پذیرش

$(q_0, aab, Z) \vdash$

$(q_1, ab, aZ) \vdash$

$(q_1, b, aaZ) \vdash$

$(q_1, \lambda, aZ) \vdash$ علامت انتهای رشته را دید ولی پشته خالی نیست

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

$(q_0, abab, z) \vdash$ این رشته را نخواهد پذیرفت.

اگر بتوانیم مسئله را تبدیل به رشته حل کرد خیلی راحت می‌توانیم به این ماشین تبدیل کنیم

مسئله قبل اگر به جای q_0 ، q_1 بگذاریم پذیرنده چه زبانی است؟

" " " " " " " q_1, q_2 " " " " " " " ؟

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

هر که دیدیم به جای 2 a ، a می‌گذاریم:

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

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

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

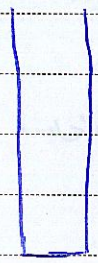
$\delta(q_1, b, a) = \{(q_2, \lambda)\}$

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

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

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

abbababbaaba



اگر برای هر a یک b یک بخوانند یک تیک اگر هم

اگر پشته خالی باشد هر چی خواندند stack می‌گذارد اگر چیزی خواند

مخالف سر stack باشد pop کند

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

جلسه بیستم 1387.9.11

Push down Automata

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

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

$L = \{w \mid w \in \{a,b\}^*, \#a's = \#b's\}$

مسئله را می‌خوانیم اگر stack خالی بود در داخل آن می‌گذاریم اگر حالت سر stack بود pop می‌کنیم در غیر اینصورت push می‌کنیم

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

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

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

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

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

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

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

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

زمانی که در q_1 هستیم انتظار نداریم پشته خالی شود در پشته دستگاه اضافه می‌کنیم که اگر در وسط پشته پشته خالی شد بتوانیم آن را pop کنیم

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

$\cancel{\delta(q_1, b, z) = \{(q_1, az)\}}$

اگر متغیر از q_0 استفاده کنیم این دو حالت حذف می‌شوند.

در این حالت مهم نیست پس می‌توانیم به جای q_1 از q_0 استفاده کنیم تا در حالت رشته باشیم

دلیل استفاده از دو حالت، اینست که حالت تعیین کننده نیست آن چه تعیین کننده است جفت $\delta(a,b)$ است

مثال:

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

اگر وسط این رشته را بدانیم که ساده است

w را push می‌کنیم تا به وسط برسیم

بعد w^R را می‌خوانیم اگر آنچه می‌خوانیم با سر پشته یکسان بود حذف می‌کردیم اگر در آخر پشته خالی بود در نتیجه رشته پذیرفته است اگر علامت‌ها یکسان نبود اعلام عدم پذیرش می‌کنیم

abbab babba

در حالت کلی باید وقتی به وسط رشته رسیدیم switch کنیم تا الان push می‌کردیم حداز وسط رشته باید pop کنیم

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:

Year. Month. Date. ( )

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

این یک مثالی است از یک PDA غیرقطعی

an deterministic PDA (NPDA)

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

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

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

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

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

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

هرچی بود باید push کنیم →

وقتی به q_2 برسیم دو حالتهای مختلفی دارد ←

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

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

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

این باعث غیرقطعی شدن ماشین می‌شود پس باید دو مورد مربوط به این هر تعدی را ضد نشان دهد ⇐

دو مورد که علامت حذف شده

pop push

یا برسیم به انتهای رشته pop می‌کنیم

هر کدام برای بدست آوردن PDA قطعی با مشکلات مواجه می‌شود چون یکی باید وسط رشته را به ما بگوید

برای حل این مشکل باید زبان را تغییر دهیم ←

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

دو مورد زیر به δ اضافه می‌شوند حالتهایی که باعث عدم قطعیت شدند حذف می‌شوند

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

$\delta(q_1, c, b) = \{(q_2, b)\}$ ← علت عدم قطعیت حذف شده است.

زبان نوع دوم قطعی Deterministic Content Free Language (DCFL)

یک زبان نوع دوم قطعی است اگر بتوان برای آن یک PDA قطعی بدست آورد.

زبان نوع دوم غیرقطعی Non-Deterministic Content Free Language (NCFL)

یک زبان نوع دوم غیرقطعی است اگر نتوان برای آن یک PDA قطعی بدست آورد

مثل WW^R که یک Content Free غیرقطعی است

علت عدم قطعیت اینست که وسط رشته را نمی‌دانیم

دانشگاه صنعتی امیرکبیر
دانشکده مهندسی کامپیوتر و فناوری اطلاعات

Subject:
Year. Month. Date. ( )

مثال ۴)

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

$L_1 = \{a^n b^n \mid n \geq 0\} \cup \{a^n b^{2n} \mid n \geq 0\}$ ← یک نمونه از زبان غیرتعینی

← اگر یک زبان داشته باشیم که نوع دوم هست یا نیست چجوری کار کنیم؟ یک راه نوع دوم رو با یک PDA برای آن طراحی کنیم فرقی نمی‌کند که DPDA یا NPDA باشد

مثال)

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

بشته را برای حفظ اطلاعات استفاده می‌کنیم تا بتوانیم کار کنیم

در این مثال تعداد a و b مهم است برای این کار a ها را push می‌کنیم اگر b داریم به ازای هر b یک a pop می‌کنیم اگر c داریم و پشته خالی بود این رشته عضو نیست

aaaaabbbbbccccc

مثال)

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

برای این مثال a ها push می‌کنیم از روی b ها می‌گذریم یعنی نضیه نمی‌کنیم به c که رسیدیم شروع می‌کنیم به pop کردن

$\delta(q_0, b, ?) = \{(q_1, ?)\}$ ← از روی b ها می‌گذریم

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

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

$\delta(q_1, b, a) = \{(q_1, a)\}$ ⇒ این دو رابطه از روی b ها

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

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

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

$\{a^n b^{2m} a^m \mid n, m \geq 0\}$ از روی a ها می‌گذریم

برای هر 2 تا b، یک a در پشته push می‌کنیم

دانشکده مهندسی کامپیوتر
اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

مثال) برای این رشته باید دو جفت را بشمریم و بگوییم به هم مساوی است.

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

از عملیات می‌توانیم دو بار استفاده کنیم

به هر طریقی سعی کنیم که به پشته حل کنیم نمی‌شود ← عملیات در پشته منطقی را استفاده می‌شود.

می‌شود برسیش یک DPA طراحی کرد پس زبان نوع دوم نیست

مثال)

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

a ها را push می‌کنیم تا وقتی پشته خالی نشده ادامه می‌دهیم

b می‌خوانیم a ها را pop می‌کنیم اگر پشته خالی شد

b ها push می‌کنیم تا به c می‌رسیم و b ها را pop می‌کنیم

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

مثال) یک زبان برای کدام است

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

w را push می‌کنیم

abbab abbab

حالا باید pop کنیم که باید به stack برویم که مقداری اطلاعات را از دست داده‌ایم چون باید هر چی روی آن هست pop کنیم

پس زبان نوع دوم نیست

مثال

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

این رد می‌شود جلو برید

اطلاعات از بین می‌رفت

علت عدم تعیین مثال *)

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

$L_1=\{a^n b^n \mid n\geq 0\}$

$L_2=\{a^n b^{2n} \mid n\geq 0\}$

یک رشته از L می‌آید که یا عضوی از L_1 یا عضوی از L_2 است بنابراین که عضو کدام باشد باید یک یا دو تا push a کنیم

$\delta(q_0,a,z)=\{(q_1,az),(q_2,aaz)\}$

حالت اولیه L_1 — L — حالت اولیه L_2

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

مثالی که یک PDA غیر قطعی را

$L = \{a^n b^m c^p \mid n=m \text{ or } m=p\} = L_1 \cup L_2$

$L_1 = \{a^n b^n c^p \mid n,p \geq 0\}$ و $L_2 = \{a^n b^m c^m \mid m,n \geq 0\}$

می‌بینیم برخورد داخلی است ← یا بگذاریم در پشته
یا از آن بگذریم ← علت عدم قطعیت

بسته به اینکه از L_1 آمده یا L_2

PDA طراحی کنید

$L = \{a^3 b^n c^n \mid n \geq 0\} \Rightarrow \delta(q_0, a, z) = \{(q_1, z)\}$

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

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

در واقع q_3 حالت اولیه زبانی می‌شود که $L_1 = \{b^n c^n \mid n \geq 0\}$ را کپی می‌کنیم ← از اینجا به بعد

L_1 نشان می‌دهد

مثال:

$L = \{w \mid n_a(w) = n_b(w) + 1\}$

یک b در پشته قرار می‌دهیم بعد چک می‌کنیم اگر پشته خالی بود پذیرفته می‌شود.

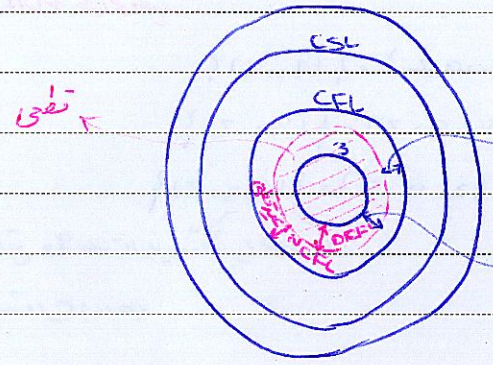
دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date.

جلسه بیست و یکم 87.9.16

PDA ← push down automata ← DPDA ← قطعی
NPDA ← غیرقطعی

CFL ← context Free Language ← NCFL ← غیرقطعی
DCFL ← قطعی



۳ قطعی

$L=\{a^n b^m \mid n=m \text{ or } m=2n\}$ زبان های نوع سوم به صورت زیر تبدیل شده و بخش می باشد

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

هر زبان حاوی:

یک مثال: یک گرامر به عنوان ورودی می گیرد

Given: Grammer $=(N, T, P, S)$

Find: NPDA accepting the same Language as Grammer defines

به عنوان خروجی یک NPDA معادل می دهد.

یک مثال مثال می زنیم:

$M=(\{q_0, q_1, q_2\}, T, V \cup \{z\}, \delta, q_0, Z, \{q_f\})$

حالت های اولیه پشته — حالت نهایی

فرض می کنیم گرامر ورودی به فرم نرمال گریباخ است.

دانشکده مهندسی کامپیوتر و فناوری
اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

مثال) δ هارا می‌خواهیم بسازیم

δ به چند قسمت تبدیل می‌شود

δ: $\boxed{\delta(q_0, \lambda, Z) = \{(q_1, SZ)\}}$ ← ابتدایی

← میانی

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

این δ ها را از روی قواعد حساب می‌کنیم

if $A \to ax$ then

Add $\delta(q_1, a, A) = \{(q_1, x)\}$

حالا قواعد مربوط به این مثال را می‌نویسیم:

	$\delta(q_1, a, S)$
$S \to aA$	$\delta(q_1, a, S) = \{(q_1, A)\}$
$A \to aABC$	$\delta(q_1, a, A) = \{(q_1, ABC)\}$
$A \to bB$	$\delta(q_1, b, A) = \{(q_1, B)\}$
$A \to a$	$\delta(q_1, a, A) = \{(q_1, \lambda)\}$
$B \to b$	$\delta(q_1, b, B) = \{(q_1, \lambda)\}$
$C \to c$	$\delta(q_1, c, C) = \{(q_1, \lambda)\}$

پایین را غیر منطقی می‌کند

یک LMD برای aabbbc

$S \Rightarrow aA$

$\Rightarrow aaABC$

$\Rightarrow aabBBC$

$\Rightarrow aabbBC$

$\Rightarrow aabbbC$

مستقل از زمینه

اولین تابعی که استفاده می‌کنیم δ ابتدایی است.

δ ابتدایی بعد از اینکه چیزی خواند S را وارد پشته می‌کند

$(q_0, aabbbc, Z) \vdash$

$(q_1, aabbbc, SZ) \vdash$

$(q_1, abbbc, AZ) \vdash$

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

$(q_1, bbbc, ABCZ) \vdash$

$(q_1, bbC, BBCZ) \vdash$

$(q_1, bC, BCZ) \vdash$

$(q_1, C, CZ) \vdash$

$(q_1, \lambda, Z) \vdash$ ⟹ از δ های استفاده می‌کنیم

(q_f, λ, Z)

← یک نظر یک بیان مراحل تولید و پذیرش رشته وجود دارد

و دو به حالت‌های بجای اعلام پذیرش رشته و تولید رشته

مثال (10-189)

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

این ماشین غیرقطعیست

① $\delta(q_0, a, Z) = \{(q_1, a), (q_2, \lambda)\}$

② $\delta(q_1, b, a) = \{(q_1, b)\}$

③ $\delta(q_1, b, b) = \{(q_1, b)\}$

④ $\delta(q_1, a, b) = \{(q_2, \lambda)\}$

اگر از حالت اولیه می‌تواند به دو حالت برود در حقیقت اجتماع دو زبان است و می‌تواند انتخاب کند

فرض می‌کنیم اولی باشد ← ببیند a ببیند ← Z را به a جایگزین می‌کند و دارد q_1 می‌شود و چون a برپشته است

پس بیاید از ② استفاده کند ←

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

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

گروه دیگری هم هست ← می‌تواند مستقیماً دارد q_2 شود ← تنها a را بخواند

$L = \{a\}$

پس در کل زبانی که می‌پذیرد ← $L = \{a\} \cup \{ab^n a \mid n \geq 1\}$

به چه شکلی رشته‌ها را بپذیرد رشته از کجا آمده

از بین گزینه‌هایی که دارد یک را انتخاب می‌کند و یک را به جهت نایل حصول خود برسانند که در

کامپیوتر جهت back tracking است ← غیرقطعی ← دقت تصمیم‌گیری می‌سازیم

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

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

زیرمجموعه‌ی متناهی رشته باشیم ← چون از سمت راست δ تعداد گزینه‌ها متناهی است.

$$\delta(q, a, x) = \{(q_1, yx), (q_2, \lambda x), \dots\}$$

$$\rightarrow \delta: Q \times (\Sigma \cup \{\lambda\}) \times T \longrightarrow 2^{Q \times T^*}$$ خودش متناهی است

جواب این حالت را پیدا کنیم

این چه زیرمجموعه‌ای می‌تواند باشد از مجموعه‌ی نامتناهی

در این حالت سمت راست δ تعداد گزینه‌ها متناهی است که خیلی می‌شود

یعنی گزینه درستش معنی ندارد چون اگر بخواهیم بین آن‌ها انتخاب کنیم باید آن‌ها را بشناسیم که بی‌نهایت زمان احتیاج است.

$$\delta: Q \times (\Sigma \cup \{\lambda\}) \times T \longrightarrow 2^{(Q \times (T \cup T^* \cup \{\lambda\}))}$$ ⇐ یک شرط دارد

صفحه 189-15

از T^* یا از T^2 یا T' یا λ

رشته‌ی با طول 2

این زیرمجموعه متناهی خواهد بود

سر رشته را با رشته‌ی با طول 2 را جایگزین می‌کنیم و یا pop می‌کنیم

اگر δ ما به این صورت باشد آیا می‌توانیم همه‌ی کارهایی که قبلاً انجام می‌دادیم الان هم انجام دهیم

آیا این ماشین معادل * است؟

بله معادل هست. هر چیزی که با * می‌ساختیم با این هم می‌توانیم یعنی می‌توانیم شبیه‌سازی کنیم ← در این جا باید ببینیم می‌توانیم هر رشته‌ای با هر طولی را با این تابع سر رشته قرار داد ← با تعداد δ این کار انجام می‌شود

← با تعداد از این δ ها می‌توانیم هر طولی را بسازیم

آیا این ماشین هم معادل است؟

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

خیر نیست زیرا نمی‌تواند طول‌های خود را شبیه‌سازی کند

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

فصل هشتم:

خصوصیات زبان‌های نوع دوم

- دو قضیه pumping ← برای زبان‌های نوع دوم به طور عام
- " " " " " " خطی (Linear CFL)

- خصوصیات بستاری
- مسائل قابل تصمیم‌گیری برای زبان‌های نوع دوم

قرارداد: زبان‌های نوع دوم خطی: (LCFL)

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

گرامر: گرامر‌های نوع دوم خطی:

یک گرامر نوع دوم خطی است اگر در سمت راست هر قاعده حداکثر یک متغیر وجود شده باشد

مثال) گرامر نوع دوم خطی ← $S \to ab \mid aSb$ ← $L = \{a^n b^n \mid n \geq 1\}$

یک متغیر — صفر تا متغیر

تمرین: یک زبان نوع دوم غیرخطی است اگر نتوان برای آن یک گرامر نوع دوم خطی بدست آورد.

یک زبان نوع دوم غیرخطی ← $L = \{w \mid w \in \{a,b\}^* , \#a's = \#b's\}$

$S \to aSbS \mid bSaS \mid \lambda$

$S \to aSb \mid bSa \mid SS \mid \lambda$

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

جلسه بیست و دوم

خصوصیات زبان‌های نوع دوم

- زبان‌های نوع دوم خطی ← گرامر نوع دوم خطی بتوان برای آن نوشت

$L = \{a^n b^n \mid n \geq 1\}$ $\quad S \to aSb \mid ab$

- زبان‌های نوع دوم غیرخطی ← گرامر نوع دوم خطی نتوان برای آن نوشت

$L = \{w \mid w \in \{a,b\}^*, \#a's = \#b's\}$ $\quad S \to aSb \mid bSa \mid SS \mid \lambda$

قضیه‌ی pumping برای زبان‌های نوع دوم

۱) قضیه pumping برای زبان‌های نوع دوم بطور عام

فرض کنید L یک زبان نوع دوم نامتناهی است. در این صورت یک عدد مثبت صحیح (m) وجود دارد بطوریکه برای هر $w \in L$ ، $|w| \geq m$ ، این رشته را بتوان بصورت $uvxyz$ تجزیه کرد.

(شرایط تجزیه $|vy| \geq 1$ ، $|vxy| \leq m$)

بطوریکه $uv^i xy^i z$ برای تمام i ها برقرار باشد ... و ۱، ۲، ...

← از این قضیه می‌توانیم استفاده کنیم که نشان دهیم یک زبان نوع دوم نیست

مثال) نشان دهید که $L = \{a^n b^n c^n \mid n \geq 0\}$ یک زبان نوع دوم نیست

در اینجا یک سن و ترتیب در نظر می‌گیریم

رشته‌ای از عضو زبان $\quad m^2$

رشته برای شکستن به ترتیب داده می‌شود

حالا باید نشان دهیم به ازای هر شکستن ترتیب من ناحیه‌ای که حرکت برنده است

من $\quad$ ترتیب

$a^m b^m c^m \quad m$

$\to aaa\ldots abbb\ldots bccc\ldots c$

شرایط تجزیه داریم $|vy| \geq 1$ ← در نتیجه قسمتی از v و y می‌تواند λ شوند. روی z و u نیز شرطی نداریم

یک پنجره vxy داریم که طول آن کمتر از m باید باشد که می‌تواند هر جای رشته قرار گیرد.

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

فرض می‌کنیم که این پنجره از a انتخاب شود در این صورت داریم:

$$\left.\begin{array}{l} v=a^{k'} \\ y=a^{k''} \end{array}\right\} \rightarrow \underbrace{\underset{u}{a^{k}}\,\underset{v}{a^{k'}}\,\underset{x}{a^{k''}}\,\underset{y}{a^{k'''}}}\;\underset{z}{b^{m}c^{m}}$$

اگر $i=0$ قرار دهیم که در اصل یک a می‌ریزد

حالا اگر پنجره‌ای تعدادی a و b قرار گیرد در این صورت تعدادی a و تعدادی b می‌ریزد بنابراین انتخاب c بهم می‌خورد برای حالت‌های دیگر نیز این مشکل وجود دارد.

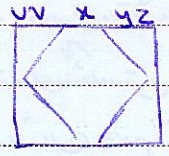
تنها امید این است که پنجره طوری قرار گیرد که هم a و هم b و هم c در آن وجود داشته باشد که این هم امکان‌پذیر نیست زیرا طول vxy باید کمتر از m باشد و طول هر سری a و هر سری b و هر سری c، m است.

2) قضیه pumping برای زبان‌های نوع دوم خطی.

فرض کنید L یک زبان نوع دوم خطی نامتناهی است در این صورت یک عدد ثابت صحیح m وجود دارد بطوریکه برای هر $w\in L$ و $|w|\geq m$، این رشته را می‌توان به صورت uvxyz تجزیه کرد در شرایط تجزیه $|vxy|\geq 1$ و $|vyz|\leq m$ به طوریکه برای تمام i ها $uv^{i}xy^{i}z\in L$ $i=0,1,2,\dots$

← برای اینکه نشان دهیم یک زبان نوع دوم خطی نیست

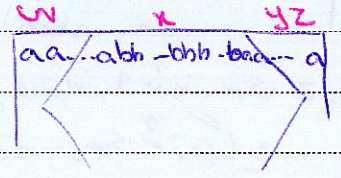
مثال) نشان دهید که $L=\{w \mid w\in\{a,b\}^{*},\ \#a's=\#b's\}$ نوع دوم خطی نیست



منظور از شرط $|uvyz|\leq m$ رشته‌ای است که به شکل زیر هستند یعنی با کوچک و بزرگ کردن x نمی‌خواهیم که طول uv+yz کوچکتر یا بزرگتر شود

به هم از بزرگی m در قیب استفاده می‌کنیم

رشته	قیب
$aa^{m}b^{m}b^{m}a^{m}$	m



دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

خصوصیات بستاری زبان های نوع دوم

قضیه ۱ ← خانواده زبان های نوع دوم تحت $\cup$ و $\cdot$ و $*$ بسته می باشد

یعنی دو زبان نوع دوم به ازای اعمال بالا زبان دوم خواهند بود.

اگر L_1 و L_2 نوع دوم باشند در این صورت $L_1 \cup L_2$ ، $L_1 L_2$ ، L_1^* هم نوع دوم هستند

اثبات: چون L_1 نوع دوم است می توان یک گرامر نوع دوم برای آن نوشت

L_1: $S_1 \to \dots$

چون L_2 " " "

Subject:
Year. Month. Date. ( )

قضیه 3) خانواده های زبان های نوع دوم نسبت به متمم بسته نیست

اثبات) از طریق فرض خلف و قضیه دمورگان

$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$ ← قضیه دمورگان

← فرض می کنیم که نسبت به متمم بسته باشند

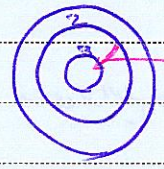
فرض کنیم $\overline{L_1}$ ← نوع دوم
" " $\overline{L_2}$ ← " " "

$\overline{L_2}$ و $\overline{L_1}$ ← نوع دوم ⇐ $\overline{\overline{L_1} \cup \overline{L_2}}$ ⇐ نوع دوم

درصورتی که $L_1 \cap L_2$ نوع دوم نیست ※ ⇐ قضیه ثابت می شود.

قضیه 4) خانواده زبان های نوع دوم تحت اشتراک متمم بسته است.

← اشتراک یک زبان نوع دوم با یک زبان نوع سوم از نوع دوم است.



اشتراک 2 و 3 در 2 می افتد

اثبات در صفحه 223 کتاب

مثال) نشان دهید که زبان زیر از نوع دوم است

$L = \{a^n b^n \mid n \geq 0 , n \neq 100\}$

$L_1 = \{a^n b^n \mid n \geq 0\} - \{a^{100} b^{100}\} \Rightarrow L_1 = L$

قضیه 5) خانواده زبان های نوع دوم تحت تفریق متمم بسته است

اگر L نوع دوم و R نوع سوم باشد در این صورت L-R نوع دوم است

$L - R = L \cap \overline{R}$

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

R متمم ← $\bar{R}$ ← متمم L نوع دوم

⇐ تحت قضیه 4 ← نوع دوم است ⇐ $\bar{L}$ نیز نوع دوم است

ادامه مثال قبل)

$\{a^{100}b^{100}\}$ → زبان نوع سوم متمم است

$\{a^nb^n \mid n\geq 0\}$ ⇒ زبان نوع دوم است

طبق قضیه 5 داریم $\{a^nb^n \mid n\geq 0\} - \{a^{100}b^{100}\}$ نوع دوم خواهد بود

حل کتاب برای این مسئله:

$L_1 = \{a^{100}b^{100}\} \rightarrow L = \{a^nb^n \mid n\geq 0\} \cap \bar{L_1}$

که قضیه ی 4 قابل اثبات است

L_1 ← نوع سوم ← $\bar{L_1}$ ← نوع سوم

$\{a^nb^n \mid n\geq 0\}$ ← نوع دوم است

⇐ $\{a^nb^n \mid n\geq 0\} \cap \bar{L_1}$ نوع دوم است

مثال) نشان دهید که زبان زیر نوع دوم نیست

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

- می توان از اثبات برای $a^nb^nc^n$ استفاده کرد

- از یافته های این جلسه

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

مجموعه ی L مجموعه ای است که داریم حالا با اضافه کردن حالتی که تعداد a و b و c به صورت مساوی نیست

سرهم می آیند می توانیم به زبان $\{a^nb^nc^n\}$ که نوع دوم نیست برسیم

$L(a^*b^*c^*)$ ← نوع سوم است

اگر L نوع دوم باشد ⇐ $L \cap L(a^*b^*c^*)$ نوع دوم خواهد بود

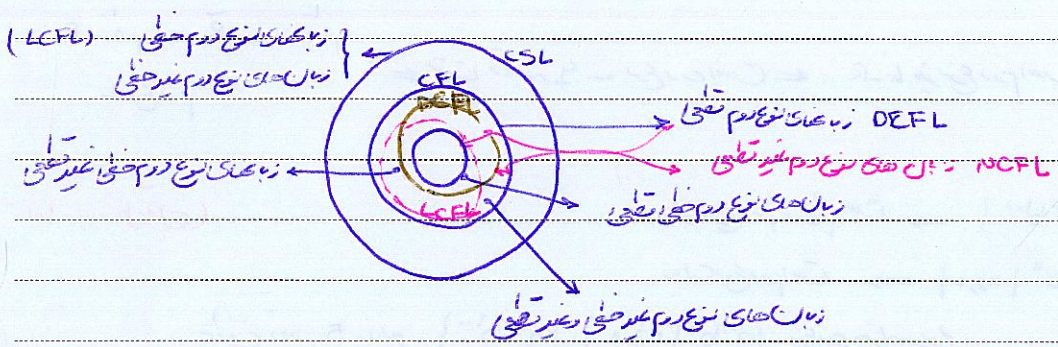
در این صورت $a^nb^nc^n$ نیز باید نوع دوم باشد در صورتی که اینطور نیست ✗

پس قضیه اثبات می شود.

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

هرم زبانها



زبان‌های نوع سوم به طور کامل در DCFL قرار دارند زیرا به طور کامل قطعی و منظم هستند

R کاملاً در LCFL قرار می‌گیرد زیرا هر زبان نوع سوم خطی است

چون گرامرهای نوع سوم همگی خطی هستند

LCFL و DCFL اشتراکی دارند که R در آن قرار دارد

* زبان‌های غیرقطعی

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

وقتی α را می‌بینیم دو حالت وجود دارد یا a^n یا a^{2n} پس در پشته قرار دهیم

غیرقطعی است

وقتی می‌توان برای آن گرامری نوشت که سمت راست آن تعداد یک متغیر وجود دارد

$S \to S_1 | S_2$
$S_1 \to aS_1b | \lambda$
$S_2 \to aS_2bb | \lambda$

* خطی قطعی

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

* زبان‌های غیرقطعی و غیرخطی

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

* زبان‌های قطعی غیرخطی

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

* زبان‌های نوع دوم نیستند

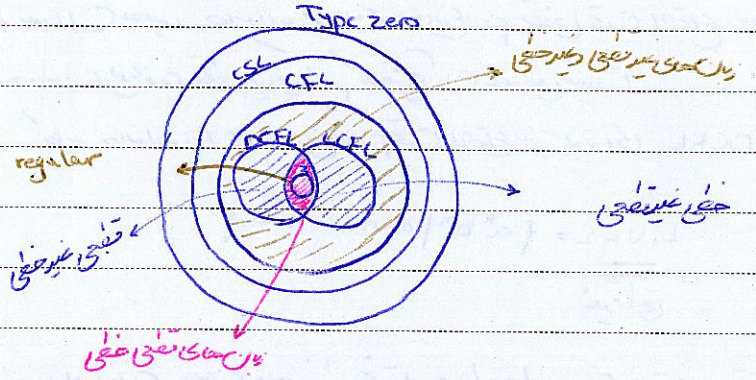
$L = \{a^n b^n c^n \mid n \geq 0\}$

دانشگاه مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

جلسه بیست و سوم 87/9/23

هم زبان‌ها:



← نشان دهید که خانواده زبان‌های نوع دوم تحت همومورفیزم بسته است. (برای یک زبان نوع دوم می‌توان گرامر نوع دوم نوشت)

اگر L زبان نوع دوم باشد h(L) هم نوع دوم است.

$S \to abAB$ $\quad h(a)=00$

$A \to aA \mid b$ $\quad h(b)=1$

$B \to bB \mid a$

اگر به جای a و b، h(a) و h(b) قرار دهیم h(L) خواهد شد. با این تعریف هم نوع زبان تغییر نمی‌کند

یک گرامر خطی می‌توانیم بنویسیم

← خانواده زبان‌های نوع دوم خطی تحت همومورفیزم بسته است.

$L = \{a^n b^n \mid n \geq 1\}$ ← نوع دوم خطی $\quad S \to aAb \mid ab$

$h(L) = \{(00)^n 1^n \mid n \geq 1\}$ $\quad S \to 00A1 \mid 001$

↳ این تغییر خطی بودن را تغییر می‌دهد و این گرامر هم خطی است

← نشان دهید خانواده زبان‌های نوع دوم تحت عملگر رفلکس بسته است

نوع دوم w $\Longrightarrow$ نوع دوم w^R

سمت راست را رفلکس می‌کنیم ← برای اینکه به راست می‌آید

نوع دوم است و تغییری که در

ساختارش رخ می‌دهد

دانشکده مهندسی کامپیوتر دانشگاه صنعتی امیرکبیر
اطلاعات و محاسبات

Subject:
Year. Month. Date. ( )

← نشان دهید خانواده زبان های نوع دوم قطعی تحت اجتماع بسته نیست.

دو زبان نوع دوم قطعی پیدا می کنیم که اجتماع آن ها قطعی نیست

کل خانواده زبان های نوع دوم تحت اجتماع بسته بوده ولی DCFL این خصیصه را ندارد.

$L_1 = \{a^n b^{2n} \mid n \geq 0\}$

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

$\underbrace{L_1 \cup L_2}_{\text{غیرقطعی}} = \{a^n b^m \mid n = m \text{ or } n = 2m\}$

← خانواده زبان های نوع دوم قطعی تحت Concatnation بسته نیست

کل خانواده تحت concatnation بسته است ولی concatnation دو قطعی الزاماً قطعی نخواهد بود.

$L_1 = \{a^n b^n \mid n \geq 0\}$

$S_1 \to aS_1b \mid \lambda$

$L = \underbrace{L_1 . L_1}_{\text{دو زبان قطعی}} = \{a^n b^n a^m b^m \mid n, m \geq 0\} \Rightarrow S \to S_1 S_1$ ← غیرقطعی

$S_1 \to aS_1b \mid \lambda$

ساده تر بخواهیم داریم چون یک تعیین برای n داریم تعداد a و b ها مساوی باشد

" " " m " " a و b در سمت دوم مساوی باشد

مسایل قابل تصمیم گیری:

① این مسئله که یک زبان نوع دوم تهی می باشد یا نه قابل تصمیم گیری است

② " " " " " " " ناتهی می باشد یا نه قابل تصمیم گیری است

یکی از مسایل غیرقابل تصمیم گیری برای زبان های نوع دوم

تساوی دو زبان نوع دوم غیرقابل تصمیم گیری است

undecidable

الگوریتمی که بتوانیم بر اساس تساوی زبان های نوع دوم را بگوییم وجود ندارد و نخواهد بود

برای اینکه نشان دهیم قابل تصمیم گیری است باید الگوریتم را پیدا کنیم

دانشکده مهندسی کامپیوتر و فناوری
اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

فرض می‌کنیم زبان را با گرامر به ما دادند

① چگونه می‌توانیم چک کنیم تهی است (زبانی تهی است که از S چیزی مشتق نشود)

یعنی هیچ رشته‌ای را تولید نمی‌کند

ممکن است اگر رشته‌هایی که در ساده‌سازی گفتیم ممکن است این جا به درد بخورد

← متغیرهایی که از جنبه اول مفید هستند را پیدا می‌کنیم ← مجموعه تمام متغیرهایی که از جنبه اول مفید است در این مجموعه

اگر S باشد تهی نیست یعنی می‌توانیم یک رشته غیرتهی بسازیم

② آیا زبان تعریف شده با گرامر غیرتهی است یا نه و آیا می‌توانیم بی‌نهایت رشته تولید کنیم یا نه؟

متغیرهای تکرار شونده looping variable

متغیر X تکرار شونده است اگر

$X \xrightarrow{*} \alpha X \beta$ ← حد واسطی

X حد واسطی دوباره ظاهر شود

چگونه متغیرهای تکرار شونده را پیدا کنیم؟

گراف وابستگی:

اسم متغیرها اسم node ها هستند (نسبت‌ها بین متغیرهای بعد از اسلش می‌آیند)

$S \to aABb$

$B \to aC \mid b$

$C \to bA \mid b$

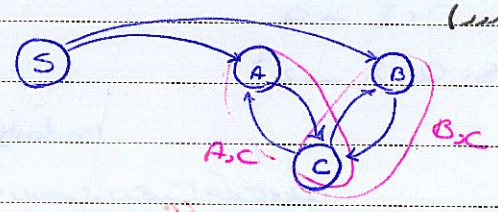
$A \to aC \mid b$

$C \to aB \mid b$

$A \to a$

$B \to b$

$C \to b$



اگر در این گراف یک سیکل باشد حداقل یک متغیر تکرار شونده خواهیم داشت

فرض کنید کلیه متغیرها دیگر امر داده شده مفید باشد (از هر دو جنبه)

بخش ②: اگر گرامر داده شده حداقل دارای یک متغیر تکرار شونده باشد زبان نامتناهی است در غیر این صورت زبان متناهی است چرا؟ فرض کردیم تمام متغیرها دیگر امر داده شده مفیدند

فرض ها ① $A \xrightarrow{*} \alpha A \beta$

② $S \xrightarrow{*} \alpha' A \beta'$

③ $A \xrightarrow{*} w$

$\Rightarrow$ بی‌نهایت رشته تولید می‌کند

دانشکده مهندسی کامپیوتر و فناوری
انجمن علمی دانشجویان دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

$S \overset{*}{\Rightarrow} \alpha' A \beta'$

$\overset{*}{\Rightarrow} \alpha' \alpha'' A \beta'' \beta'$

$\Rightarrow \alpha' \alpha'' w \beta'' \beta'$ ← فرم‌های

A نشان داده شده بجای قاعده ① برای جایگزینی استفاده کردیم

بنابر مقدارهای متفاوت A و این کار عملیات ادامه

می‌تواند ادامه کند

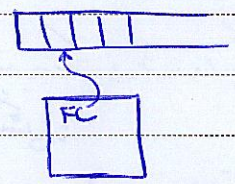
تئوری زبان‌های نوع صفر نوع یک ← CSL

فصول 9 و 10 و 11 و 12

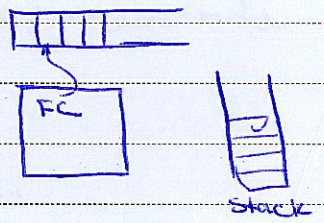
فصل یکم:

- پذیرنده برای زبان‌های نوع صفر (Turing Machine)

FA:



PDA:



Turing Machine:

$\delta: Q \times \Sigma \to Q$

$\delta: Q \times \Sigma \to 2^Q$

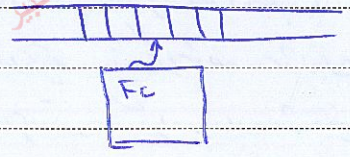
Tape حالا این دستگاه برای یک خاصیت بود readonly

در Turing Machine، Tape نامتناهی است و از هر دو جهت حرکت می‌کند

و هم می‌توانیم بنویسیم و هم بخوانیم

- infinite type
- Move in both directions
- Read / write type

انشکده مهندسی کامپیوتر و فناوری
اطلاعات دانشگاه صنعتی امیر کبیر



یکی می‌خواند بسته به آن چه خوانده ممکن است چیزی بنویسد و بسته به حالت فعلی و چیزی که خوانده ممکن است به چپ و راست حرکت کند

Subject:
Year. Month. Date. ( )

تعریف ماشین Turing machine

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

Q: حالت‌های ماشین

Σ: الفبا $\Sigma \in T$

T: Tape alphabet علائمی که روی Tape قرار می‌گیرد

q_0: حالت اولیه

$\square$: blank Symbol اگر در خانه‌ای چیزی نباشد این علامت در آن است ، $\square \in T$

F: مجموعه حالت‌های نهایی

δ: تابع انتقال

$\delta: Q \times T \rightarrow Q \times T \times (L, R)$

حالت را تعیین می‌کند

L: در خانه‌ای قرار دارد علامت چپی‌ای را بنویسید

حرکت به سمت چپ و یا راست

در اصل head ثابت است و Tape حرکت می‌کند

مثال:

$aqbbba \vdash axqbba \vdash aqxaba$

a	b	b	b	a

q

وقتی حالت q در شرف خواندن b

حالت جدید

$\delta(q, b) = \delta(q', x, R)$

حرکت به راست

نوشتن بجای b

a	x	b	b	a

q'

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

a	x	a	b	a

q''

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی شریف

اولین علامت رشته

آخرین علامت رشته

a_1	a_2	...	a_i	...	a_n

q

q را قبل از علامتی که ماشین اشاره می‌کند قرار می‌دهیم

$a_1 a_2 \ldots q\, a_i \ldots a_n \vdash$

$a_1 a_2 \ldots x q\, a_{i+1} \ldots a_n$

if $\delta(q, a_i) = (q', x, R)$

Subject:
Year. Month. Date. ( )

چه موقع پذیرش اتفاق می‌افتد؟

پذیرش: یک رشته پذیرفته می‌شود اگر ماشین وارد یکی از حالات نهایی خود شود پس متوقف گردد.

عدم پذیرش: یک رشته پذیرفته نمی‌شود " " " " " " " غیرنهایی " " " "

مثال) این ماشین چی کار می‌کند؟

$Q = \{q_0, q_1\}$

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

$T = \{a, b, \square\}$

$F = \{q_1\}$

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

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

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

$\square\square\square\ q_0 aabbb\square \vdash$ ①

$bq_0 abb \vdash$ ①

$bbq_0 bb \vdash$ ②

$bbbq_0 b \vdash$ ②

$bbbbq_0 \square \vdash$ ③

$bbbq_1 b \nvdash$ ← ماشین وارد q_1 می‌شود و متوقف می‌شود

رشته پذیرفته شد

این ماشین رشته‌ی ورودی خود را به صورت $b^*a^*b^*$ تبدیل می‌کند و از b های اول و وسط رشته متوقف می‌شود

دانشکده مهندسی کامپیوتر و فناوری
اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

جلسه بیست و پنجم 87/9/25

مثالی از تابع انتقال: این ماشین چه کاری می‌کند؟

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

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

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

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

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

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

$q_0 abba \vdash a q_1 bba$

$\vdash q_0 bba$

$\vdash a q_1 bba$

$\vdash q_0 abba$

$\vdots$

هر شش روی Tape روی علامت اول و دوم رفت و برگشت خط می‌کرد بدون رفتن متوقف نمی‌شود و اعلام نتیجه نمی‌کند

$\Rightarrow$ never halts (terminate)

* در این مثال علامت نهایی مهم نبود چون اصلاً نمی‌ایستد که بخواهد در حالت نهایی باشد

بیان یک زبان Turing machine

$L(TM) = \{ w \mid q_0 w \vdash^* x_1 q_f x_2 \text{ for } x_1, x_2 \in T^* \text{ and } q_f \in F \}$

مثال) یک ماشین تورینگ برای زبان زیر طراحی کنید

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

یک زبان نوع دوم که می‌توان برایش PDA طراحی کرد ولی برای طراحی نشدن مسئله برای این زبان TM طراحی می‌کنیم

اول a را تیک می‌کنیم و جلو می‌رویم تا اولین b را تیک کنیم و بعد برمی‌گردیم تا اولین a تیک نخورده از سمت چپ را پیدا کنیم و تیک می‌کنیم و جلو می‌رویم تا به اولین b تیک نخورده برسیم و بعد تیک می‌کنیم

① ③ ⑤ ② ④ ⑥
aaaabbbb

Subject:

Year. Month. Date. ( )

$Q = \{q_0, q_1, q_2, q_3, q_4\}$ $\Sigma = \{a, b\}$ تیک کردن ها تیک کردن ها

$F = \{q_2\}$ $T = \{a, b, x, y, \square\}$

① $\delta(q_0, a) = (q_1, x, R)$

② $\delta(q_1, a) = (q_1, a, R)$

③ $\delta(q_1, y) = (q_1, y, R)$

④ $\delta(q_1, b) = (q_2, y, L)$

⑤ $\delta(q_2, y) = (q_2, y, L)$

⑥ $\delta(q_2, a) = (q_2, a, L)$

⑦ $\delta(q_2, x) = (q_0, x, R)$

بخش اشتراکی

⑧ $\delta(q_0, y) = (q_3, y, R)$

⑨ $\delta(q_3, y) = (q_3, y, R)$

⑩ $\delta(q_3, \square) = (q_4, \square, R)$

$q_0 aaaabbbb \vdash^{①} x q_1 aaabbbb$ ~ $q_0 aaaabbbb$

$\vdash^{②} xa q_1 aabbbb$ عبور از a های تیک نخورده

$\vdash^{②} xaa q_1 abbbb$

$\vdash^{②} xaaa q_1 bbbb$

$\vdash^{④} xaaa q_2 a ybbb$ برای تیک کردن b

$\vdash^{⑥} xa q_2 aa ybbb$ حرکت به سمت چپ عبور از y ها

$\vdash\ x q_2 aaa ybbb$

$\vdash\ x q_0 aaa ybbb$

$x x q_1 aa ybbb$

$x x a q_1 a ybbb$

$\vdash^{③} x x aa q_1 ybbb$ باید برای پیدا کردن b های تیک نخورده از روی تیک خورده ها رد شود

$\vdash\ x x aa y q_1 bbb$

$\vdash\ x x aa q_2 yybb$

$\vdash\ x x a q_2 ayybb$

$\vdash\ x x q_2 aa yybb$

$\vdash\ x q_2 x aa yybb$

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

$xxq_0aayybb \vdash xxxq_1ayybb \vdash xxxaq_1yybb \vdash xxxayq_1ybb$
$\vdash xxxayyq_1bb \vdash xxxayyq_2yyb \vdash xxxaq_2yyyb \vdash xxxq_2ayyyb$
$\vdash xxxaq_2yyyb \vdash xxq_2xayyyb \vdash xxxq_0ayyyb \vdash xxxxq_1yyyb$
$\vdash xxxxyq_1yyb \vdash xxxxyyq_1yb \vdash xxxxyyyq_1b\square \vdash xxxxyyyq_2yy$
$\vdash xxxxyq_2yyy \vdash xxxxq_2yyyy \vdash xxxq_2xyyyy \vdash xxxxq_0yyyy$
$\vdash xxxxyq_3yyyy \vdash xxxxyyq_3yy \vdash xxxxyyyq_3y \vdash xxxxyyyyq_3\square$
$\vdash xxxxyyyy\square q_4\square \rightarrow$ پذیرش

$q_0abb \vdash xq_0bb \vdash q_2xbb \vdash xq_0yb \vdash xyq_3b$ ✗ می‌ایستد در حالت غیرمجاز است
$q_0abab \vdash xq_1bab \vdash q_2xyab \vdash xq_0yab \vdash xyq_3ab$ ✗ ... ...

مثال: یک TM برای زبان زیر طراحی کنید.

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

از روی ماشین طراحی شده ممکن است بتوانیم ماشینی طراحی کنیم که زبان زیر را بپذیرد. می‌دانیم حالت نهایی را تغییر دهیم یعنی باید بدانیم در چه حالت‌هایی متوقف می‌شود مثلاً باید بدانیم ماشین در چه صورت در q_1 متوقف می‌شود که به نظر می‌آید در صورتی که تعداد a ها بیشتر از b ها باشد در q_1 متوقف می‌شود مگر q_1 حالت نهایی باشد

$L=\{a^nb^n \mid n>m\}$ زبان زیر را می‌پذیرد

مسئله 5.245) این TM چه زبانی را می‌پذیرد

$M=(\{q_0,q_1,q_2,q_3\},\{a,b\},\{a,b,\square\},\delta,q_0,\square,\{q_3\})$

① $\delta(q_0,a)=(q_1,a,R)$
② $\delta(q_0,b)=(q_2,b,R)$
③ $\delta(q_1,b)=(q_1,b,R)$
④ $\delta(q_1,\square)=(q_3,\square,R)$
⑤ $\delta(q_2,b)=(q_2,b,R)$
⑥ $\delta(q_2,a)=(q_3,a,R)$

شرط هم می‌تواند b و هم a خروجی شود ← زبان اجتماع دو زبان است
وارد q_1 می‌شود
وارد q_2 می‌شود

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

اگر رشته با a شروع شود ← هر تعداد b بیاید به سمت راست حرکت می‌کند (3) تا به علامت □ برسد (6)

$\{ab^n \mid n \geq 0\}$ ← پذیرش ← q_3 می‌رود و متوقف می‌شود ← انتهای رشته همراه

اگر رشته با b شروع شود ← تا آخر b می‌خواند (5) بعد یک a می‌خواند

$\{bb^n a \mid n \geq 0\}$ ← در این صورت می‌گوییم با این رشته باید تمام شود

الزامی ندارد که توقف با رسیدن به انتهای رشته باشد a می‌خواند و به حالت نهایی می‌رود و مهم نیست که رشته تمام شده باشد یعنی این متناظر یک prefix از رشته است و می‌تواند ادامه داشته باشد و در اصل زبانی که می‌پذیرد به صورت زیر است $bb^*a(a+b)^*$ در این جا اعلام پذیرش می‌کند

مثال) یک TM طراحی کنید $\{w \mid |w| \text{ is even}\}$

برای هر مسئله قابل حل می‌توانیم یک Turing Machine طراحی کنیم یعنی با استفاده از خواندن، نوشتن، چپ یا راست رفتن می‌توانیم راه حل هر مسئله را بیان کنیم

تعریف تابع قابل محاسبه (Turing) computable function ← تمرین عملی

تابع f با دامنه D قابل محاسبه است اگر یک ماشین Turing

$M = (Q, \Sigma, T, \delta, q_0, \square, F)$ ← با استفاده از عملیات پایه مقدار تابع را محاسبه کند

وجود داشته باشد به طوری که $q_0 w \vdash^* q_f f(w)$ برای تمام $w \in D$ ← عمل نوشتن، چپ یا راست رفتن

ماشین وارد q_f شود و $f(w)$ یا ارزش w ورودی Tape باقی بگذارد

تز تورینگ Turing Thesis ← حضور ایشان نشده

* - هر محاسبه‌ای که توسط قلم و کاغذ بوسیله بشر قابل انجام باشد توسط ماشین Turing قابل انجام است.

هر چیزی که ایجادی رحیم می‌تواند توسط عملیات پایه بیان شود

- تا به حال مسئله‌ای پیدا نشده است که قابل حل باشد ولی توسط ماشین Turing حل نشود

- ماشین Turing معادل قوی‌ترین کامپیوتر حال حاضر می‌باشد ← هر مسئله‌ای که آن حل کند TM هم می‌تواند حل شود

← هر چه مسائل بیشتری را حل کند قوی‌تر است

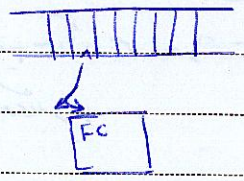
دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

جلسه بیست و پنج 87/9/30

فصل دهم: مدل‌های غیراستاندارد ماشین Turing

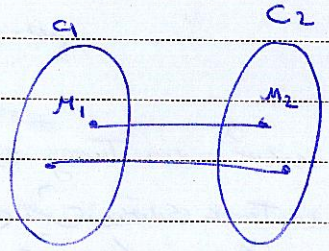
مدل استاندارد ماشین Turing: همان مدلی که قبلاً خواندیم



$\delta: Q \times T \to Q \times T \times \{L, R\}$

تعداد مدل‌های غیراستاندارد خیلی زیاد است.

تمرین: دو خانواده C_1, C_2 از ماشین‌ها معادل هستند اگر برای هر ماشین در C_1 یک ماشین معادل در C_2 وجود داشته باشد و برای هر ماشین در C_2 یک ماشین معادل در C_1 وجود داشته باشد.



M_1 معادل M_2 است اگر هر جمله‌ای که M_1 می‌تواند حل کند M_2 هم حل کند و برعکس.

← NDFA معادل DFA است.

← NPDA ، DPDA نیست. ماشین‌هایی در خانواده NPDA هستند که معادل هیچ DPDA نداریم.

Turing Machine with Stay on Option: وجود stay on توانایی ماشین را زیاد نمی‌کند.

$\delta: Q \times T \to Q \times T \times \{L, R, S\}$

$\delta(q, a) = (q', b, S)$ ← head برچسب یا راست حرکت می‌کند روی همان cell که هست

همان جایی می‌ماند در ماشین اصلی حتماً باید چپ یا راست حرکت می‌کردیم

این خانواده از ماشین‌ها معادل خانواده ماشین‌های Turing است استاندارد است.

نشان دهیم هر کدام از ماشین‌ها می‌توانند ماشین‌های خانواده دیگر را stimulate کنند

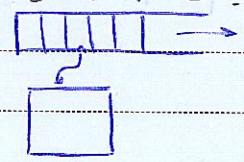
دانشکده مهندسی کامپیوتر و فناوری
اطلاعات دانشگاه صنعتی امیرکبیر

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

Semi-infinite Tape

این نوع Tape به واقعیت نزدیکتر است



ماشین Turing با نوار نیمه متناهی

نوار از یک طرف بسته و از یک طرف باز است.

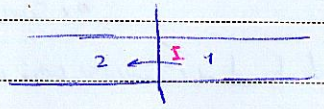
می‌تواند به چپ و راست حرکت کند ولی برای حرکت به سمت چپ محدودیت دارد

← این خانواده معادل خانواده ماشین‌های Turing استاندارد می‌باشد.

اثبات: ① هر نوار استاندارد این ماشین استاندارد هم می‌تواند ← می‌توانیم در استاندارد اجازه ندهیم که از یک خانه بیشتر به چپ حرکت کند

② با هر نوار ماشین استاندارد می‌توان انجام داد آیا این هم می‌تواند؟

باید نشان دهیم می‌توان یک Tape متناهی با یک Tape متناهی شبیه‌سازی کرد.



از یک جای تا می‌کنیم

وقتی خواست از ناحیه ۱ به ۲ برود از بالا به پایین می‌رود

از کجا می‌فهمد روی کدام است ← در 1 روی حالات q خواهد بود

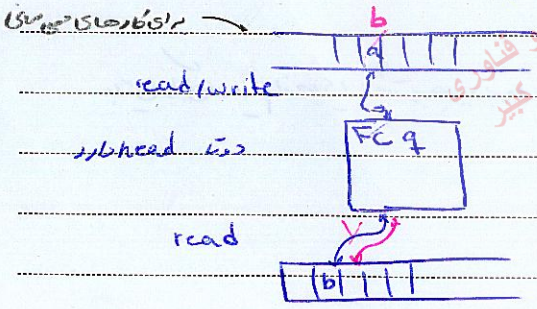
در 2 " " p خواهد بود

می‌فهمد کدام متعلق را باید تغییر دهد.

← گفتن این مقدمات برای تعریف پذیرنده برای زبان‌های نوع ۱ است.

The off-line Turing Machine

یک Tape اضافی دارد که مشابه ماشین‌های PDA است read only و حرکت در یک جهت



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

off-line روی Tape اصلی

زاحم کردن input ← مثلاً programming را راحت می‌کند.

انجمن علمی دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

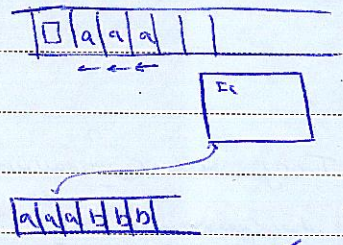
PAPCO

Subject:
Year. Month. Date. ( )

مثال: { a از دو طرف تبدیل به b می‌شود

همین کار را می‌توانیم ②

aaabbb



a را وارد FC می‌کند اگر همزمان Tape اصلی عوض شود و

off-line عمل کند (این روش هزینه‌اش کمتر است)

الگوریتم مجتب

یکی از بعدها FC حرکت کرده

①

②

□

آیا این مدل

① هر کاری استفاده از off-line استفاده نکند

② آیا هر کاری باید ببینیم می‌توانیم یک Tape متناهی در یک Tape

off-line وارد کنیم

یک Tape در شیار

با شیار ایجاد کنیم یک سمت حرکت می‌کند این را باید مدیریت کنیم

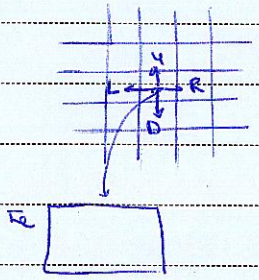
دانشگاه مهندسی کامپیوتر و اطلاعات دانشگاه صنعتی امیرکبیر

PAPCO

Subject:
Year. Month. Date. ( )

Turing Machine with Multidemensional Tape

یک tape چندبعدی



$\delta : Q \times T \to Q \times T \times \{L, R, U, D\}$

از ۹ طرف حرکت می‌کند

آیا مدل استاندارد است؟ درجه‌ی آزادی انیم سربیت کنیم

مثل (Column Major ، Row Major)

مثال) یک ماشین Turing را در نظر بگیرید که در آن در هر حرکت یا جهت‌ای نوار تغییر می‌کند و یا read/write head حرکت می‌کند و نه هر دو

یک تخلیه‌ی از این ماشین

آیا این خانواده‌ی ماشین‌ها مدل خانواده استاندارد هست یا نیست؟ یا مدل هشت‌تیک

Instruction استاندارد چندتا از Instruction های این سجیه کنیم

دانشکده مهندسی کامپیوتر و فناوری
اطلاعات دانشگاه صنعتی امیر کبیر

PAPCO

Subject:

Year. Month. Date. ( )

دانشکده مهندسی کامپیوتر و فناوری
اطلاعات دانشگاه صنعتی امیر کبیر

PAPCO

Subject:
Year. Month. Date. ( )

جلسه 87.10.2 ← حاضر چرا میشه؟ 8D

Non Standard Turing machine

- ماشین تورینگ با نوار چند شاره
- " " نیم متناهی
- off-line " " " "
- ⋮

این معادل ماشین Turing استاندارد می‌باشند

مثال) 266-6 ←

یک ماشین Turing non increasing Turing Machine برای آن می‌تواند یک سمت نیم‌خالی را به سمت non-blank جای خالی جایگزین کرد.

یعنی این نوع δ ها را می‌توانیم داشته باشیم ← $\delta(q,a)=(q',\square,R)$ ممنوعیت

در صورتی که $a=\square$ باشد این δ مجاز است

آیا این محدودیت باعث کاهش قدرت (توان) ماشین استاندارد می‌شود؟

در زبان‌های برنامه‌نویسی این چنین assignment را می‌توانیم داشته باشیم $A=\emptyset$

یک راه‌حل اینست که از متغیرهای دینامیکی استفاده کنیم یعنی در حین برنامه allocate کنیم متغیرهای دینامیکی هنگام allocate کردن مقداردهی صفر است یعنی خالی است و این متغیر جدید را می‌توانیم به جای A استفاده کنیم (در این صورت محتوی را به صورت مستقیم صفر می‌کنیم)

پس در ماشین Turing مشکل پیش می‌آید چون با قراری δ این را استثنا بگیریم برای حل مشکل نمایندگی را انجام می‌دهد

اگر مثلاً بخواهیم b را با □ جایگزین کنیم

شیار 1 را روی شیار 2 در خانه‌های آن خالی □ جستجو می‌کنیم جز b

1	a	b	a		
2	a	□	a		
3					

مسئله 7-266) این δ مجاز است اگر $\delta(q,a)=(q,b,L,R)$

$b\in T-\{\square\}$

آیا این محدودیت باعث توان ماشین می‌شود یا نه؟

انجمن کده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

مسئله 9 ـ 266) یک ماشین Turing با این محدودیت:

این δ ی زمانی است اگر a و b یکسان نباشند ← $\delta(q,a) = \delta(q',b,L \text{ or } R)$

آیا این محدودیت باعث کاهش توان ماشین می‌شود؟

اگر A متناهی باشد ⇐ A:=2 این غیرممکن است.

مثال) اگر در ماشین Turing به جای نوار نامتناهی که در هر دو جهت می‌تواند حرکت کند یک نوار نامتناهی داشته باشد که فقط می‌تواند در یک جهت (L یا R) حرکت کند آیا این باعث توان ماشین Turing می‌شود یا خیر؟ چرا؟

کاهش توان: نتواند یک برخی از زبان‌ها را پذیرش کند

← جواب: وقتی فقط طرفی روبرم نوشتن معنی ندارد چون می‌نویسیم که دوباره از آن استفاده کنیم

در اینکه هم که نامتناهی است بود باعث خود چون فقط به یک طرف می‌رود پس ماشین یک نوار

ترانشین کاهش پیدا می‌کند و تبدیل می‌شود به ماشین‌های متناهی نتوانستن به پذیرش زبان‌های نوع 3 کاهش

پیدا می‌کند

مثال) اضافه کردن یک پشته اضافی به PDA (یعنی PDA با دو تا پشته) آیا باعث می‌شود معادل ماشین Turing

بشود؟ در کل معادل چه ماشینی است.

مسئله 4 ـ 274

← اگر در PDA به جای پشته از صف استفاده کنیم معادل PDA است؟

← درایه‌ای اضافه داریم و از طوری می‌داریم

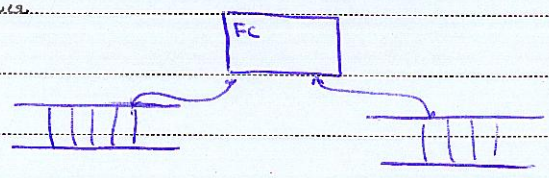
Multi tape TM: دو تا نوار نامتناهی و دو تا read/write head

(مثل اینکه در برنامه‌نویسی دو تا File داشته باشیم)

$\delta: Q \times T^2 \to Q \times T^2 \times \{L,R\}^2$

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

معادل ماشین Turing استاندارد است



باشگاه مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

مسئله 7 ، 277 : Counter automata

non-deterministic TM

ماشین در یک منبع بتواند چندین حرکت انجام دهد

$\delta : Q \times T \to 2^{Q \times T \times \{L,R\}}$ $\quad \rightsquigarrow \quad \delta(q_0, a) = \{(q_1, b, L), (q_2, a, R)\}$

آیا این TM معادل DTM می‌شود یا نه؟ آیا خانواده TM قطعی معادل خانواده TM های غیرقطعی هست؟ بله هر TM غیرقطعی دارای TM قطعی است.

عدم قطعیت را توسط روش انتقال می‌توانیم شبیه‌سازی کنیم (back Tracking) ← عدم قطعیت را در قالب یک برنامه قطعی اجرا می‌کند و برای همین الگوریتم قطعی می‌توانیم یک TM طراحی کنیم در نتیجه NTM معادل DTM است.

تا حالا به این نتایج رسیدیم:

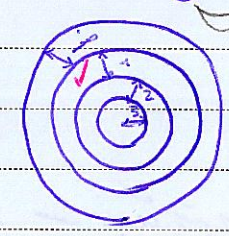
1 NDFA معادل DFA است

2 NPDA " PDA نیست

3 NTM " DTM می‌باشد

Universal TM section 10.4 , p.276

Linear Bounded Automata (LBA) ← پذیرنده برای زبان‌های نوع اول



برای هر خانواده‌ای می‌توانیم TM طراحی کنیم ولی لزومی ندارد همیشه می‌توانیم برای زبان 3 و 2 می‌توانیم ماشین‌های ساده‌تری طراحی کنیم

LBA : یک LBA یک ماشین Turing غیرقطعی است با این محدودیت که فضای مجاز استفاده روی نوار متناسب با طول رشته ورودی است که قرار است پذیرش یا عدم پذیرش آن اعلام شود.

$L = \{a^n b^n c^n \mid n \geq 1\}$ $\quad \rightsquigarrow \quad$ نوع دوم نیست یعنی نمی‌توان PDA برایش طراحی کرد یک نوع 1 است و می‌توان برایش LBA طراحی کرد

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر

Subject:
Year. Month. Date. ( )

در واقع یک TM است

فضایی که نوار دارد وابسته به طول رشته است و

در نتیجه LBA است (در این جا طول رشته برابر است)

مثال) ادعای کنیم که نوع اول است. با طراحی LBA

نشان می‌دهیم.

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

$n=4 \Rightarrow \underbrace{aaaa \ldots a}_{24}$

برای اعلام پذیرش یا عدم پذیرش باید ببینیم تعدادش فاکتوریل یک عدد است یا نه؟

$24/2 = 12/3 = 4/4 = 1$

این TM دارای یک نوار دو شیاره است.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
a	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
a	a	a	a																				

روی شیار اول رشته‌ای که قرار است به پذیرش بشود درا می‌گذاریم

روی شیار اول از هر دو تا a روی شیار بالا یکی را نگه می‌داریم

در شیار پایین یک a اضافه می‌کنیم و به ازای هر تعداد a ای که در شیار پایین است یک a را نگه می‌داریم و بعد از

این مرحله دوباره یک a به شیار پایین اضافه می‌کنیم

مثال)

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

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

TM در سینوس الگوریتم است.

Instruction ها خیلی ساده است و زیادی عمل نداریم

دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر