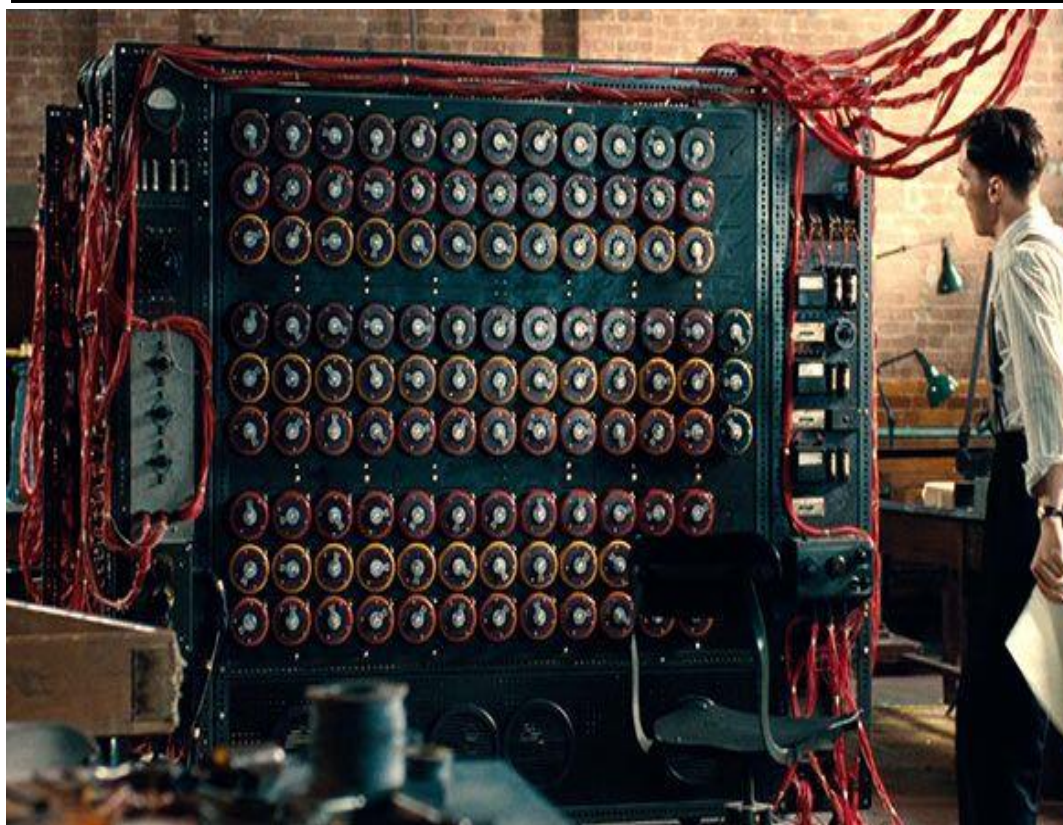


نظریه زبان‌ها و ماشین‌ها



ابوالفضل گیلک

مفاهیم پایه

جلسه اول: نظریه مجموعه ها، کاردینالیتی.

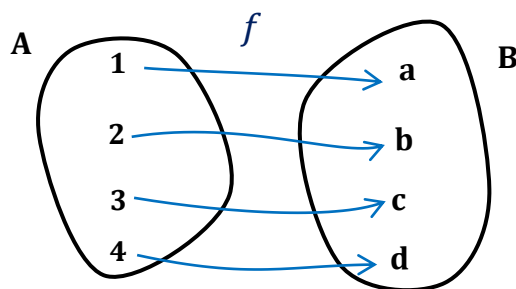


رده بندی مجموعه ها

می خواهیم مجموعه ها را با توجه به اندازه آنها رده بندی کنیم. در مجموعه های متناهی، می توانیم از "تعداد اعضا" به عنوان معیاری برای اندازه مجموعه استفاده کنیم. اما در حالت کلی به جای تعداد اعضا از مفهومی به نام (کاردینالیتی) استفاده می کنیم. ابتدا به رابطه هم توانی بین مجموعه ها توجه کنید:

رابطه هم توانی

می گوییم مجموعه های A و B هم توان (هم ارز - هم تعداد) هستند اگر و تنها اگر تابعی دوسویی (یک به یک و پوشا) از یکی به دیگری موجود باشد. چنین تابعی را یک تناظر یک به یک می نامیم.



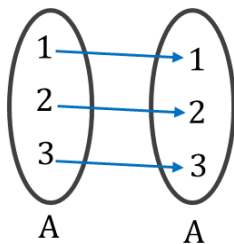
$$A \simeq B \quad \Leftrightarrow \quad \exists f: A \rightarrow B \text{ یک به یک و پوشا}$$

رابطه هم توانی یک رابطه هم ارزی است. زیرا سه ویژگی زیر را دارد:

$\forall A \quad (A \simeq A)$	بازتابی است:
$\forall A, B \quad (A \simeq B \rightarrow B \simeq A)$	متقارن است:
$\forall A, B, C \quad (A \simeq B \wedge B \simeq C \rightarrow A \simeq C)$	متعدی است:

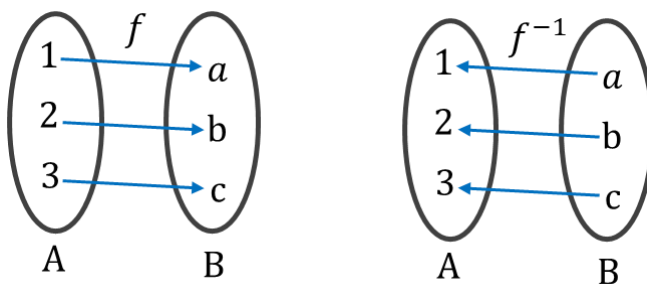
بازتابی بودن:

روی مجموعه دلخواه A تابع همانی با ضابطه $I(x) = x$ نشان می دهد: $A \simeq A$



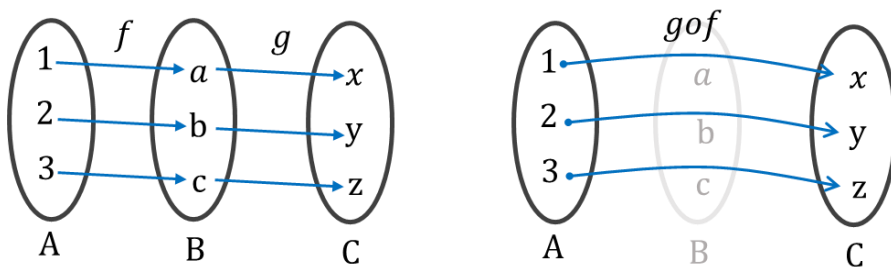
متقارن بودن:

اگر $f: A \rightarrow B$ یک به یک پوشا باشد تابع معکوس آن، از B به A یک به یک و پوشاست.



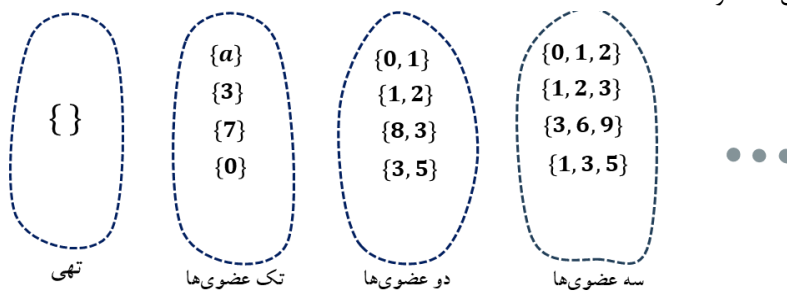
متعدی بودن:

اگر $f: A \rightarrow B$ و $g: B \rightarrow C$ یک به یک پوشا باشند ترکیب آنها $g \circ f$ از A به C یک به یک و پوشاست.



هر رابطه هم‌ارزی باعث افراز و دسته‌بندی اشیاء در کلاس‌های هم‌ارزی می‌شود. رابطه هم‌توانی نیز، مجموعه‌ها را در کلاس‌های مجزایی رده‌بندی می‌کند. مثلاً تمام مجموعه‌های سه‌عضوی در یک کلاس قرار دارند

چون با هم هم‌توان هستند. مجموعه‌های دو عضوی همه با هم در یک کلاس قرار دارند، مجموعه تهی فقط با خودش هم‌کلاس است و



حالا به هر کدام از این کلاس‌ها یک کاردینال نسبت می‌دهیم.

کاردینال (عدد اصلی):

کاردینال هر مجموعه متناهی، همان تعداد اعضایش است.

مجموعه‌های متناهی	کاردینال
$\emptyset = \{ \}$	0
$\{1\}$	1
$\{1, 2\}$	2
...	...
$\{1, 2, \dots, k\}$	K

$$A = \{1, 2, \dots, k\} \Rightarrow \text{card}(A) = k$$

توجه: در نظریه زبان‌ها، مجموعه الفبا که آن را با Σ نشان می‌دهیم یک مجموعه متناهی است.

توجه: تساوی $\text{Card}(A) = \text{Card}(B)$ به معنای هم‌توان بودن دو مجموعه است:

$$\text{Card}(A) = \text{Card}(B) \Leftrightarrow A \simeq B \Leftrightarrow \exists f: A \rightarrow B \text{ (یک‌به‌یک و پوشا)}$$

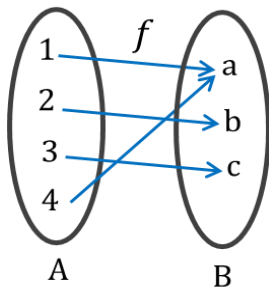
اما نامساوی $\text{Card}(A) \geq \text{Card}(B)$ چه زمانی برقرار است و اصلاً نحوه مقایسه کردن کاردینال‌ها چگونه است؟

لم:

شرط لازم و کافی برای آن که کاردینال A بزرگتر مساوی کاردینال B باشد؛ وجود تابع پوشا از A به B است.

$$\text{Card}(A) \geq \text{Card}(B) \Leftrightarrow \exists f: A \rightarrow B \text{ (پوشا)}$$

در این شکل تعداد اعضای A بیشتر از B است، می‌بینیم تابعی پوشا از A به B وجود دارد:

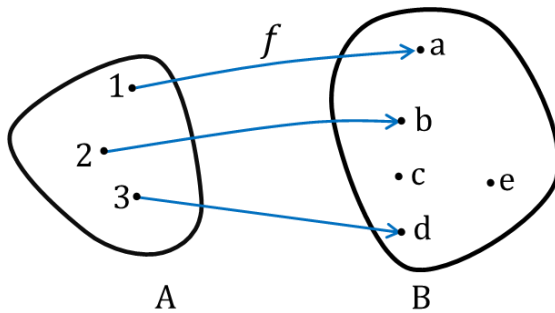


لم:

شرط لازم و کافی برای آن که کاردینال A کوچکتر مساوی کاردینال B باشد وجود تابع یک‌به‌یک از A به B است.

$$\text{Card}(A) \leq \text{Card}(B) \Leftrightarrow \exists f: A \rightarrow B \text{ (یک‌به‌یک)}$$

در شکل زیر چون تعداد اعضای A کمتر است، تابعی یک به یک از A به B وجود دارد:



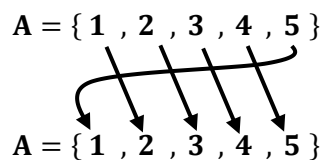
تعریف مجموعه نامتناهی:

اگر مجموعه A با یکی از زیرمجموعه های اکید خودش هم‌توان باشد می‌گوییم نامتناهی است. به عبارت دیگر: مجموعه A نامتناهی است اگر تابعی یک به یک و غیرپوشا از A به A موجود باشد.

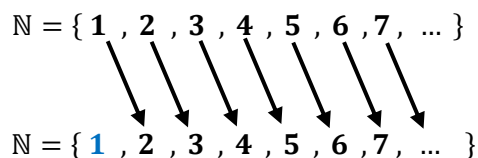
$$\exists B \subset A \text{ s.t. } A \simeq B \Leftrightarrow A \text{ نامتناهی است}$$

توضیح:

در مجموعه متناهی A ، اگر $f: A \rightarrow A$ یک به یک باشد حتما پوشا می‌شود. یعنی نمی‌توانید یک تناظر یک به یک بین A و یک زیرمجموعه اکید A برقرار کنید.



اما در مجموعه نامتناهی مانند $\mathbb{N}$ می‌توان تابع یک به یک ولی غیرپوشا از $\mathbb{N}$ به $\mathbb{N}$ پیدا کرد. برای مثال: $f(n) = n + 1$ یک تناظر یک به یک بین $\mathbb{N}$ و یکی از زیرمجموعه های اکید خودش است.



طبق شکل، مجموعه $\mathbb{N}$ با مجموعه اعداد طبیعی $n \geq 2$ هم‌توان است.

چند مثال از مجموعه های نامتناهی

$$\mathbb{N} = \{1, 2, 3, 4, 5, 6, 7, \dots\}$$

$$\mathbb{R} = (-\infty, \infty)$$

$$\mathbb{N}_e = \{2, 4, 6, 8, \dots\}$$

$$\mathbb{R}^2 = \mathbb{R} \times \mathbb{R}$$

$$\mathbb{N}_o = \{1, 3, 5, 7, \dots\}$$

$$\mathbb{Z} = \{\dots, -3, -2, -1, 0, 1, 2, 3, \dots\}$$

$$\mathbb{Q} = \left\{ \frac{n}{m} \mid n, m \in \mathbb{Z}, m \neq 0 \right\}$$

$$\mathbb{N} \times \mathbb{N} = \{(n, m) \mid n, m \in \mathbb{N}\}$$

$$\Sigma^* = \{\lambda\} \cup \Sigma \cup \Sigma^2 \cup \Sigma^3 \cup \dots$$

می‌دانیم که:

$$\mathbb{N} \subseteq \mathbb{Z} \subseteq \mathbb{Q} \subseteq \mathbb{R}$$

مفهوم چگال بودن (مطالعه آزاد)

مجموعه‌های $\mathbb{N}$ و $\mathbb{Z}$ در اعداد حقیقی چگال نیستند. بین دو عدد صحیح متوالی فاصله‌ای وجود دارد که در آن فاصله عدد صحیحی وجود ندارد. مثلاً بین 1 و 2 عدد صحیحی وجود ندارد.



اما مجموعه اعداد گویا ($\mathbb{Q}$) چگال است. در هر بازه (a, b) که $a < b$ بی‌نهایت عدد گویا وجود دارد. به همین دلیل فیثاغورثیان تصور می‌کردند هر اندازه در طبیعت، یک عدد گویا است. (این تصور نادرست را داشتند که هر نقطه / از محور x یک عدد گویاست) بعدها اما متوجه وجود حفره‌هایی در سیستم اعداد گویا شدند. برای مثال قطر مربعی به ضلع واحد باید $\sqrt{2}$ باشد اما این عدد با هیچ کسری به فرم $\frac{n}{m}$ برابر نیست. اثبات گویا نبودن $\sqrt{2}$ خالی از لطف نیست.

تمرین: ثابت کنید $\sqrt{2}$ گویا نیست.

برهان خلف: فرض کنیم $\sqrt{2}$ گویا باشد. پس می‌توان آن را به فرم $\sqrt{2} = \frac{n}{m}$ نوشت که n و m اعداد صحیح هستند. همچنین با ساده کردن صورت و مخرج تا حد امکان، می‌توان فرض کرد n و m عامل مشترک ندارند و نسبت به هم اولند.

$$\sqrt{2} = \frac{n}{m} \Rightarrow n = \sqrt{2}m \Rightarrow n^2 = 2m$$

n^2 زوج است پس n زوج است:

$$n = 2k \Rightarrow n^2 = 4k^2 \Rightarrow 4k^2 = 2m \Rightarrow 2k^2 = m$$

پس m زوج است. پس n و m هر دو بر 2 بخش پذیر شدند این تناقض است زیرا طبق فرض n و m نسبت به هم اولند. با این تناقض، فرض خلف رد می‌شود در نتیجه $\sqrt{2}$ عدد گویا نیست.

یادآوری:

اعداد حقیقی به دو مجموعه اعداد گویا ($\mathbb{Q}$) و اعداد گنگ ($\mathbb{Q}^c$) افراز می‌شود.

اعداد گویا به فرم $\frac{n}{m}$ هستند ($n, m \in \mathbb{Z}$) و در بسط اعشاری آنها همیشه دوره تکرار وجود دارد. اعداد گنگ به فرم $\frac{n}{m}$ قابل نمایش نیستند و در بسط اعشاری آنها دوره تکرار وجود ندارد.

مجموعه های نامتناهی شمارا:

اگر مجموعه نامتناهی A با $\mathbb{N}$ هم‌توان باشد، آن را شمارای نامتناهی گوئیم.

$$\text{Card}(A) = \text{Card}(\mathbb{N}) \Leftrightarrow A \text{ شمارای نامتناهی است}$$

مجموعه‌های $\mathbb{N}$ و $\mathbb{Z}$ و $\mathbb{Q}$ و $\mathbb{N} \times \mathbb{N}$ و $\sum^*$ شمارای نامتناهی‌اند.

مجموعه های ناشمارا:

اگر $\text{Card}(A) > \text{Card}(\mathbb{N})$ باشد A را ناشمارا گوئیم. به بیان دیگر، A ناشماراست اگر و تنها اگر هیچ تابع پوشایی از $\mathbb{N}$ به A موجود نباشد.

$$\text{Card}(A) > \text{Card}(\mathbb{N}) \Leftrightarrow A \text{ ناشماراست}$$

مجموعه‌های $\mathbb{R}$ و $\mathbb{R}^2$ و (a, b) و $[a, b]$ و $\mathbb{C}$ ناشمارا هستند.

بررسی چند مجموعه معروف:

الف: مجموعه اعداد طبیعی زوج با $\mathbb{N}$ هم‌توان است پس شمارای نامتناهی است.

$$\mathbb{N} = \{1, 2, 3, 4, \dots\}$$

$$f(n) = 2n$$

$$\mathbb{N}_e = \{2, 4, 6, 8, \dots\}$$

ب: مجموعه اعداد طبیعی فرد با $\mathbb{N}$ هم‌توان است پس شمارای نامتناهی است.

$$\mathbb{N} = \{1, 2, 3, 4, \dots\}$$

$$f(n) = 2n - 1$$

$$\mathbb{N}_o = \{1, 3, 5, 7, \dots\}$$

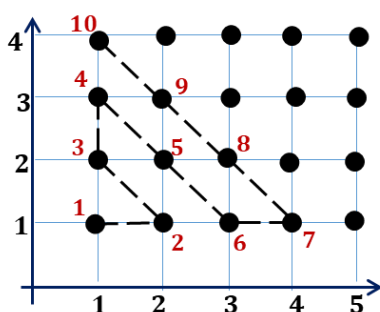
ج: مجموعه اعداد صحیح $\mathbb{Z}$ ، با $\mathbb{N}$ هم‌توان است پس شمارای نامتناهی است. برای نشان دادن این مطلب، اعداد طبیعی را به دو دسته زوج‌ها و فرد‌ها تفکیک می‌کنیم. تابع f زوج‌ها را به اعداد صحیح نامنفی و فرد‌ها را به اعداد صحیح منفی می‌نگارد:

$$f(n) = \begin{cases} \frac{n-2}{2} & \text{زوج } n \\ -\frac{n+1}{2} & \text{فرد } n \end{cases}$$

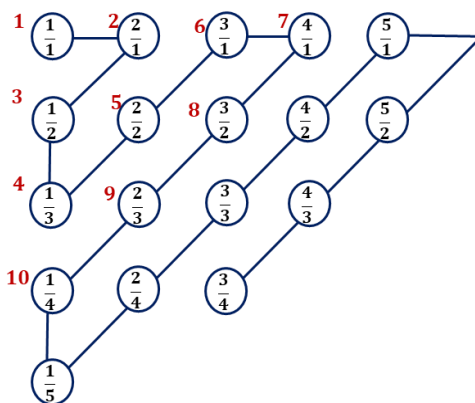
$$\mathbb{N} = \{1, 2, 3, 4, 5, 6, 7, 8, \dots\} = \{1, 3, 5, 7, \dots\} \cup \{2, 4, 6, 8, \dots\}$$

$$\mathbb{Z} = \{\dots, -3, -2, -1, 0, 1, 2, 3, \dots\} = \{-1, -2, -3, \dots\} \cup \{0, 1, 2, 3, \dots\}$$

د: مجموعه $\mathbb{N} \times \mathbb{N} = \{(n, m) \mid n, m \in \mathbb{N}\}$ شمارای نامتناهی است. هر زوج مرتب (n, m) یک نقطه در شکل زیر است. همانطور که نشان داده‌ایم می‌توان هر زوج مرتب را با یک عدد طبیعی برچسب گذاری کرد. پس مجموعه $\mathbb{N} \times \mathbb{N}$ با مجموعه اعداد طبیعی $\mathbb{N}$ هم‌توان است:



ه: مجموعه اعداد گویا $\mathbb{Q}$ شمارای نامتناهی است. اثبات مشابیه $\mathbb{N} \times \mathbb{N}$ است. اعداد گویای مثبت را می‌توان به این شکل با اعداد طبیعی برچسب گذاری کرد. همین‌روال برای اعداد گویای منفی قابل تکرار است.



و: مجموعه همه رشته‌های متناهی (Σ^*) شمارای نامتناهی است:

مجموعه الفبا (Σ) ناتهی و متناهی است. فرض کنیم $\Sigma = \{a, b\}$ دو عضوی باشد. Σ^m مجموعه همه رشته‌های به طول m است. تعداد این رشته‌ها $|\Sigma|^m$ است. مثلاً $\Sigma^2 = \{aa, ab, ba, bb\}$ چهارعضوی است و $\Sigma^3 = \{aaa, aab, \dots, bbb\}$ هشت عضوی است. همچنین $\Sigma^0 = \{\lambda\}$ که λ رشته تهی است.

Σ^* : مجموعه تمام رشته های (متناهی) قابل تولید با الفبای Σ را با این علامت نشان می دهیم:

$$\Sigma^* = \{\lambda\} \cup \Sigma \cup \Sigma^2 \cup \Sigma^3 \cup \Sigma^4 \cup \dots$$

$$\text{card}(\Sigma^*) = \text{card}(\mathbb{N})$$

معرفی نماد B^A :

B^A مجموعه همه توابع از A به B است:

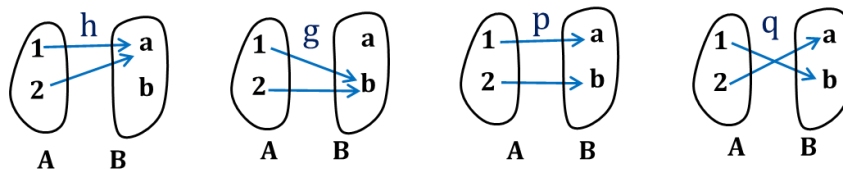
$$B^A = \{f \mid f \text{ is function from } A \text{ to } B\}$$

تعداد این توابع برابر است با:

$$\text{card}(B^A) = \text{card}(B)^{\text{card}(A)}$$

مثال: مجموعه همه توابع از $A = \{1, 2\}$ به $B = \{a, b\}$ را با B^A نشان می دهیم که شامل 4 تابع است:

$$B^A = \{h, g, p, q\}$$



یک نمونه معروف:

$\{0, 1\}^{\mathbb{N}}$ مجموعه توابع از $\mathbb{N}$ به $\{0, 1\}$ است. هر کدام از این توابع یک دنباله باینری است. پس $\{0, 1\}^{\mathbb{N}}$ مجموعه همه دنباله های باینری است:

$$\{0, 1\}^{\mathbb{N}} = \left\{ \begin{array}{l} \langle 0 \ 1 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1 \dots \rangle, \\ \langle 1 \ 1 \ 1 \ 0 \ 1 \ 1 \ 1 \ 0 \ 1 \ 1 \ 0 \ 0 \dots \rangle, \\ \langle 1 \ 1 \ 1 \ 1 \ 0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 1 \ 1 \dots \rangle, \\ \langle 1 \ 1 \ 0 \ 0 \ 1 \ 0 \ 0 \ 1 \ 1 \ 0 \ 0 \ 0 \dots \rangle, \\ \langle 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \dots \rangle, \\ \langle 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 1 \dots \rangle, \\ \dots \end{array} \right\}$$

نشان می دهیم که این مجموعه، ناشماراست و کاردینال آن اکیدا بزرگتر از $\text{card}(\mathbb{N})$ است. یادآوری: اگر $\text{Card}(A) > \text{Card}(\mathbb{N})$ باشد A را ناشمارا گوییم. برای اثبات ناشمارا بودن A کافیست نشان دهیم هیچ تابع پوشایی از $\mathbb{N}$ به A وجود ندارد.

قضیه: مجموعه همه دنباله های باینری ناشماراست:

اثبات: (برهان خلف) فرض کنیم تابع پوشایی مانند f از $\mathbb{N}$ به $\{0, 1\}^{\mathbb{N}}$ وجود داشته باشد:

$$f: \mathbb{N} \rightarrow \{0, 1\}^{\mathbb{N}}$$

$$\begin{aligned}
 f(1) &= < 010101010101 \dots > \\
 f(2) &= < 111011101100 \dots > \\
 f(3) &= < 111100001111 \dots > \\
 f(4) &= < 110010011000 \dots > \\
 f(5) &= < 111111111111 \dots > \\
 f(6) &= < 000000000011 \dots > \\
 &\dots \qquad \dots
 \end{aligned}$$

حالا دنباله باینری y را طوری می‌نویسیم که اولین رقم آن با اولین رقم $f(1)$ تفاوت داشته باشد، در دومین رقم با دومین رقم $f(2)$ اختلاف داشته باشد، در سومین رقم با سومین رقم $f(3)$... و به همین ترتیب در رقم k مخالف با رقم k ام از $f(k)$ باشد.

$$\begin{aligned}
 f(1) &= < \mathbf{0}10101010101 \dots > \\
 f(2) &= < 1\mathbf{1}1011101100 \dots > \\
 f(3) &= < 11\mathbf{1}100001111 \dots > \\
 f(4) &= < 110\mathbf{0}10011000 \dots > \\
 f(5) &= < 1111\mathbf{1}1111111 \dots > \\
 f(6) &= < 00000\mathbf{0}000011 \dots > \\
 &\dots \qquad \dots
 \end{aligned}
 \Rightarrow y = (100101 \dots)$$

با توجه به نحوه نوشته شدن y ، معلوم است $y \neq f(1)$ و $y \neq f(2)$ و ... پس y در برد f قرار ندارد. یعنی تابع f پوشا نیست. (تناقض با فرض). پس ثابت کردیم تابع پوشایی از $\mathbb{N}$ به $\{0,1\}^{\mathbb{N}}$ وجود ندارد. در نتیجه $\{0,1\}^{\mathbb{N}}$ ناشماراست.

توجه: به تکنیک به کار رفته در این برهان، برای تولید y ، تکنیک قطری کانتور می‌گویند.

مجموعه توانی:

مجموعه همه زیرمجموعه‌های X را با $P(X)$ نشان می‌دهیم و به آن مجموعه توانی X می‌گوییم. از علامت های 2^X یا $\{0,1\}^X$ نیز برای اشاره به این مجموعه استفاده می‌شود. هر زیرمجموعه از X مرتبط است با یک دنباله باینری که عضویت یا عدم عضویت عناصر را نشان می‌دهد.

مثال:

$$X = \{a, b, c\}$$

$$\begin{aligned}
 P(X) &= \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{b, c\}, \{a, c\}, \{a, b, c\}\} \\
 &\cong \{000, 100, 010, 001, 110, 011, 101, 111\}
 \end{aligned}$$

همواره داریم:

$$\text{card } X = \alpha \Leftrightarrow \text{card } P(X) = 2^\alpha$$

اعداد اصلی $\aleph_0$ و $\mathfrak{c}$:

کاردینال $\aleph_0$ را با $\aleph_0$ (aleph-nought, aleph-zero) نشان می‌دهیم: $\text{card}(\mathbb{N}) = \aleph_0$
 پس کاردینال هر مجموعه شمارای نامتناهی $\aleph_0$ است.

$$\text{card}(\mathbb{N}) = \text{card}(\mathbb{Z}) = \text{card}(\mathbb{Q}) = \text{card}(\mathbb{Z}^*) = \aleph_0$$

اما مجموعه های $\{0,1\}^{\mathbb{N}}$ و $\mathbb{R}$ نامشمارا هستند. کاردینال این مجموعه‌ها را با $\mathfrak{c}$ نشان می‌دهیم.

$$\text{card}(\mathbb{R}) = \text{card}(\{0,1\}^{\mathbb{N}}) = \mathfrak{c}$$

◀ دقت کنید که مجموعه‌های نامشمارا با کاردینال‌های بزرگتر از $\mathfrak{c}$ نیز وجود دارند.

یک رابطه مهم:

$$2^{\aleph_0} = \mathfrak{c}$$

مجموعه همه زیرمجموعه های $\mathbb{N}$ با مجموعه $\{0,1\}^{\mathbb{N}}$ معادل و هم ارز است. مطابق شکل هر دنباله باینری نشان دهنده یک زیرمجموعه $\mathbb{N}$ است. کفایت به جایگاه رقم های 1 توجه کنیم:

$$\{0,1\}^{\mathbb{N}} = \left\{ \begin{array}{l} 010101010101\dots \rightarrow \{2,4,6,8,\dots\} \\ 101010101010\dots \rightarrow \{1,3,5,7,\dots\} \\ 111111111111\dots \rightarrow \{1,2,3,4,5,\dots\} \\ 110000000000\dots \rightarrow \{1,2\} \\ 000000000000\dots \rightarrow \{\} \\ \vdots \end{array} \right.$$

نظیر هر زیرمجموعه از $\mathbb{N}$ یک دنباله باینری وجود دارد و بالعکس. یعنی $P(\mathbb{N}) \approx \{0,1\}^{\mathbb{N}}$.
 بنابراین مجموعه همه زیرمجموعه‌های $\mathbb{N}$ ، هم‌توان با $\mathbb{R}$ است. پس بین اعداد کاردینال و این رابطه وجود دارد:

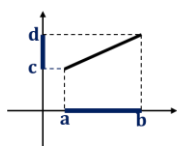
$$P(\mathbb{N}) \approx \{0,1\}^{\mathbb{N}} \approx \mathbb{R} \Rightarrow \text{card}P(\mathbb{N}) = 2^{\aleph_0} = \text{card}\mathbb{R} = \mathfrak{c}$$

قضیه: بازه‌ی $(0,1)$ نامشمار است.

[اثبات این حکم مشابه اثبات نامشمارا بودن $\{0,1\}^{\mathbb{N}}$ است. از تکنیک قطری کانتور استفاده می‌شود.]

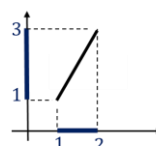
قضیه: فرض کنیم $a < b$ باشد. همه بازه های زیر نامشمارا هستند.

$(-\infty, \infty)$, $[0, \infty)$, (a, b) , $[a, b]$, (a, ∞) , $(-\infty, b]$ نامشمارا هستند و کاردینال آنها $\mathfrak{c}$ است.



$$f: [a, b] \rightarrow [c, d]$$

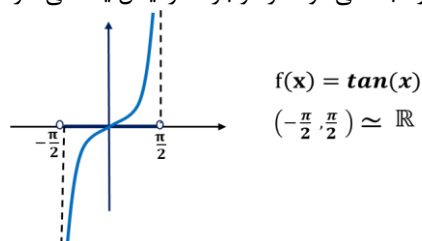
$$f(x) = \frac{d-c}{b-a}(x-a) + c$$



$$f: [1, 2] \rightarrow [1, 3]$$

$$f(x) = 2x - 1$$

در تصویر بالا با ذکر مثال هم‌توان (هم‌ارز) بودن بازه‌های بسته را نشان داده ایم. برای مثال نشان داده ایم که بازه‌های $[1,2]$ و $[1,3]$ هم‌توان هستند. در تصویر پایین تابع تانژانت یک تناظر یک به یک بین $(-\frac{\pi}{2}, \frac{\pi}{2})$ و $(-\infty, +\infty)$ ایجاد می‌کند و ثابت می‌شود هر دو بازه کاردینال یکسانی دارند.



قضیه: اجتماع مجموعه متناهی و مجموعه شمارای نامتناهی، مجموعه شمارای نامتناهی است.

ابتدا عناصر مشترک را از B برداریم چون تاثیری در اجتماع ندارد.

قضیه: اجتماع تعدادی شمارا مجموعه شمارای نامتناهی، مجموعه شمارای نامتناهی است.

$$\forall i \ A_i \simeq \mathbb{N} \Rightarrow \bigcup_{i=1}^{\infty} A_i \simeq \mathbb{N}$$

مثال: اجتماع دو مجموعه شمارای نامتناهی، شمارای نامتناهی است.

بدون کاسته شدن از کلیت فرض کنیم مجموعه‌ها مجزا هستند.

$$A = \{a_1, a_2, a_3, a_4, a_5, \dots\}$$

$$B = \{b_1, b_2, b_3, b_4, b_5, \dots\}$$

$$\left. \begin{array}{l} A \simeq \mathbb{N}_e \\ B \simeq \mathbb{N}_o \end{array} \right\} \Rightarrow A \cup B \simeq \mathbb{N}_e \cup \mathbb{N}_o \Rightarrow A \cup B \simeq \mathbb{N}$$

قضیه مجموعه توانی:

ثابت می‌شود که برای هر عدد اصلی (متناهی یا نامتناهی) همواره: $2^\alpha > \alpha$

$$\text{مثلاً: } 2^c > c \text{ و } 2^{\aleph_0} > \aleph_0$$

فرض پیوستار (مطالعه بیشتر):

فرض پیوستار می‌گوید: بین $\aleph_0$ و $c = 2^{\aleph_0}$ هیچ کاردینالی وجود ندارد.

تا کنون هر کدام از زیرمجموعه‌های نامتناهی $\mathbb{R}$ را که بررسی کرده‌ایم یا با $\mathbb{N}$ هم‌توان بود یا با $\mathbb{R}$ هم‌توان بود. آیا مجموعه‌ای مانند A وجود دارد که کاردینالش بین $\aleph_0$ و c باشد؟ ثابت شده است که تلاش برای پاسخ دادن به این سوال بر مبنای اصول موضوع پذیرفته شده نظریه مجموعه‌ها (ZFC) تلاشی بیهوده است. به عبارتی این حکم نه قابل اثبات است و نه قابل رد است. می‌توانیم بپذیریم که: چنین مجموعه‌ای وجود ندارد و اعداد کاردینال $\aleph_0$ و c متوالی هستند و بین آنها هیچ کاردینالی قرار ندارد. (فرض پیوستار). همچنین می‌توانیم بپذیریم که چنین مجموعه‌ای وجود دارد. در هر دو حالت یک سیستم سازگار از اصول موضوع خواهیم داشت.

چکیده موارد مهم:

Σ : مجموعه الفباست و متناهی است.

Σ^* : مجموعه همه رشته‌های متناهی است و با $\mathbb{N}$ هم‌توان است. پس شمارای نامتناهی است.

$P(\Sigma^*)$: مجموعه همه زبان‌ها است و با $P(\mathbb{N})$ هم‌توان است. پس ناشمارا است.

هم‌ارزی‌ها و تساوی‌های مهم:

$$\{0,1\}^{\mathbb{N}} \approx P(\mathbb{N}) \approx [a,b] \approx (c,d) \approx \mathbb{R} \rightarrow 2^{\aleph_0} = c$$

$$\mathbb{N} \times \mathbb{N} \approx \mathbb{N} \rightarrow \aleph_0 \aleph_0 = \aleph_0$$

$$\mathbb{R} \times \mathbb{R} \approx \mathbb{R} \rightarrow cc = c$$

جدول کاردینال‌ها:

1- مجموعه‌های متناهی

مجموعه‌های متناهی	کاردینال
$\emptyset = \{ \}$	0
$\{ 1 \}$	1
$\{ 1, 2 \}$	2
$\{ 1, 2, 3 \}$	3

...	...
$\mathbb{N}_k = \{1, 2, \dots, k\}$	k
مجموعه الفبا Σ	تعداد الفبا

2- مجموعه های شمارای نامتناهی

شمارای نامتناهی	کاردینال
$\mathbb{N}$	$\aleph_0$
$\mathbb{N}_e$	$\aleph_0$
$\mathbb{N}_o$	$\aleph_0$
$\mathbb{Z}$	$\aleph_0$
$\mathbb{N} \times \mathbb{N}$	$\aleph_0$
$\mathbb{Q}$	$\aleph_0$
Σ^* همه رشته های متناهی	$\aleph_0$

3- مجموعه های ناشمارای هم توان با $\mathbb{R}$

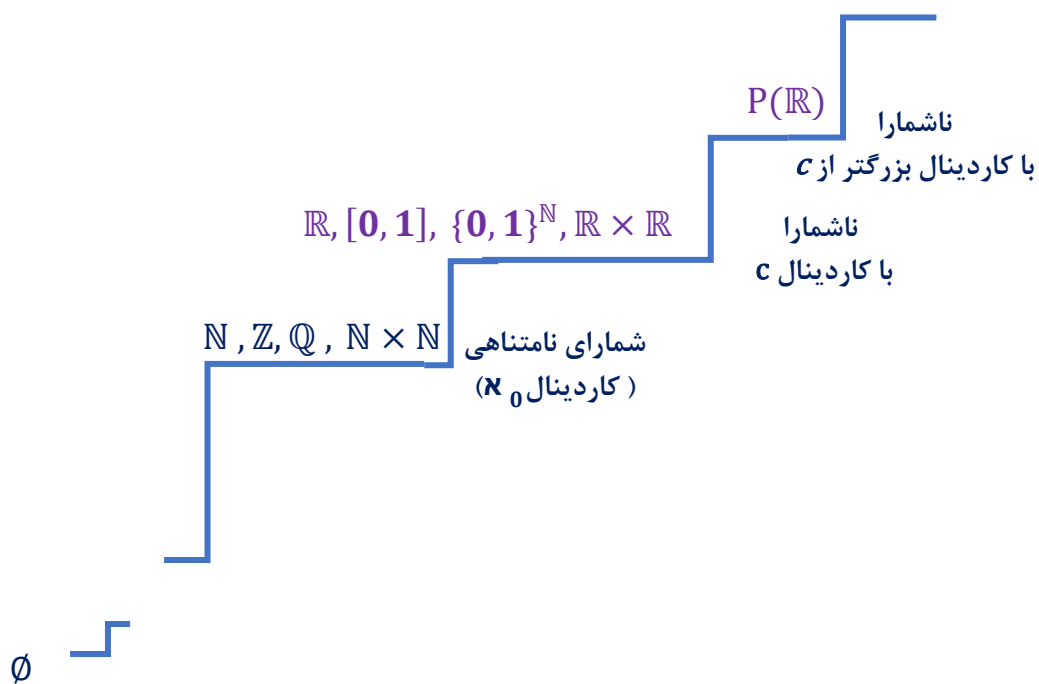
نا شمارای هم توان با $\mathbb{R}$	کاردینال
$\{0,1\}^{\mathbb{N}}$	$2^{\aleph_0} = c$
$P(\mathbb{N})$	$2^{\aleph_0} = c$

$[a, b]$	c
(c, d)	c
$\mathbb{Q}^c$	c
$\mathbb{R}$	c
$\mathbb{R} \times \mathbb{R}$	c
مجموعه همه زبانها $P(\Sigma^*)$	$2^{\aleph_0} = c$

4- مجموعه های ناشمارای بزرگتر

نا شمارای بزرگتر از $\mathbb{R}$	کاردینال
$P(\mathbb{R})$	2^c

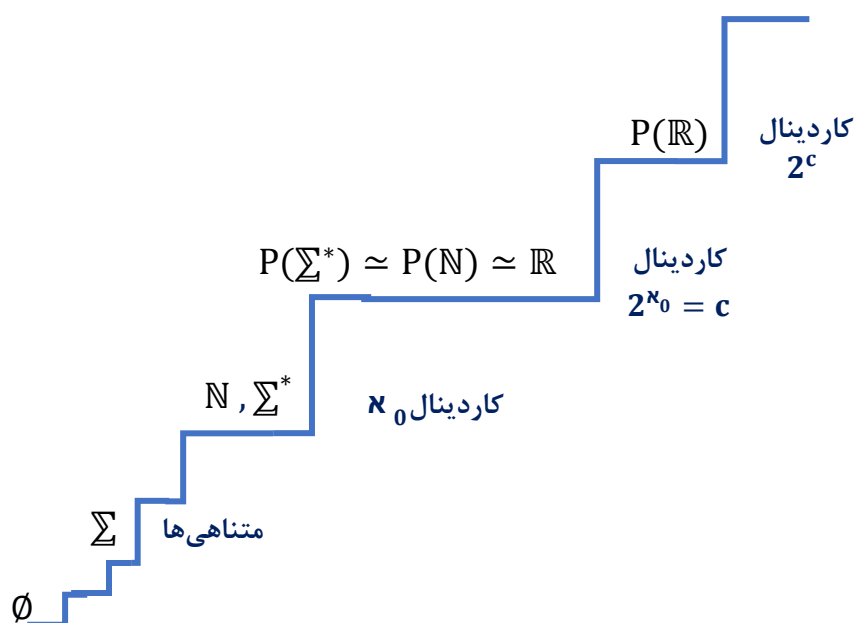
5- مقایسه پله ای مجموعه ها:



متناهی ها Σ

Σ

6- مقایسه پله‌ای مجموعه های توانی:





-2-

فصل اول: مفاهیم پایه.

جلسه دوم: عملگرها روی رشته ها و زبان ها، تعریف گرامر

عملگرها روی رشته ها:

مجموعه الفبا را با Σ نشان می‌دهیم که مجموعه ای ناتهی و متناهی است. هر رشته یک دنباله متناهی از الفبا است. برای مثال رشته $w = abab$ رشته ای به طول 4 تشکیل شده از الفبای a و b است. رشته تهی (blank- empty- null) را با نماد λ یا ϵ نمایش می‌دهیم. تعداد الفبای موجود در رشته w را طول (اندازه) w می‌نامیم و با $|w|$ نشان می‌دهیم. طول رشته λ صفر است. به این نمونه‌ها توجه کنید:

$$w = abaaab \Rightarrow |w| = 6$$

$$w = \lambda \Rightarrow |w| = 0$$

$$v = ab\lambda a \Rightarrow |v| = 3$$

عملگر الحاق (concatenation):

الحاق رشته w و رشته v رشته wv است. واضح است که ترتیب در الحاق اهمیت دارد. اگر $w = ab$ و $v = aa$ آنگاه $wv = abaa$ اما $vw = aaab$ پس عمل الحاق خاصیت جابجایی ندارد. (آبلی نیست). هرگاه w رشته ای به طول n و v رشته ای به طول m باشد آنگاه $|wv| = n + m$. رشته تهی در الحاق خنثی است: $w\lambda = w$, $\lambda w = w$. از نماد w^n برای n بار الحاق رشته w با خودش استفاده می‌کنیم. به این نمونه ها توجه کنید:

$$w^0 = \lambda, \quad w^1 = w, \quad w^2 = ww,$$

دقت کنید: $(xw)^n$ با $x^n w^n$ برابر نیست.

$$(xw)^n = \underbrace{xw \ xw \ xw \ \dots \ xw}_n \neq x^n w^n$$

بستار ستاره‌ی Σ^* :

مجموعه تمام رشته های متناهی با الفبای Σ را با علامت Σ^* نشان دادیم که شامل رشته تهی هم می شود. مجموعه تمام رشته های متناهی ناتهی را Σ^+ می نامیم.

$$\Sigma^* = \{\lambda\} \cup \Sigma \cup \Sigma^2 \cup \Sigma^3 \cup \Sigma^4 \cup \dots$$

$$\Sigma^+ = \Sigma \cup \Sigma^2 \cup \Sigma^3 \cup \Sigma^4 \cup \dots$$

در نتیجه:

$$\Sigma^+ = \Sigma^* - \{\lambda\}$$

توجه: عملگر بستار ستاره برای هر مجموعه از رشته ها نیز تعریف می شود.

$$A^* = \{\lambda\} \cup A \cup A^2 \cup A^3 \cup A^4 \cup \dots$$

عملگر وارون: (reverse)

فرض کنیم $w = a_1 a_2 a_3 \dots a_n$ که هر a_i یک الفبا باشد. معکوس یا وارون w را با w^R نشان می دهیم و با عکس کردن ترتیب الفبا به دست می آید:

$$w = a_1 a_2 a_3 \dots a_n \Rightarrow w^R = a_n a_{n-1} \dots a_2 a_1$$

مثلاً:

$$w = abc \Rightarrow w^R = cba$$

به وضوح w و w^R اندازه یکسان دارند:

$$|w^R| = |w|$$

دقت کنید:

$$(wx)^R = x^R w^R$$

$$(uvw)^R = w^R v^R u^R$$

مثال: اگر $w = abc$ و $x = bbaa$:

$$(wx)^R = (abcbbaa)^R = aabbcbba = x^R w^R$$

نتیجه:

$$(w^n)^R = (w^R)^n$$

مثال:

$$(w^2)^R = (ww)^R = w^R w^R = (w^R)^2$$

یک نکته ساده: عملگر وارون، خودش را خنثی می‌کند. اگر از یک رشته دوبار وارون بگیریم، به همان رشته اولیه خواهیم رسید. به طور کلی تعداد زوجی وارون گرفتن، یک عمل خنثی است:

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

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

رشته های آینه ای:

w آینه ای است اگر و تنها اگر $w^R = w$. برای مثال $aabbaa$ و $bacab$ آینه ای هستند.

رشته های آینه ای به طول زوج به فرم vv^R هستند:

$$w = \underbrace{aab}_{v} \underbrace{baa}_{v^R}$$

رشته های آینه ای به طول فرد به فرم vav^R هستند که a یک الفباست.

$$w = \underbrace{bac}_{v} \underbrace{acab}_{v^R}$$

مجموعه های مرتبط با رشته های آینه ای

$\{w \mid w^R = w\}$	مجموعه رشته های آینه ای
$\{w w^R \mid w \in \Sigma^*\}$	رشته های آینه ای به طول فرد
$\{w a w^R \mid w \in \Sigma^*, a \in \Sigma\}$	رشته های آینه ای به طول فرد

زبان (language):

هر زیرمجموعه از Σ^* یک زبان نام دارد. پس هر زبان مجموعه ای دلخواه از رشته هاست. Σ^* بزرگترین زبان با الفبای Σ است. تهی، کوچکترین زبان است. چند نمونه از زبان ها:

$$L_1 = \{ \}$$

$$L_2 = \{\lambda\}$$

$$L_3 = \{a, b, aa, ab\}$$

$$L_4 = \{a^n b^m \mid n, m \geq 0\}$$

$$L_5 = \{a^{3n} \mid n \geq 1\}$$

عملگرها روی زبان ها:

هر زبان یک مجموعه است. پس تمام عملگرهای مجموعه ای (اجتماع، اشتراک، تفاضل، مکمل) روی زبان ها نیز تعریف می شوند.

توضیح	عملگرهای مجموعه ای
رشته هایی که عضو L_1 یا L_2 هستند.	$L_1 \cup L_2$
رشته هایی مشترک L_1 و L_2 .	$L_1 \cap L_2$
رشته هایی که عضو L_1 هستند اما عضو L_2 نیستند.	$L_1 - L_2$
رشته های غیرمشترک (که فقط عضوی از دو زبان هستند).	$L_1 \Delta L_2$
رشته هایی که عضو L_1 نیستند.	$L_1^c = \overline{L_1} = \Sigma^* - L_1$

مثال: اشتراک دو زبان L_1 و L_2 برابر با L است.

	$L_1 - L_2$			$L_1 \Delta L_2$	
--	-------------	--	--	------------------	--

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

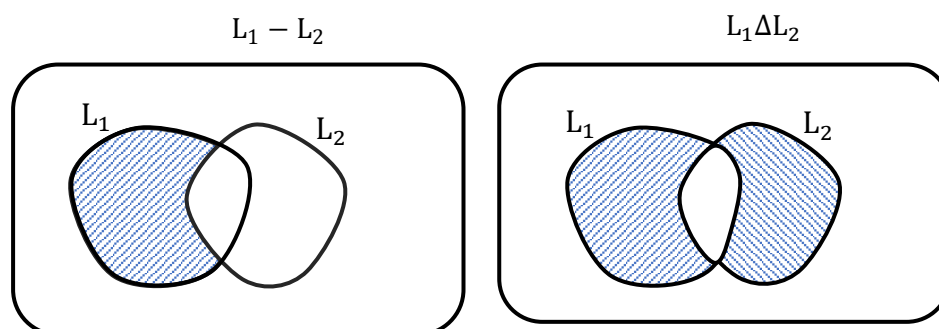
$$L_2 = \{a^n b^n c^k \mid n, k \geq 0\}$$

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

یادآوری هایی از نظریه مجموعه ها:

$$L_1 - L_2 = L_1 \cap L_2^C$$

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



عملگرهای وارون، الحاق، بستار:

با وارون کردن تمام رشته های عضو L زبان L^R به دست می آید:

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

مثال:

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

الحاق دو زبان:

$$L_1 L_2 = \{w_1 w_2 \mid w_1 \in L_1, w_2 \in L_2\}$$

L^2 الحاق L با خودش است.

مثال:

$$L = a^+ = \{a, a^2, a^3, \dots\} \Rightarrow L^2 = LL = \{a^2, a^3, a^4, \dots\}$$

در این مثال $L^2 \subseteq L$ است. اما در حالت کلی رابطه ای بین آنها وجود ندارد.

مثال :

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

در این مثال L و L^2 هیچکدام زیرمجموعه دیگری نیست.

توجه: اگر $\lambda \in L$ آنگاه هر $w \in L$ را می‌توان به صورت λw نوشت پس $w = \lambda w \in L^2$ یعنی $L \subseteq L^2$ می‌شود.

$$\lambda \in L \Rightarrow L \subseteq L^2$$

توجه: در ضرب دکارتی: $|A \times B| = |A||B|$ اما در الحاق دو زبان داریم:

$$|L_1 L_2| \leq |L_1||L_2|$$

مثال: اگر $L_1 = \{\lambda, a, aa\}$ و $L_2 = \{b, ab\}$ آنگاه:

$$L_1 L_2 = \{b, ab, aab, aaab\}$$

می‌بینیم که $|L_1| = 3$ و $|L_2| = 2$ اما $|L_1 L_2| = 4$

بستار زبان L :

$$L^* = \{\lambda\} \cup L \cup L^2 \cup L^3 \cup \dots$$

$$L^+ = L \cup L^2 \cup L^3 \cup \dots$$

توجه: $L^0 = \{\lambda\}$ است، حتی اگر L تهی باشد.

توجه: همواره $\lambda \in L^*$ و $L^+ \subseteq L^*$ است. همچنین اگر $\lambda \in L$ باشد آنگاه $\lambda \in L^+$.

اگر L شامل رشته تهی باشد داریم:

$$L^+ = L^*$$

اگر L شامل رشته تهی نباشد داریم:

$$L^+ = L^* - \{\lambda\}$$

نکته:

طبق قانون دمورگان، مکمل اجتماع و اشتراک چنین است:

$$(L_1 \cup L_2)^c = L_1^c \cap L_2^c$$

$$(L_1 \cap L_2)^c = L_1^c \cup L_2^c$$

نکته:

عملگر بستار روی اجتماع و اشتراک، و تفاضل توزیع نمی‌شود.

البته این احکام به صورت یک طرفه برقرارند (نه به صورت تساوی):

$$L_1^* \cup L_2^* \subseteq (L_1 \cup L_2)^*$$

$$(L_1 \cap L_2)^* \subseteq L_1^* \cap L_2^*$$

نماد $n_a(w)$:

تعداد تکرار الفبای a در رشته w را با $n_a(w)$ نشان می‌دهیم.

سوالات تالیفی:

مسئله 1: هرگاه $L = \{b, ab, aba\}$ کدام گزینه در L^* قرار ندارد؟

$ab^3abab^2a^2b$ -4	ab^3ab^2ab -3	$(ba)^3b$ -2	b^2aba^2b -1
---------------------	-----------------	--------------	----------------

مسئله 2: کدام نادرست است؟

$(L^+)^* = L^*$ -4	$(L^*)^+ = L^+$ -3	$(L^*)^* = L^{**}$ -2	$(L^+)^+ = L^{++}$ -1
--------------------	--------------------	-----------------------	-----------------------

مسئله 3: کدام یک از این جملات درستند؟

الف: اگر $L^2 \subseteq L$ آنگاه L نامتناهی است.

ب: $L \subseteq L^2$ اگر و تنها اگر $\lambda \in L$

1- الف	2- ب	3- هردو	4- هیچکدام
--------	------	---------	------------

مسئله 4: بررسی کنید که برای چه زبان هایی تساوی های زیر درست است؟

$$\bar{L}^* = (\bar{L})^*$$

$$\bar{L}^+ = (\bar{L})^+$$

پاسخ سوالات تالیفی:

مسئله 1: گزینه 4

گزینه های 1 و 2 و 3 را می توان با الحاق $\{b, ab, aba\}$ ساخت.

مسئله 2: گزینه 3

گزینه 3 نادرست است. یک مثال نقض این زبان است:

$$L = \{a\} \quad \Rightarrow \quad \begin{aligned} L^* &= \{\lambda, a, aa, \dots\} \\ L^+ &= \{a, aa, \dots\} \\ (L^*)^+ &= \{\lambda, a, aa, \dots\} \end{aligned}$$

شکل صحیح این گزینه به صورت $(L^*)^+ = L^*$ است.

مساله 3: گزینه 4

الف: نادرست است. اگر $L^2 \subseteq L$ آنگاه می توان نتیجه گرفت که $L = \{\}$ یا $L \in L$ یا L نامتناهی است.
 ب: نادرست است. $L \subseteq L^2$ اگر و تنها اگر $L = \{\}$ یا $L \in L$.
 اگر در صورت سوال ناتهی بودن زبان فرض می شد، جمله (ب) درست می شد.

مساله 4:

الف: تساوی: $\bar{L}^* = (\bar{L})^*$ هیچ گاه درست نیست زیرا $\lambda \in L^*$ و $\lambda \in (\bar{L})^*$ پس $\lambda \notin \overline{(L^*)}$.
 پس یک طرف شامل λ و طرف دیگر فاقد λ است.

ب: تساوی $\overline{L^+} = (\bar{L})^+$ برای برخی زبان ها مانند Σ^* و $\emptyset$ برقرار است.

فصل اول: مفاهیم پایه

جلسه دوم بخش دوم: گرامرها

گرامرها (grammar):

برخی زبان ها (نه همه آنها) دارای گرامر هستند. هر گرامر یک چهارتایی $G = (V, T, S, P)$ است. V مجموعه ای متناهی از متغیرها است که با حروف بزرگ نمایش داده می شوند. T یک مجموعه متناهی از نمادها (شامل الفبا) است که با حروف کوچک نشان داده می شوند و به آنها پایانه نیز می گوئیم. S یکی از متغیرهاست که به عنوان متغیر آغازین استفاده می شود. P مجموعه ای از قواعد تولید است. در این گرامر:

$$\begin{aligned} G: \\ S &\rightarrow aSb|cA \\ A &\rightarrow Ac|\lambda \end{aligned}$$

متغیرها، پایانه ها و قواعد عبارتند از:

$$G = \left(\overbrace{\{S, A\}}^V, \overbrace{\{a, b, c\}}^T, S, \overbrace{\{S \rightarrow aSb, S \rightarrow cA, A \rightarrow Ac, A \rightarrow \lambda\}}^P \right)$$

مجموعه تمام رشته هایی که با شروع از S و استفاده از قواعد تولید به دست می آیند زبان تولید شده توسط گرامر نام دارد و آن را با $L(G)$ نشان می دهیم.

مثال 1: هر زبان متناهی، دارای گرامر است. برای مثال $L = \{a, ab, aab\}$ را در نظر بگیرید. گرامرهای G_1 و G_2 هر دو زبان L را تولید می کنند:

G_1 :
 $S \rightarrow a$
 $S \rightarrow ab$
 $S \rightarrow aab$

G_2 :
 $S \rightarrow aA|a$
 $A \rightarrow aB|b$
 $B \rightarrow b$

$$L(G_1) = L(G_2) = \{a, ab, aab\}$$

چنین گرامرهایی را معادل می نامیم.

توجه: همه زبان ها دارای گرامر نیستند. هر زبانی که دارای گرامر باشد، در واقع دارای بینهایت گرامر و بی نهایت ماشین معادل است. در مورد ماشین ها در جلسات بعد صحبت می کنیم.

ماشین	گرامر	زبان
	G_1 : $S \rightarrow a ab aab$ G_2 : $S \rightarrow aA a$ $A \rightarrow aB b$ $B \rightarrow b$	$L = \{a, ab, aab\}$

مثال 2: برای زبان L گرامر پیدا کنیم.

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

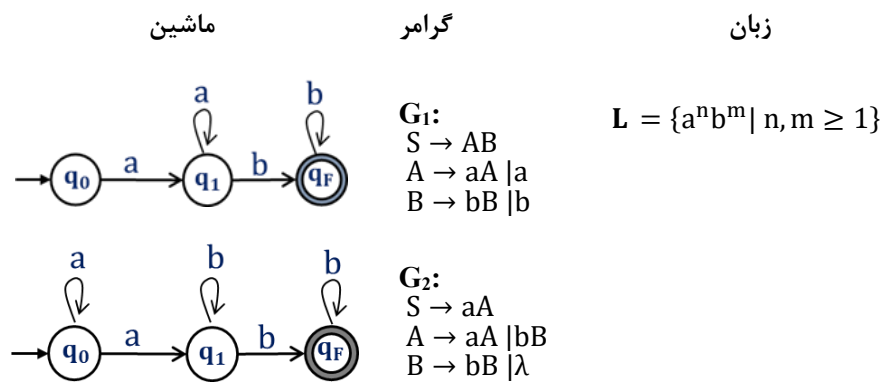
G_1 :
 $S \rightarrow AB$
 $A \rightarrow aA | a$
 $B \rightarrow bB | b$

G_2 :
 $S \rightarrow aA$
 $A \rightarrow aA | bB$
 $B \rightarrow bB | \lambda$

گرامرهای G_1 و G_2 هر دو دقیقاً زبان L را تولید می کنند:

$$L(G_1) = L(G_2) = L$$

گرامر G_1 قاعده تولید $S \rightarrow AB$ دارد که در سمت راست شامل دو متغیر است و اصطلاحاً غیرخطی است. اما گرامر G_2 خطی است در سمت راست همه قواعدش حداکثر یک متغیر به کار رفته است. با این حال هر دو گرامر معادلند چون هر دو یک زبان را تولید می کنند.



مثال 3: برای زبان L گرامر پیدا کنیم.

$$L = \{a^n b^m \mid n = m\} = \{a^n b^n \mid n \geq 0\} = \{\lambda, ab, a^2 b^2, a^3 b^3, \dots\}$$

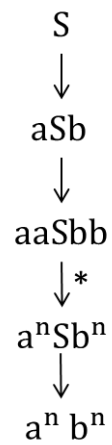
برای آنکه تعداد مساوی a و b در هر رشته تولید شود و فرم رشته ها $a^n b^n$ باشد از قواعد

$$G:$$

$$S \rightarrow aSb \mid \lambda$$

استفاده می کنیم.

مراحل تولید رشته ها با شروع از S : علامت $*$ نشانه استفاده چندباره از یک قاعده است.



توجه:

در مثال قبل اگر اگر خواهیم $n \geq 1$ باشد و رشته تهی تولید نشود از گرامر $S \rightarrow aSb \mid ab$ استفاده می کنیم.

مثال 4: برای زبان L گرامر پیدا کنیم.

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

پاسخ:

$$\begin{array}{l}
 G: \\
 S \rightarrow aSb \mid A \\
 A \rightarrow aA \mid \lambda
 \end{array}$$

$$\begin{array}{c}
 S \\
 \downarrow * \\
 a^k S b^k \\
 \downarrow \\
 a^k A b^k \\
 \downarrow * \\
 a^k a^i A b^k \\
 \downarrow \\
 a^k a^i b^k
 \end{array}
 \qquad
 L(G)=L$$

مثال ۵: برای زبان L گرامر پیدا کنیم.

$$L = \{0^{2n}1^{2m+1} \mid n, m \geq 0\} \quad \Sigma = \{0,1\}$$

G_1 :

$S \rightarrow AB$
 $A \rightarrow 00A \mid \lambda$
 $B \rightarrow 11B \mid 1$

G_2 :

$S \rightarrow 00S \mid B1$
 $B \rightarrow B11 \mid \lambda$

G_3 :

$S \rightarrow 00S \mid 1B$
 $B \rightarrow 11B \mid \lambda$

گرامرهای G_1 و G_2 و G_3 معادلند و همگی زبان L را تولید می‌کنند:

$$L(G_1) = L(G_2) = L(G_3) = L$$

ماشین	گرامر	زبان
	G_1 : $S \rightarrow AB$ $A \rightarrow 00A \mid \lambda$ $B \rightarrow 11B \mid 1$	$L = \{0^{2n}1^{2m+1} \mid n, m \geq 0\}$
	G_2 : $S \rightarrow 00S \mid B1$ $B \rightarrow B11 \mid \lambda$	
	G_3 : $S \rightarrow 00S \mid 1B$ $B \rightarrow 11B \mid \lambda$	

مثال ۶: برای زبان L گرامر پیدا کنیم.

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

G:

$$S \rightarrow AB$$

$$A \rightarrow aAb \mid \lambda$$

$$B \rightarrow aBbb \mid \lambda$$

$$\underbrace{a^m b^m}_A \underbrace{a^n b^{2n}}_B$$

مثال ۷ (مهم): برای زبان L گرامر پیدا کنیم.

$$L = \{w \in \Sigma^* | n_a(w) = n_b(w)\}$$

در فصل های آینده در تمرینات زیادی از این زبان استفاده می شود. گرامرهای معروف تر این زبان را در جدول مشاهده می کنید:

گرامر	زبان
$S \rightarrow aSb bSa SS \lambda$	$L = \{w \in \Sigma^* n_a(w) = n_b(w)\}$
$S \rightarrow aSbS bSaS \lambda$	
$S \rightarrow SaSb SbSa \lambda$	
$S \rightarrow aB bA \lambda$ $A \rightarrow SaS$ $B \rightarrow SbS$ $A \rightarrow a$ و $B \rightarrow b$ هم می توانند اضافه شوند.	
$S \rightarrow aB bA \lambda$ $A \rightarrow a aS bAA$ $B \rightarrow b bS aBB$	
$S \rightarrow aBS bAS \lambda$ $A \rightarrow a bAA$ $B \rightarrow b aBB$	

مثال ۸ (مهم): برای زبان L گرامر پیدا کنیم.

$$L = \{a^{2^n} \mid n \geq 0\} = \{a^1, a^2, a^4, a^8 \dots\}$$

$$S \rightarrow [Ra] \mid a$$

$$Ra \rightarrow aaR$$

$$R] \rightarrow E$$

$$aL \rightarrow La$$

$$aE \rightarrow Ea$$

$$R] \rightarrow L]$$

$$[L \rightarrow [R$$

$$[E \rightarrow \lambda$$

به طور خلاصه:

هر بار که متغیر R از چپ به راست می آید تعداد الفبای a را دو برابر میکند. سپس میتوان با L به ابتدای رشته برگشت یا با E از حلقه تکرار خارج شد. E باعث خاتمه یافتن تکرارها و حذف کروشها از ابتدا و انتها می شود.



3-

فصل اول: مفاهیم پایه

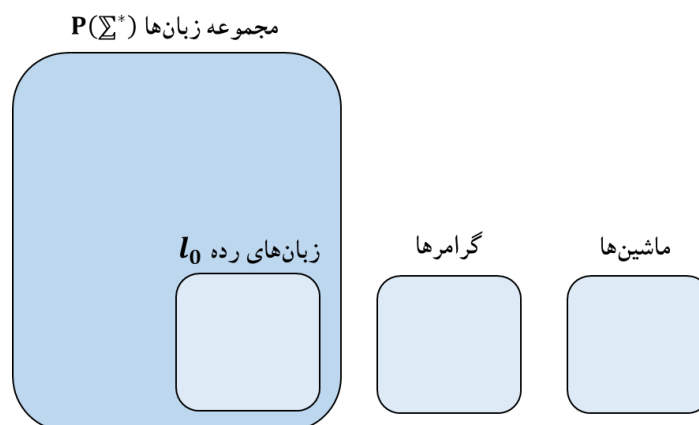
جلسه سوم بخش یک: رده بندی گرامرها _ رده بندی زبان ی دارای گرامر

مجموعه زبان ها، مجموعه گرامرها، مجموعه ماشین ها

می خواهیم کاردینال سه مجموعه را با هم مقایسه کنیم: مجموعه زبان ها، مجموعه گرامرها و مجموعه ماشین ها. مجموعه همه زبان ها ناشمار است:

$$\text{card}(P(\Sigma^*)) = 2^{N_0} = c$$

مجموعه همه گرامرها شمارای نامتناهی است. مجموعه همه ماشین ها شمارای نامتناهی است. پس شمارای زبان وجود دارد که گرامری ندارند. مجموعه زبان های دارای گرامر، شمارای نامتناهی است و به آن رده ℓ_0 می گوییم. مجموعه ماشین ها، مجموعه گرامرها، و مجموعه زبان های رده ℓ_0 ، در تناظر با هم قرار دارند. کاردینال همه آنها N_0 است.



گرامرهای بدون محدودیت (Unrestricted Grammars)

گرامر بدون محدودیت کلی ترین حالت گرامر است. قواعد تولید آن به این فرم هستند:

$\alpha \rightarrow \beta$	$\alpha \in (TUV)^+ - (T)^*$	$\beta \in (TUV)^*$
----------------------------	------------------------------	---------------------

β رشته دلخواهی از متغیرها و الفبا است. اما α باید دستکم یک متغیر داشته باشد. به بیان دیگر:
 $\alpha \in (TUV)^* V (TUV)^*$

مثال: چند نمونه از قواعد تولید در گرامر بدون محدودیت:

$$A \rightarrow aCCb$$

$$A \rightarrow \lambda$$

$$aaB \rightarrow bB$$

مثال: گرامر بدون محدودیت G تولید کننده زبان $L = \{a^{2^n} \mid n \geq 0\}$ است.

G:

$$S \rightarrow [Ra] \mid a$$

$$Ra \rightarrow aaR$$

$$R] \rightarrow E$$

$$aL \rightarrow La$$

$$aE \rightarrow Ea$$

$$R] \rightarrow L]$$

$$[L \rightarrow [R$$

$$[E \rightarrow \lambda$$

$$L = \{a^{2^n} \mid n \geq 0\} = \{a^1, a^2, a^4, a^8, \dots\}$$

گرامرهای حساس به متن (Context Sensitive Grammars)

در قواعد تولید گرامر حساس به متن شرط $|\alpha| \leq |\beta|$ را داریم:

$\alpha \rightarrow \beta$	$\alpha \in (TUV)^+ - (T)^*$	$\beta \in (TUV)^*$	$ \alpha \leq \beta $
----------------------------	------------------------------	---------------------	-------------------------

در این نوع گرامر قواعد انقباضی مانند $A \rightarrow \lambda$ و $aA \rightarrow b$ وجود ندارند. هیچ قاعده‌ای باعث کاهش اندازه رشته نمی‌شود.

عدم تولید λ :

α دستکم یک متغیر دارد. پس $|\alpha| \geq 1$ و طبق شرط $|\alpha| \leq |\beta|$ پس $1 \leq |\beta|$

یعنی β نمی‌تواند رشته تهی باشد. به عبارتی گرامر حساس به متن نمی‌تواند رشته تهی تولید کند. برای

آنکه زبان L حساس به متن باشد کافیهست $L - \{\lambda\}$ با یک گرامر حساس به متن تولید شود.

مثال: نشان دهید زبان L حساس به متن است:

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

پاسخ: کفایت برای $L - \{\lambda\}$ گرامر حساس به متن پیدا کنیم.

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

گرامرهای G_1 و G_2 این زبان را تولید می‌کنند:

G_1 :

$$S \rightarrow aSBc$$

$$cB \rightarrow Bc$$

$$SB \rightarrow b$$

$$bB \rightarrow bb$$

G_2 :

$$S \rightarrow aHbc|abc$$

$$Hb \rightarrow bH$$

$$Hc \rightarrow Pbcc$$

$$bP \rightarrow Pb$$

$$aP \rightarrow aa|aaH$$

گرامر G_1 حساس به متن نیست چون قاعده $SB \rightarrow b$ را دارد. اما گرامر G_2 حساس به متن است.

نحوه تولید $a^2b^2c^2$ را در هر دو گرامر نشان داده‌ایم:

در گرامر G_1 :

$$\begin{array}{c} S \\ \downarrow \\ aSBc \\ \downarrow \\ aaSBcBc \\ \downarrow \\ aaSBBcc \\ \downarrow \\ aabBcc \\ \downarrow \\ aabbcc \end{array}$$

در گرامر G_2 :

در گرامر G_2 :

$$\begin{array}{c} S \\ \downarrow \\ aHbc \\ \downarrow \\ abHc \\ \downarrow \\ abPbcc \\ \downarrow \\ aPbbcc \\ \downarrow \\ aabbcc \end{array}$$

گرامرهای مستقل از متن (Context Free Grammars)

در این گرامرها همه قواعد تولید، به فرم $A \rightarrow \beta$ هستند که A یک متغیر است.

$A \rightarrow \beta$	$A \in V$	$\beta \in (TUV)^*$
-----------------------	-----------	---------------------

نمونه هایی از قواعد تولید گرامر مستقل از متن:

$$S \rightarrow aABb, A \rightarrow bA, B \rightarrow b, B \rightarrow \lambda$$

مثال اول: نشان دهید زبان L_1 مستقل از متن است:

$$L_1 = \{a^n b^m c^k \mid n = m + k\}$$

پاسخ: گرامر مستقل از متن G_1 تولید کننده L_1 است پس L_1 مستقل از متن است.

G_1 :

$$S \rightarrow aSc \mid A$$

$$A \rightarrow aAb \mid \lambda$$

نحوه تولید $a^n b^m c^k$ ($n = m + k$) را در این گرامر نشان داده ایم:

$$\begin{array}{c}
 S \\
 \downarrow^* \\
 a^k S c^k \\
 \downarrow \\
 a^k A c^k \\
 \downarrow \\
 a^k a^m A b^m c^k \\
 \downarrow^* \\
 a^{k+m} b^m c^k
 \end{array}$$

$$a^{k+m} b^m c^k = a^k \underbrace{a^m b^m}_A c^k$$

مثال دوم: نشان دهید زبان L_2 مستقل از متن است:

$$L_2 = \{a^i b^j c^k d^l \mid i = j, k = 2l\}$$

پاسخ: گرامر مستقل از متن G_2 تولید کننده L_2 است پس L_2 مستقل از متن است.

$$G_2: \quad \begin{array}{l} S \rightarrow AB \\ A \rightarrow aAb \mid \lambda \\ B \rightarrow c^2 B d \mid \lambda \end{array} \quad \begin{array}{c} \underbrace{a^i b^i}_A \quad \underbrace{c^{2l} d^l}_B \end{array}$$

گرامرهای خطی (Linear Grammars)

هر گرامر مستقل از متن که در سمت چپ هر قاعده آن حداکثر یک متغیر آمده باشد، خطی است. در این گرامرها همه قواعد تولید، به این فرم هستند:

$A \rightarrow xBy$ $A \rightarrow x$	$A \in V$	$B \in V$ $x, y \in T^*$
--	-----------	-----------------------------

گرامرهای G_1 و G_2 هر دو مستقل از متن هستند اما فقط G_1 خطی است.

$$G_1: \quad \begin{array}{l} S \rightarrow aSc \mid A \\ A \rightarrow aAb \mid \lambda \end{array}$$

$$G_2: \quad \begin{array}{l} S \rightarrow AB \\ A \rightarrow aAb \mid \lambda \\ B \rightarrow c^2 B d \mid \lambda \end{array}$$

$$L(G_1) = \{a^n b^m c^k \mid n = m + k\}$$

$$L(G_2) = \{a^i b^j c^k d^l \mid i = j, k = 2l\}$$

گرامرهای منظم (نرمال) (Regular Grammars)

اگر همه قواعد در گرامر خطی به فرم $A \rightarrow xB$ و $A \rightarrow x$ باشند آن را خطی راست می‌نامیم.
اگر همه قواعد به فرم $A \rightarrow Bx$ و $A \rightarrow x$ باشند آن را خطی چپ می‌نامیم. گرامرهای خطی راست و گرامرهای خطی چپ را گرامر منظم می‌نامیم.

$A \rightarrow xB$ $A \rightarrow x$	$A \in V$	$B \in V$ $x \in T^*$
---	-----------	--------------------------

$A \rightarrow Bx$ $A \rightarrow x$	$A \in V$	$B \in V$ $x \in T^*$
---	-----------	--------------------------

مثال: نشان دهیم زبان $L = \{w \mid |w| \equiv 0 \pmod{2}\}$ دارای گرامر منظم است. ($\Sigma = \{a, b\}$)
پاسخ:

L مجموعه رشته‌های به طول زوج است. کفایت گرامر G تمام رشته‌های تولید شده با بلوک‌های به طول 2 را تولید کند. این بلوک‌ها چهارتا هستند: aa, ab, ba, bb

$G:$
 $S \rightarrow aaS \mid abS \mid baS \mid bbS \mid \lambda$

$$L(G) = L$$

نوع گرامر	قواعد تولید	سمت چپ	سمت راست
بدون محدودیت	$\alpha \rightarrow \beta$	$\alpha \in (TUV)^+ - (T)^*$	$\beta \in (TUV)^*$
حساس به متن	$\alpha \rightarrow \beta$	$\alpha \in (TUV)^+ - (T)^*$	$\beta \in (TUV)^*$ $ \alpha \leq \beta $
مستقل از متن	$A \rightarrow \beta$	$A \in V$	$\beta \in (TUV)^*$
خطی	$A \rightarrow xBy$ $A \rightarrow x$	$A \in V$	$B \in V$ $x \in T^*$
خطی راست (منظم)	$A \rightarrow xB$ $A \rightarrow x$	$A \in V$	$B \in V$ $x \in T^*$
خطی چپ (منظم)	$A \rightarrow Bx$ $A \rightarrow x$	$A \in V$	$B \in V$ $x \in T^*$

فصل اول: مفاهیم پایه

جلسه سوم بخش دو: معرفی مقدماتی ماشین ها

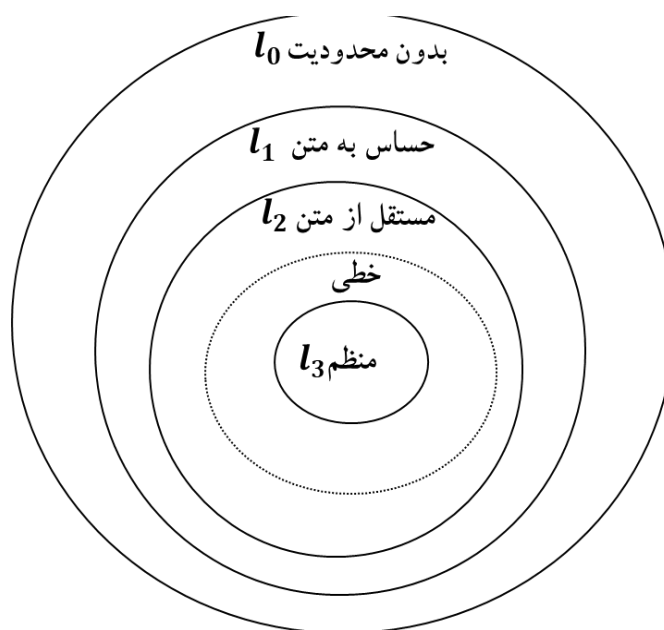
رده بندی زبان های دارای گرامر:

رده l_0 : تمام زبانهای دارای گرامر (بدون محدودیت)

رده l_1 : زبانهایی که دارای گرامر حساس به متن برای $L - \{\lambda\}$ هستند.

رده l_2 : تمام زبانهای دارای گرامر مستقل از متن.

رده l_3 : تمام زبانهای دارای گرامر منظم.



ماشین ها:

در یک نگاه کلی، هر ماشین دارای یک ابزار برای گرفتن ورودی، یک واحد کنترل برای انجام پردازش و یک راه برای نمایش خروجی است. ماشین رشته w را از نوار ورودی می خواند، واحد کنترل ضمن خواندن الفباهای رشته ورودی، می تواند از حالتی به حالت دیگر تغییر حالت بدهد، در هر حالت مجموعه ای از دستورات که همان انتقال ها هستند قابل اجراست. ماشین ممکن است یک فضای حافظه قابل خواندن و نوشتن داشته باشد که در طول انجام محاسبات از آن استفاده کند، نحوه کار واحد کنترل با مجموعه ای از حالات و توابع انتقال مشخص می شود. در پایان ماشین یک خروجی به ما خواهد داد. در تعریف رسمی و

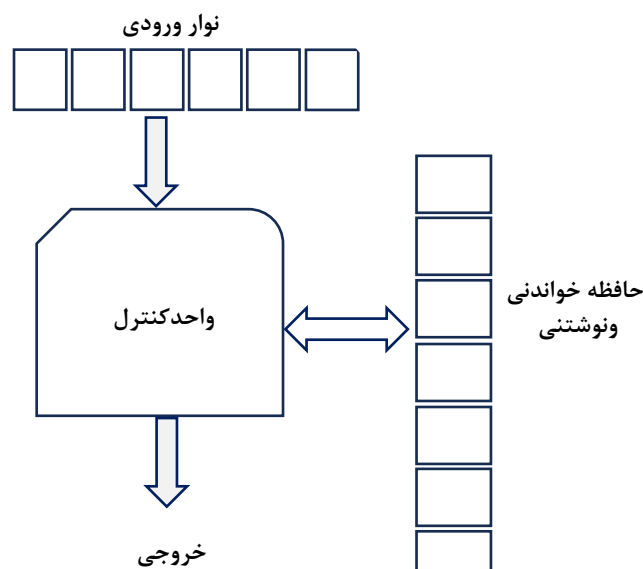
پذیرفته شده، تعداد حالات ماشین‌ها متناهی است. هرگاه از ماشین صحبت می‌کنیم به طور پیش‌فرض، تعداد حالات متناهی است. (مگر آنکه خلاف آن ذکر شود.)
ماشین‌ها به لحاظ مأموریت و کارکرد در دو دسته قرار دارند:

الف- ماشین پذیرنده:

به ازای هر رشته ورودی w ، خروجی yes یا no خواهد داشت که به معنای پذیرش یا عدم پذیرش w است. رشته‌هایی که پذیرفته می‌شوند در زبان پذیرفته شده توسط ماشین قرار دارند.

ب- ماشین تبدیل‌کننده (تراگذر-محاسبه‌گر)

برای هر رشته ورودی w ، مقدار $f(w)$ را در خروجی تولید می‌کند. تابع f را تابعی محاسبه‌پذیر توسط ماشین می‌نامیم. مثلاً ماشینی که عدد باینری $w \in (0+1)^+$ را می‌گیرد و $w+1$ را تولید می‌کند. طرح کلی ماشین‌ها را در شکل مشاهده می‌کنید



ماشین‌های پذیرنده‌ای که در فصل‌های بعد مطالعه می‌کنیم عبارتند از:

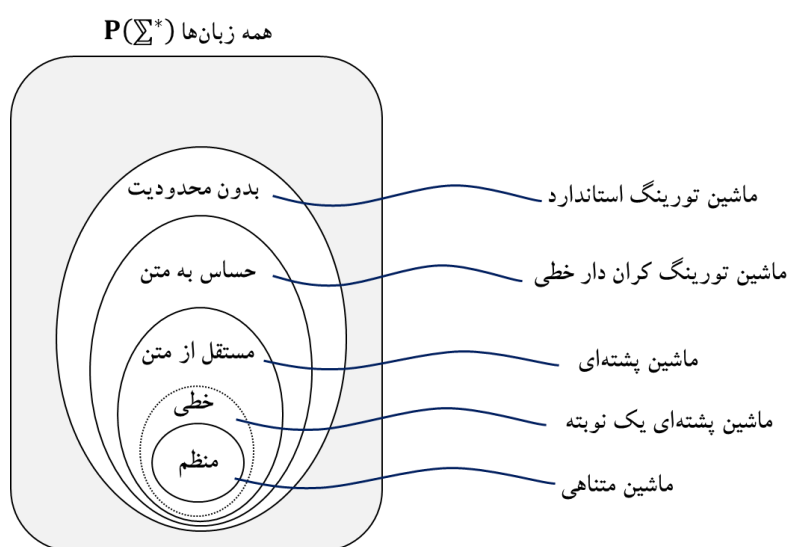
الف- ماشین متناهی.

ب- ماشین پشته‌ای.

ج- ماشین تورینگ کران دار خطی.

د- ماشین تورینگ استاندارد.

توان محاسباتی ماشین‌ها که در فصل‌های بعد مطالعه می‌کنیم:



۵۵- گرامر زیر چه زبانی را تولید می کند؟

* λ نمایانگر رشته به طول صفر است.

* در تمام گزینه ها $m, n, k, k_1, k_2 \geq 0$

$$S \rightarrow aSa \mid B \mid \lambda$$

$$Ba \rightarrow bbBaa$$

$$aB \rightarrow aaBbbb$$

$$B \rightarrow \lambda$$

$$L(G) = \{a^n b^{2n} a^{2n}\} \cup \{a^{2n} b^{3n} a^n\} \quad (۱)$$

$$L(G) = \{a^n b^{2k} a^{n+k}\} \cup \{a^{n+k} b^{3k} a^n\} \quad (۲)$$

$$L(G) = \{a^n b^{2k} a^{n+2k}\} \cup \{a^{n+2k} b^{3k} a^n\} \quad (۳)$$

$$L(G) = \{a^{n+k_1} b^m a^{n+k_2} \mid 2k_1 \leq m \leq 3k_2\} \quad (۴)$$

پاسخ: گزینه ۲

$$S \rightarrow aSa \mid B \mid \lambda$$

$$\textcircled{1} Ba \rightarrow b^2 Ba^2$$

$$\textcircled{2} aB \rightarrow a^2 Bb^3$$

$$B \rightarrow \lambda$$

$$S \xrightarrow{*} a^n S a^n \rightarrow a^n B a^n \begin{cases} \xrightarrow{\textcircled{1}*} a^n b^{2k} B a^{n+k} \rightarrow a^n b^{2k} a^{n+k} \\ \xrightarrow{\textcircled{2}*} a^{n+m} B b^{3m} a^n \rightarrow a^{n+m} b^{3m} a^n \end{cases}$$

$$\downarrow$$

$$a^n a^n$$

مبانی نظریه محاسبه:

۱۰۶- در مورد زبان‌های قابل تعریف بر روی الفباهای ناتهی A و B با حداقل دو حرف که $A \subset B$ ، کدام گزینه درست است؟

علوم کامپیوتر ۹۵

(۱) بین مجموعه‌های A^* و B^* تناظر یک به یکی وجود ندارد.

(۲) برای هر زبان $L_1 \subseteq A^*$ و هر زبان $L_2 \subseteq B^*$ داریم $L_1 \subseteq L_2$.

(۳) برای هر زبان $L_1 \subseteq B^*$ یک زبان $L_2 \subseteq A^*$ وجود دارد، که تعداد اعضای L_1 و L_2 برابر باشند.

(۴) هر زبان $L_1 \subseteq B^*$ را می‌توان به صورت اجتماع دو زبان از A^* و $(B-A)^*$ نوشت.

پایخ: ۳

$$N_0 = \text{card } N$$

$$\text{card } A^* = N_0$$

$$\text{card } B^* = N_0$$

$\Rightarrow$ تناظر یک به یکی بین A^* و B^* وجود دارد.

نادرستی گزینه ۱:

نادرستی گزینه ۲:

فرض کنیم $A = \{a, b\}$ و $B = \{a, b, c\}$.

$$L_1 = \{a, b, ab\} \subseteq A^*$$

$$L_2 = \{abc\} \subseteq B^*$$

نادرستی گزینه ۴:

$$L_1 = \{a^2, b^2, ab, abc\} \subseteq B^* \quad \text{قابل تفکیک به دو زیرمجموعه از } A^* \text{ و } (B-A)^* \text{ نیست.}$$

هو $L \subseteq A^*$ یا کاردینال منتهی دارد یا کاردینال N_0 .

هو $L \subseteq B^*$ یا کاردینال منتهی دارد یا کاردینال N_0 .

درستی گزینه ۳:

تمرین - اگر L, M و N زبان هایی با الفبای $\Sigma = \{a, b\}$ باشند کدام درست است؟ (علوم کامپیوتر ۹۱)

$$LM \cap LN \subseteq L(M \cap N) \quad (۲)$$

$$LM \cap LN = L(M \cap N) \quad (۱)$$

$$(M \cap N)L = L(M \cap N) \quad (۴)$$

$$L(M \cap N) \subseteq LM \cap LN \quad (۳)$$

پاسخ: گزینه ۳

$$\begin{aligned} (M \cap N) \subseteq M & \Rightarrow L(M \cap N) \subseteq LM \cap LN \\ (M \cap N) \subseteq N & \end{aligned}$$

در حالت کلی تساوی برقرار نیست مثلاً:

$$\begin{aligned} L &= \{a, a^2\} \\ M &= \{a, b^2\} \\ N &= \{a^2, b\} \end{aligned} \Rightarrow \begin{aligned} L(M \cap N) &= L\emptyset = \emptyset \\ LM \cap LN &= \{a^2, ab^2, a^3, a^2b^2\} \cap \{a^3, ab, a^4, a^2b\} = \{a^3\} \end{aligned}$$

فصل دوم: ماشین متناهی

جلسه چهارم بخش یک: ماشین متناهی قطعی (DFA)

ماشین متناهی قطعی (DFA)

هر ماشین متناهی قطعی یک پنج تایی $M = (Q, \Sigma, \delta, q_0, F)$ است. Q یک مجموعه متناهی است که حالات ماشین نام دارند. یکی از حالات q_0 است و تعدادی دلخواهی از آنها را حالات نهایی (حالات پذیرش) در نظر می‌گیریم. F مجموعه حالات نهایی است. تابع انتقال $\delta: Q \times \Sigma \rightarrow Q$ به ازای هر حالت $q_i \in Q$ و هر الفبای $a \in \Sigma$ یک حالت $q_j \in Q$ را به عنوان نتیجه انتقال مشخص می‌کند: $\delta(q_i, a) = q_j$. از یک گراف برای نمایش حالات و انتقال‌ها استفاده می‌کنیم.

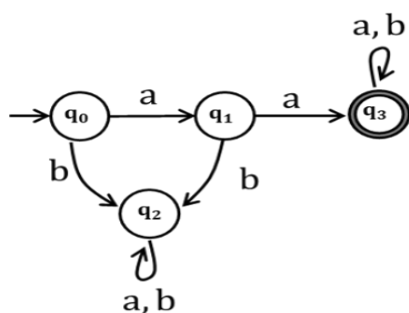


ماشین متناهی قطعی رشته $w = a_1 a_2 \dots a_m$ را از چپ به راست مصرف می‌کند و در پایان اگر به یک حالت پذیرش رسید رشته را می‌پذیرد. مجموعه رشته‌های پذیرفته شده توسط ماشین را زبان $L(M)$ می‌نامیم.

مثال: (تابع انتقال در ماشین متناهی قطعی)

به تابع انتقال در جدول زیر و گراف نمایش ماشین M توجه کنید. برای هر حالت و هر الفبا دقیقاً یک حالت به دست می‌آید.

مثال 1: به تابع انتقال در جدول زیر و در شکل توجه کنید. در هر حالت برای هر الفبا دقیقاً یک انتقال داریم. این ماشین dfa است. زبان $L(M)$ شامل چه رشته‌هایی است؟ (چه رشته‌هایی توسط M پذیرفته می‌شوند؟)



δ	a	b
q₀	q ₁	q ₂
q₁	q ₃	q ₂
q₂	q ₂	q ₂
q₃	q ₃	q ₃

جواب: فقط رشته‌هایی که با a^2 آغاز شوند پذیرفته می‌شوند.

$$L(M) = \{a^2v \mid v \in \Sigma^*\}$$

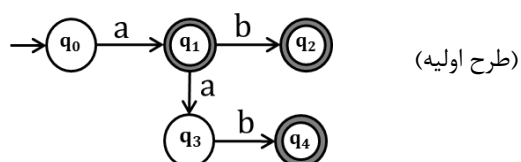
یافتن (DFA) پذیرنده زبان L .

(مسائلی از Peter Linz)

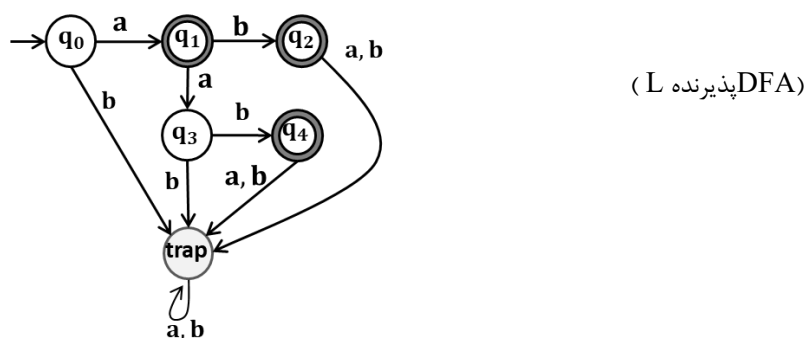
مثال 2: برای زبان متناهی L ماشین پذیرنده قطعی بیابید.

$$L = \{a, ab, aab\}$$

پاسخ: برای زبان متناهی، ابتدا بدون آنکه دور ایجاد شود، هر کدام از رشته های عضو L را به حالت پذیرش می‌رسانیم:



طرح اولیه یک ماشین به دست می‌آید. حالا تمام انتقال‌های مشخص نشده را به حالت تله هدایت می‌کنیم. (حالتی غیرپذیرش که امکان خروج از آن وجود ندارد).

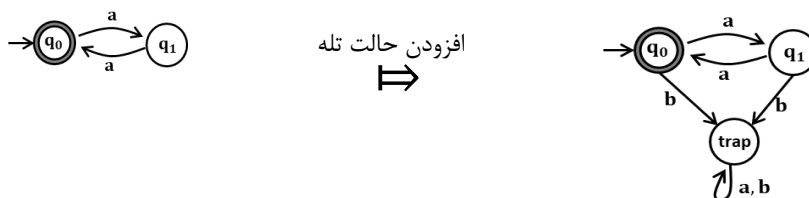


برای زبان متناهی، نباید ماشین DFA دارای دور یا طوقی باشد که به حالت پذیرش دسترسی دارد. در صورت وجود چنین دوری، تعداد رشته هایی که پذیرفته می‌شوند، بینهایت است.

مثال 3: برای زبان L ماشین پذیرنده قطعی بیابید.

$$L = \{\lambda, a^2, a^4, a^6, \dots\}$$

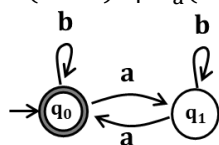
برای آنکه فقط تعداد زوج پذیرفته شود یک حلقه به طول 2 با الفبای ایجاد می‌کنیم. یک حالت پذیرش و یک حالت غیرپذیرش در این حلقه داریم. سپس انتقال های دیگر را تعریف می‌کنیم. در صورت وجود الفبای b در هر کجای رشته w، آن رشته پذیرفته نمی‌شود پس با b وارد حالت تله می‌شویم.



توجه: همانطور که در این مثال دیدیم، برای پذیرفته شدن λ توسط DFA باید q_0 حالت پذیرش باشد.
 $\lambda \in L(M) \Leftrightarrow q_0 \in F$

مثال 4: برای زبان L ماشین پذیرنده قطعی بسازید.

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



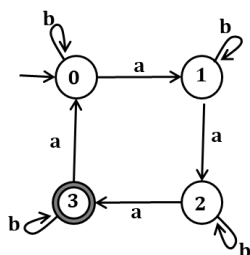
هرگاه وجود یا عدم وجود همچنین تعداد یک الفبا در پذیرش رشته w بی‌تاثیر باشد انتقال‌های آن الفبا را به صورت طوق از هر حالت به همان حالت تعریف می‌شوند. در این زبان الفبای b در عضویت یا عدم عضویت w بی‌تاثیر است.

معرفی نماد:

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

مثال 5: برای زبان L ماشین پذیرنده قطعی بسازید.

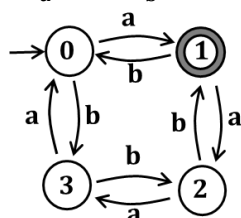
$$L = \{w \in (a+b)^* \mid n_a(w) \bmod 4 = 3\}$$



برای نمایش باقیمانده به پیمانه 4 یک حلقه به طول 4 با انتقال a می‌سازیم. الفبای b در پذیرش بی‌تاثیر است پس با انتقال b از هر حالت به همان حالت برمی‌گردیم.

مثال 6: برای زبان L ماشین پذیرنده قطعی بیابید.

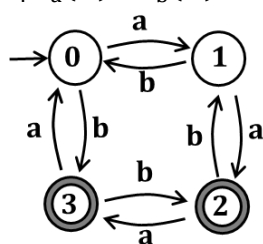
$$L = \{w \mid n_a(w) - n_b(w) \bmod 4 = 1\}$$



به $n_a(w) - n_b(w)$ به پیمانه 4 مهم است. به دلیل تفاضل، انتقال های a و b خلاف جهت هم تعریف می شوند و برای پیمانه 4 یک حلقه به طول 4 می سازیم.

مثال 7: برای زبان L ماشین پذیرنده قطعی بیابید.

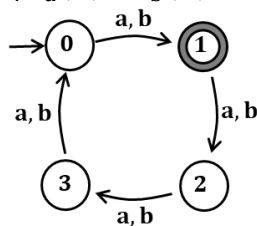
$$L = \{w \mid n_a(w) - n_b(w) \bmod 4 \geq 2\}$$



پذیرش را در حالتی قرار می دهیم که $n_a(w) - n_b(w)$ به پیمانه 4، برابر با 2 یا 3 باشد.

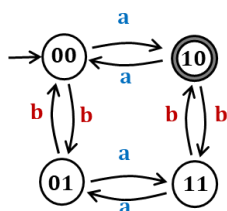
توجه: در مثال های قبل اگر $n_a(w) + n_b(w)$ مهم بود، انتقال های a و b هم جهت رسم می شدند. اگر بخواهیم برای این زبان ماشین پذیرنده قطعی بیابیم:

$$L = \{w \mid n_a(w) + n_b(w) \bmod 4 = 1\}$$



انتقال ها هم جهت هستند. حالت پذیرش با توجه به شرط خواسته شده تعیین می شود.

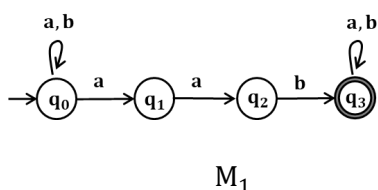
مثال 8: برای زبان L ماشین پذیرنده قطعی بیابید.
 $L = \{w \mid n_a(w) \bmod 2 = 1 \text{ and } n_b(w) \bmod 2 = 0\}$



برای هر یک از الفباهای a و b یک دور به طول 2 لازم داریم. این دورها به شکل مستقل از هم عمل می کنند. در هر حالت، اولین رقم، همنهشتی a و دومی همنهشتی b را نشان می دهد.

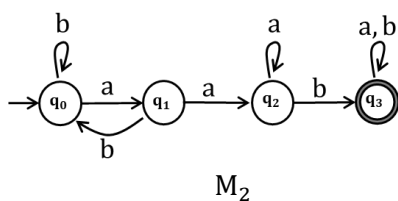
مثال 9: فرض کنیم زبان L شامل تمام رشته هایی است که زیر رشته aab دارند. ماشین پذیرنده قطعی L را بیابید.

$$L = L(\Sigma^* aab \Sigma^*)$$



با توجه به عبارت منظم $\Sigma^* aab \Sigma^*$ ماشین M_1 پذیرنده این زبان است اما این ماشین قطعی نیست. مثلاً از q_0 دو انتقال a داریم. این ماشین غیرقطعی است. ماشین های غیر قطعی را در جلسات بعد مطالعه می کنیم.

ماشین M_2 یک پذیرنده DFA این زبان است.

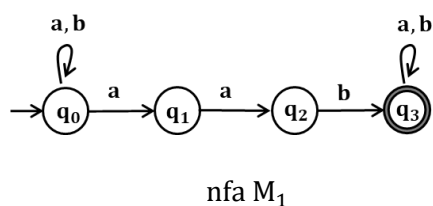


کوچکترین رشته عضو این زبان که با آن به حالت پذیرش می رسیم aab است.

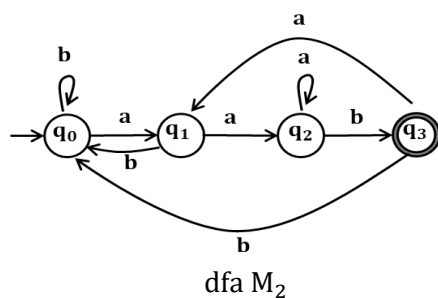
فصل دوم: ماشین متناهی

جلسه چهارم بخش دوم: ماشین متناهی قطعی (DFA)

مثال 10: فرض کنیم L مجموعه تمام رشته هایی است که پسوند aab دارند. (به aab ختم می شوند) یک ماشین متناهی قطعی پذیرنده برای L بیابید.



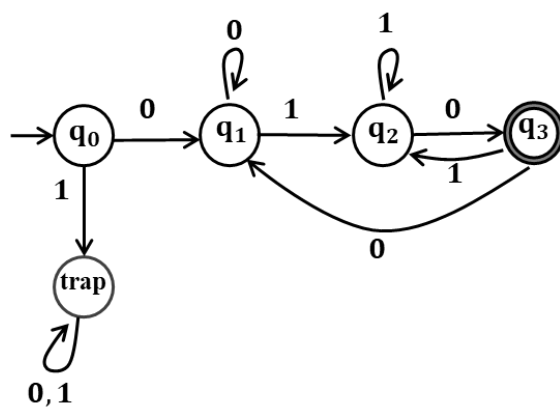
پاسخ: رشته های این زبان به فرم $\Sigma^* aab$ هستند. ماشین غیر قطعی M_1 پذیرنده این زبان است. برای یافتن ماشین قطعی ابتدا با کوچکترین رشته عضو L مسیری از حالت اولیه تا حالت پذیرش در نظر می گیریم. سپس سایر انتقال ها را به شکل مناسب تعریف می کنیم.



ماشین قطعی M_2 پذیرنده این زبان است.

مثال 11: فرض کنیم $\Sigma = \{0,1\}$. زبان L شامل تمام رشته هایی است که با 0 آغاز شده و به 10 ختم می شوند. پذیرنده dfa زبان L را طرح کنید.

پاسخ: رشته های مورد نظر به فرم $0\{0,1\}^*10$ هستند. کوچکترین رشته ای که پذیرفته می شود 010 است. پس یک مسیر ساده بدون دور از حالت اولیه به حالت پذیرش با رشته 010 خواهیم داشت. هر رشته ای با 1 شروع شود وارد حالت تله می شود. رشته ای که پذیرفته می شود اگر در ادامه الفبای 1 داشته باشد به برمی گردد و برای پذیرش مجدد به 0 نیاز دارد. به همین ترتیب سایر انتقال ها به دست می آیند.

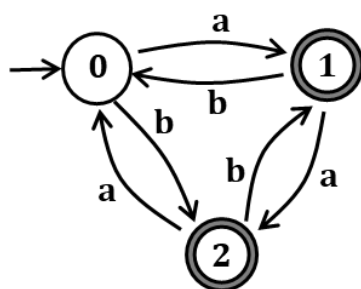


dfa M

مثال 12: برای زبان L ماشین پذیرنده قطعی بیابید.

$$L = \{w \mid n_a(w) + 2n_b(w) \bmod 3 \geq 1\}$$

پاسخ:



dfa M

باقیمانده ها به پیمانه 3 عبارتند از 0 و 1 و 2 که یک دور به طول 3 می سازند. در محاسبات به پیمانه 3. عدد k با $k - 3$ برابر است.

پس $2 \equiv -1$ است. یعنی $n_a(w) + 2n_b(w)$ با $n_a(w) - n_b(w)$ برابر است. پس تفاضل $n_a(w) - n_b(w)$ پس با الفبای a رو به جلو و با الفبای b رو به عقب حرکت می کنیم.

همچنین می توان چنین فرض کرد که ضریب $+2n_b(w)$ یعنی هر الفبای b ما را دو گام به جلو ببرد که نتیجه با شکل بالا یکسان است.

مثال 13: یک نمونه مهم

برای زبان L ماشین پذیرنده قطعی بیابید.

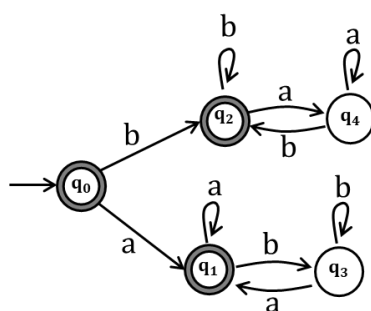
$$\Sigma = \{a, b\}$$

$$L = \{w \mid n_{ab}(w) = n_{ba}(w)\}$$

پاسخ: وقتی مجموعه الفبا $\{a, b\}$ است:

مساوی بودن تعداد ab ها و ba ها معادل است با این که اگر رشته ای با a شروع شده به a ختم شود و اگر با b شروع شده به b ختم شود. مثل آن است که دو طبقه a و b داشته باشیم و هر ab به معنای رفتن از طبقه a به طبقه b باشد و بالعکس ba به معنای رفتن از b به a باشد. وقتی تعداد این تغییرات برابر است یعنی از هر طبقه ای شروع کنیم در پایان نیز در همان طبقه باشیم.

$$L = \{w \mid n_{ab}(w) = n_{ba}(w)\} = a\Sigma^*a + b\Sigma^*b$$



dfa M

مثال 14: (PETER LINZ) برای زبان L و L^2 ماشین پذیرنده قطعی بیابید.

برای زبان L ماشین پذیرنده قطعی بیابید.

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

$$L = \{w \mid n_{ab}(w) = n_{ba}(w)\}$$

پاسخ: وقتی مجموعه الفبا $\{a, b\}$ است:

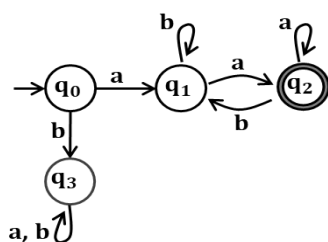
مساوی بودن تعداد ab ها و ba ها معادل است با این که اگر رشته ای با a شروع شده به a ختم شود و اگر با b شروع شده به b ختم شود. مثل آن است که دو طبقه a و b داشته باشیم و هر ab به معنای رفتن از طبقه a به طبقه b باشد و بالعکس ba به معنای رفتن از b به a باشد.

وقتی تعداد این تغییرات برابر است یعنی از هر طبقه ای شروع کنیم در پایان نیز در همان طبقه باشیم.

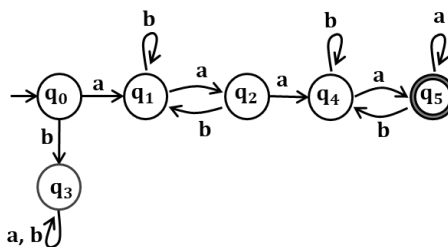
$$L = \{w \mid n_{ab}(w) = n_{ba}(w)\} = a\Sigma^*a + b\Sigma^*b$$

$$L = \{awa : w \in \{a,b\}^*\}$$

$$L^2 = \{aw_1aaw_2a : w_1, w_2 \in \{a,b\}^*\}$$



dfa M



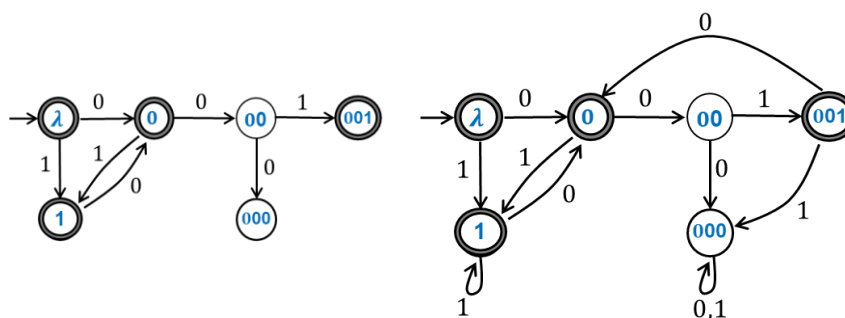
dfa M

مثال 15: (PETER LINZ) برای زبان L ماشین پذیرنده قطعی بیابید.

رشته‌هایی عضو این زبان هستند که: هر جا زیررشته 00 در آنها ظاهر شد بلافاصله بعد از آن الفبای 1 آمده باشد.

(L:) Every 00 is followed immediately by a 1

پاسخ: یک طرح پایه داریم و سپس انتقال‌ها را کامل می‌کنیم:



تکمیل انتقال‌ها

طرح اولیه

dfa M

فصل دوم: ماشین متناهی

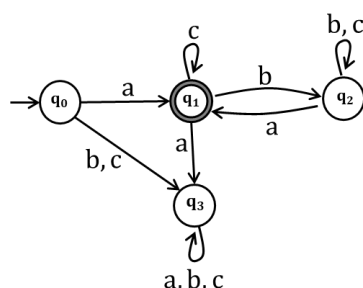
جلسه پنجم بخش یک: ماشین متناهی قطعی (DFA) و غیرقطعی (NFA)

نماد δ^* در ماشین قطعی

برای نمایش نتیجه انتقال با مصرف رشته ای از چند الفبا، از علامت δ^* استفاده می‌کنیم که به معنای چندبار استفاده از δ است. برای آنکه رشته $a_0 a_1 \dots a_m$ مصرف شود ابتدا باید a_0 مصرف شود سپس نوبت به سایر الفبا می‌رسد یعنی

$$\delta^*(q_0, a_0 a_1 \dots a_m) = \delta^*(\delta(q_0, a_0), a_1 \dots a_m)$$

تابع انتقال در DFA برای λ تعریف یعنی $\delta(q_i, \lambda)$ نداریم اما می‌توانیم نتیجه مصرف صفر الفبا را با استفاده از δ^* به این صورت بنویسیم: $\delta^*(q_i, \lambda) = q_i$ زیرا در DFA بدون مصرف الفبا، تغییر حالت رخ نمی‌دهد و در q_i باقی می‌مانیم.



مثال‌هایی از δ^* در DFA مقابل:

$$\delta^*(q_1, cb^2ab) = q_2$$

$$\delta^*(q_0, acba^2) = q_3$$

زبان منظم:

اولین تعریف از زبان منظم را به این صورت می‌پذیریم:

زبان L منظم است $\Leftrightarrow$ ماشین پذیرنده متناهی قطعی M موجود باشد که $L(M) = L$.

بنابراین برای اثبات منظم بودن یک زبان کفایت یک DFA پذیرنده برای آن پیدا کنیم. با نمونه‌هایی از زبان‌های دارای پذیرنده در قسمت قبل آشنا شدیم. اکنون به ماشین متناهی غیرقطعی و تفاوت آن با ماشین متناهی قطعی توجه کنید.

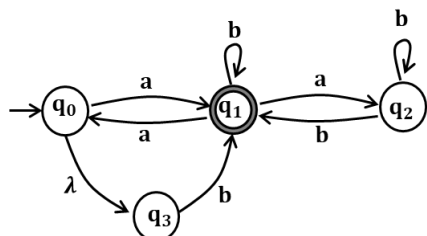
ماشین متناهی غیرقطعی (NFA)

ماشین متناهی غیرقطعی نیز یک پنج تایی $M = (Q, \Sigma, \delta, q_0, F)$ است. مجموعه متناهی Q از حالات را دارد. q_0 حالت اولیه است. $F \subseteq Q$ مجموعه حالات نهایی (پذیرش) است. تفاوت NFA با DFA در دو چیز است:

1- تابع انتقال $\delta: Q \times (\Sigma \cup \{\lambda\}) \rightarrow P(Q)$ علاوه بر الفبا، برای λ نیز تعریف می‌شود. به عبارتی ممکن است از حالت q_i بدون مصرف الفبا به حالت q_j برود. این نوع انتقال را انتقال λ می‌نامیم.

2- تابع انتقال $\delta: Q \times (\Sigma \cup \{\lambda\}) \rightarrow P(Q)$ به هر جفت (q_i, a) یک زیرمجموعه از Q را نظیر می‌کند یعنی حاصل $\delta(q_i, a)$ می‌تواند تهی، یک حالت، یا چند حالت باشد.

مثال: در ماشین متناهی غیر قطعی (NFA) زیر به محاسبات انجام شده توجه کنید.



$$\delta^*(q_3, a) = \{\}$$

$$\delta^*(q_0, \lambda) = \{q_0, q_3\}$$

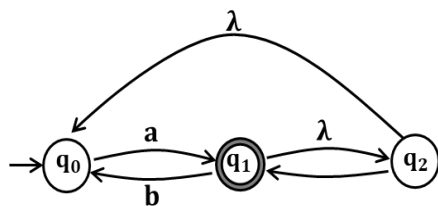
$$\delta^*(q_0, b) = \{q_1\}$$

$$\delta^*(q_1, a) = \{q_0, q_2, q_3\}$$

در آخرین مورد توجه کنید که رشته a می‌تواند به صورت‌های $a = \lambda a = a\lambda = \lambda a\lambda$ مصرف شود.

تمرین: (Peter Linz)

حاصل $\delta^*(q_1, a)$ در این ماشین کدامست؟



پاسخ: رشته a می‌تواند به صورت‌های

$$a = \lambda a = a\lambda = \lambda a\lambda = \lambda^m a \lambda^k$$

مصرف شود. در نتیجه:

$$\delta^*(q_1, a) = \{q_0, q_2, q_3\}$$

بستار λ برای حالت q :

مجموعه تمام حالتی که با شروع از q بدون مصرف الفبا فقط با انتقال λ در دسترس هستند را بستار q می‌نامیم. این مجموعه را با $\lambda - \text{closure}(q)$ نشان می‌دهیم.

مثال:

$$\lambda - \text{closure}(q_0) = \{q_0, q_3, q_4\}$$

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

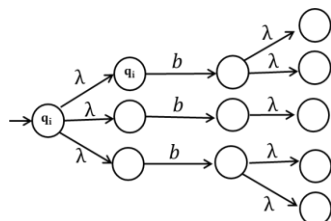
$$\lambda - \text{closure}(q_2) = \{q_2\}$$

$$\lambda - \text{closure}(q_3) = \{q_3, q_4\}$$

$$\lambda - \text{closure}(q_4) = \{q_4\}$$

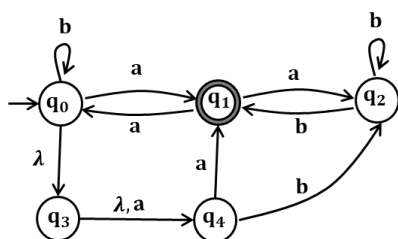
محاسبه $\delta^*(q_i, b)$ با استفاده از بستار:

$$\delta^*(q_i, b) = \lambda - \text{closure}(\delta(\lambda - \text{closure}(q_i), b))$$



شکل مقابل نحوه عملکرد این فرمول را توضیح می‌دهد. قرار داشتن در q_i به معنای قرار داشتن در تمام حالات بستار q_i است. حالا انتقال b را انجام می‌دهیم. در مرحله آخر باید از تمام حالات به دست آمده بستار بگیریم.

مثال: در این ماشین غیرقطعی حاصل $\delta^*(q_0, b)$ را بیابید.



بستار q_0 مجموعه $\{q_0, q_3, q_4\}$ است. از این حالات با الفبای b به q_0, q_2 انتقال وجود دارد. حالا بستار q_0 و q_2 روی هم مجموعه توجه $\{q_0, q_3, q_4, q_2\}$ است.

$$\delta^*(q_0, b) = \{q_0, q_3, q_4, q_2\}$$

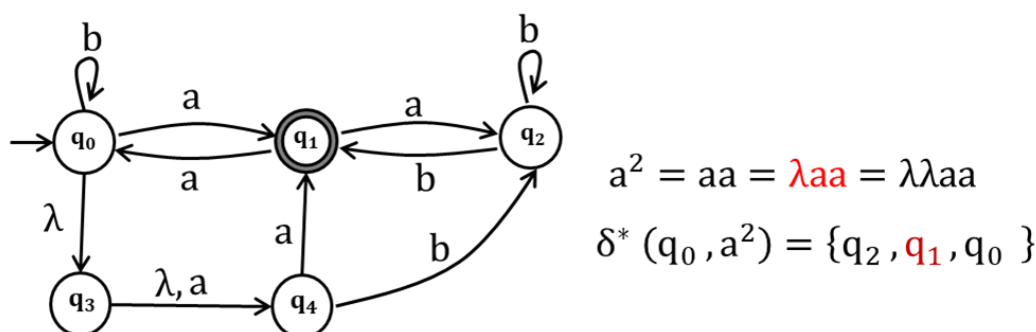
پذیرش رشته در ماشین متناهی غیر قطعی:

شرط پذیرش در DFA چنین بود: $w \in L(M) \Leftrightarrow \delta^*(q_i, w) \in F$

وقتی ماشین NFA رشته w را مصرف می‌کند، یک زیرمجموعه از Q به دست می‌آید. به هر کدام از روشهای مصرف w یک $walk$ می‌گوییم. اگر دستکم یکی از حالات به دست آمده حالت پذیرش (پایانی) باشد، w مورد پذیرش قرار می‌گیرد.

$$w \in L(M) \Leftrightarrow \delta^*(q_i, w) \cap F \neq \phi$$

مثال: در ماشین زیر رشته a^2 پذیرفته می‌شود زیرا حداقل یکی از $walk$ ها به حالت پذیرش می‌رسد. کفایت آن را به صورت λaa مصرف کنید.



تبدیل NFA به DFA معادل:

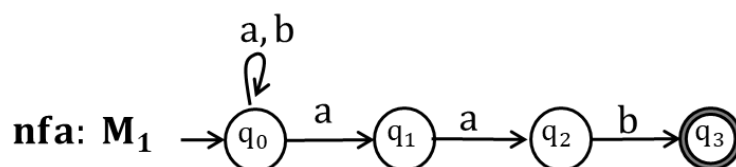
هر NFA دارای DFA معادل است. الگوریتمی وجود دارد که در چند مرحله متناهی یک NFA را به DFA معادل تبدیل می‌کند. وقتی می‌خواهیم برای زبان منظم L ماشین پذیرنده متناهی طراحی کنیم، معمولاً یافتن NFA ساده‌تر است زیرا امکانات بیشتری به ما می‌دهد. سپس می‌توانیم با طی چند مرحله آن را به DFA تبدیل کنیم.

مثال ۱: (مثالی از جلسه قبل)

برای زبان $L = \sum^* aab$ ابتدا ماشین پذیرنده متناهی غیرقطعی بیابید. سپس ماشین قطعی معادل با آن را بیابید.

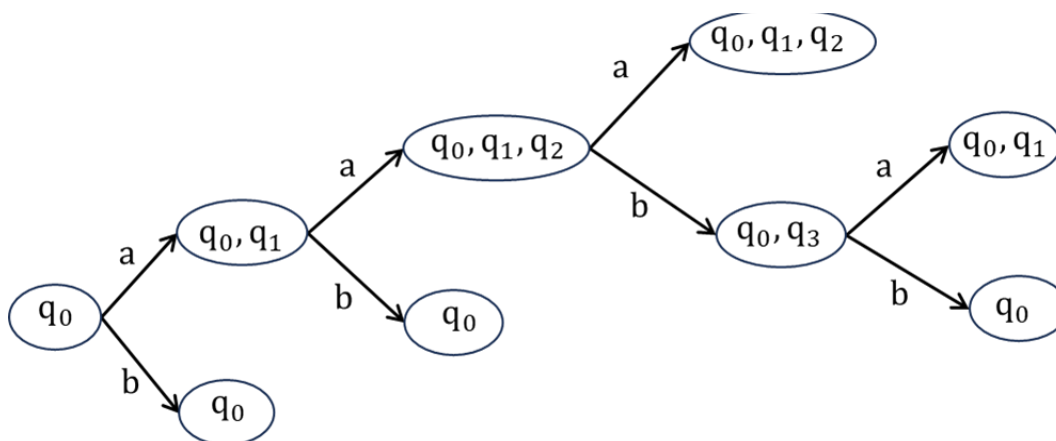
گام اول: طراحی ماشین پذیرنده متناهی غیرقطعی

$$L = (a + b)^* aab$$

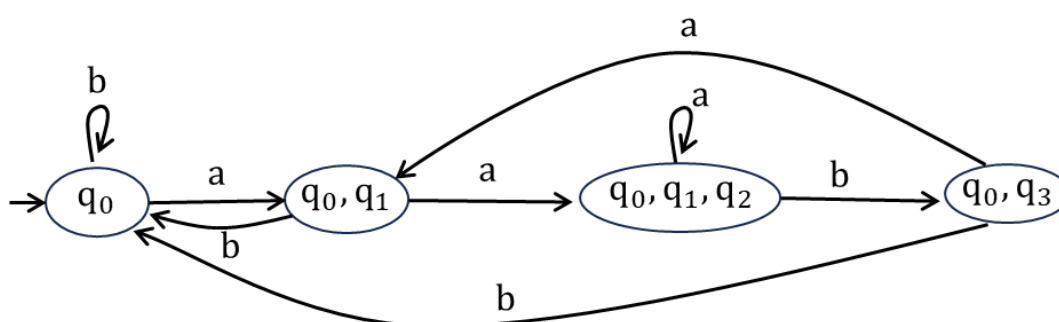


گام دوم: رسم درخت انتقال با شروع از حالت اولیه

از مجموعه $\{q_0\}$ شروع می‌کنیم. هر بار حاصل انتقال با هر الفبا را محاسبه می‌کنیم. هر گره از درخت یک زیرمجموعه از حالات است. گره‌های تکراری را بسط نمی‌دهیم. تا وقتی درخت را ادامه می‌دهیم که گره جدیدی تولید نشود و همه برگ‌ها تکراری باشند.

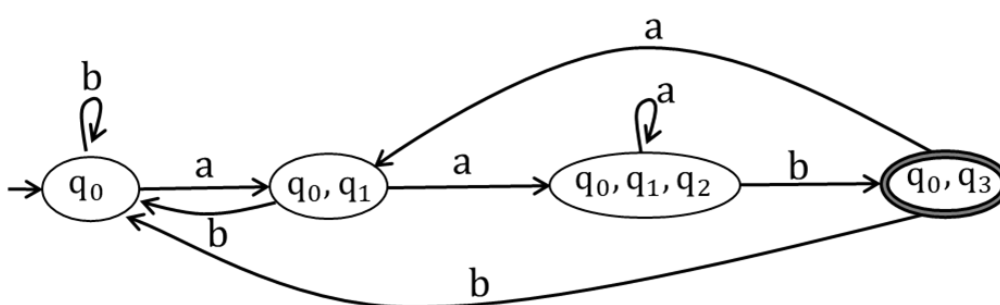


گام سوم: رسم ماشین پذیرنده متناهی قطعی:
هر گره یک حالت از DFA است. $\{q_0\}$ حالت اولیه است.



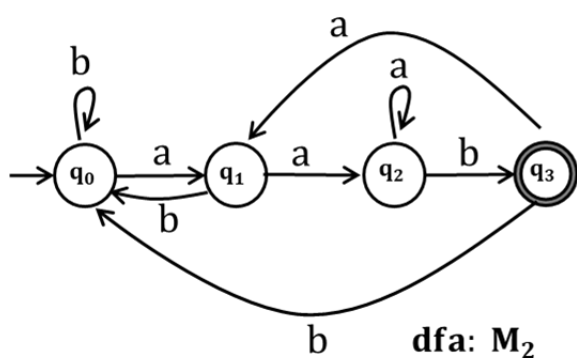
dfa: M_2

گام چهارم: تعیین حالات پذیرش (M_2 پایانی)
هر حالت از M_2 که حداقل شامل یک حالت پذیرش M_1 باشد، حالت پذیرش ماشین dfa است. همچنین اگر می‌خواستیم λ پذیرفته شود باید $\{q_0\}$ حالت پذیرش می‌بود. در این مثال q_3 حالت پذیرش ماشین nfa بود پس $\{q_0, q_3\}$ حالت پذیرش dfa است.



dfa: M_2

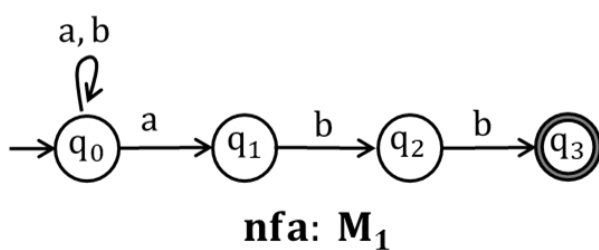
در پایان: با نامگذاری مجدد می توان M_2 را ساده تر نمایش داد:



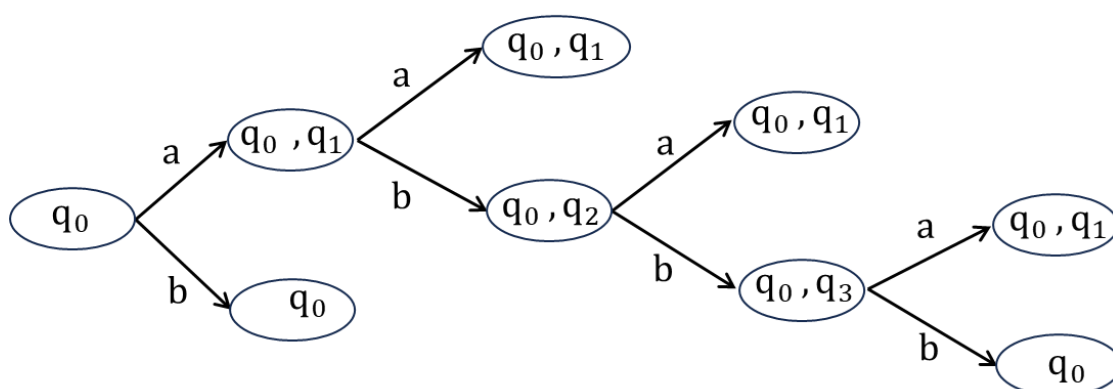
مثال ۲: برای زبان $L = \Sigma^*abb$ ابتدا ماشین پذیرنده متناهی غیرقطعی و سپس ماشین معادل قطعی بیابید.

گام اول: طراحی ماشین پذیرنده متناهی غیرقطعی

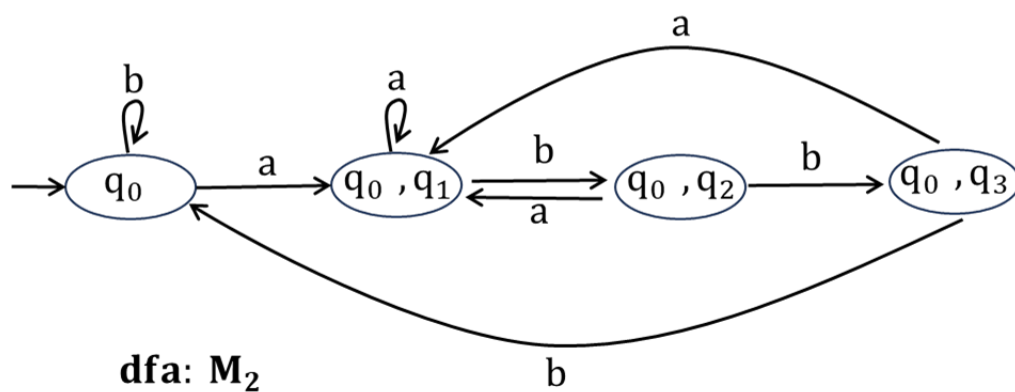
$$L = (a + b)^*abb$$



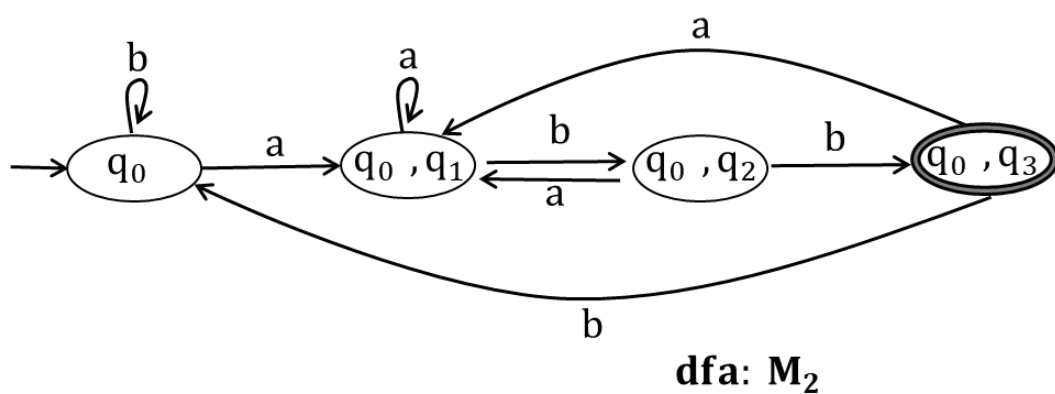
گام دوم: رسم درخت انتقال با شروع از حالت اولیه



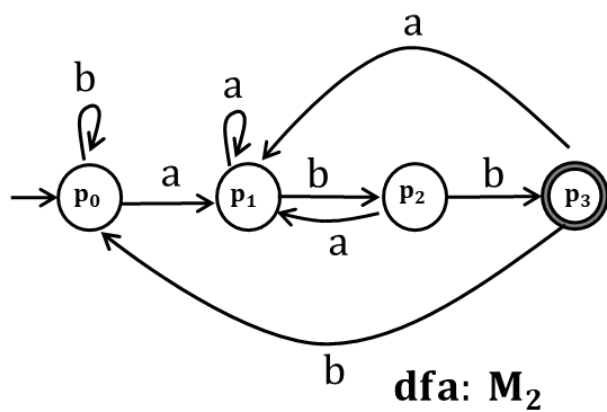
گام سوم: رسم ماشین پذیرنده متناهی قطعی:



گام چهارم: تعیین حالات پذیرش (پایانی) در M_2



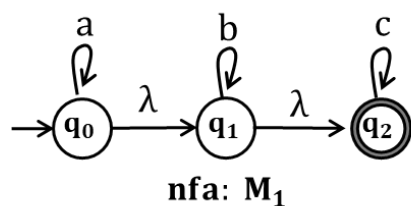
با نامگذاری مجدد می‌توان M_2 را ساده‌تر نمایش داد:



مثال ۳:

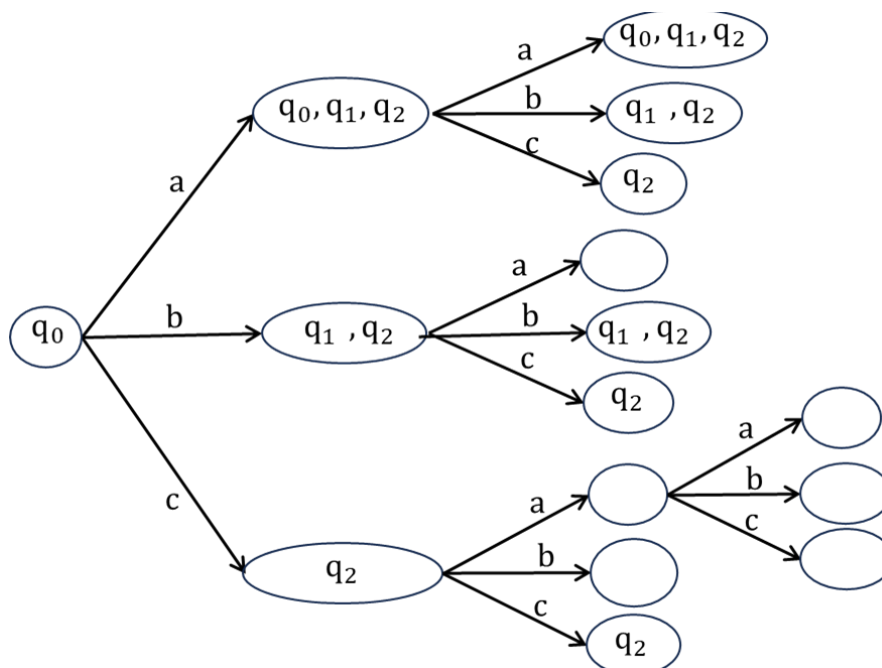
زبان L مجموعه همه رشته های به فرم $a^*b^*c^*$ است. برای L ماشین غیرقطعی پذیرنده یافته و آن را به معادل قطعی تبدیل کنید.

گام اول: طراحی ماشین پذیرنده متناهی غیرقطعی



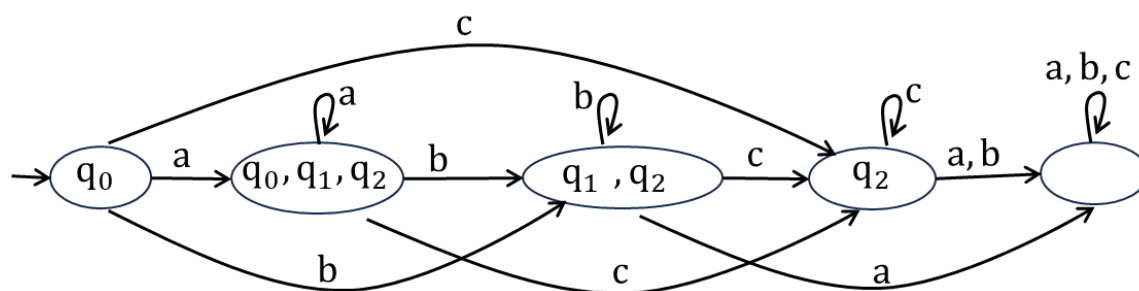
توجه کنید: در nfa با اینکه λ عضو L است ناچار نشدیم q_0 را حالت پذیرش بگیریم.

گام دوم: رسم درخت انتقال با شروع از حالت اولیه



(مجموعه تهی یک حالت تله در dfa می شود.)

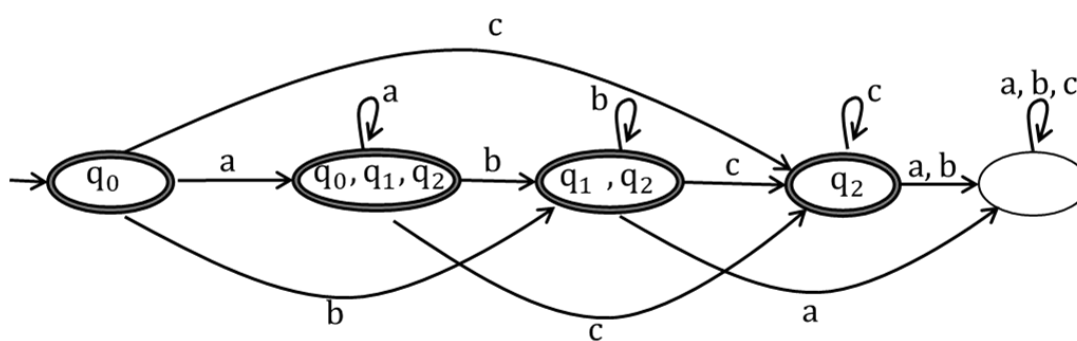
گام سوم: رسم ماشین پذیرنده متناهی قطعی:



dfa: M_2

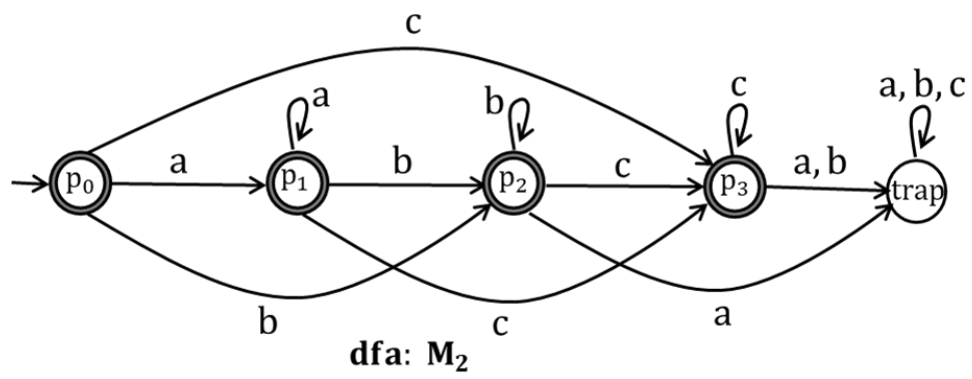
گام چهارم: تعیین حالات پذیرش (پایانی) M_2

در dfa: چون $\lambda \in L$ پس $\{q_0\}$ باید حالت پذیرش باشد. همچنین تمام حالاتی که شامل q_2 هستند حالت پذیرش محسوب می شوند.



dfa: M_2

با نامگذاری مجدد می توان M_2 را ساده تر نمایش داد:



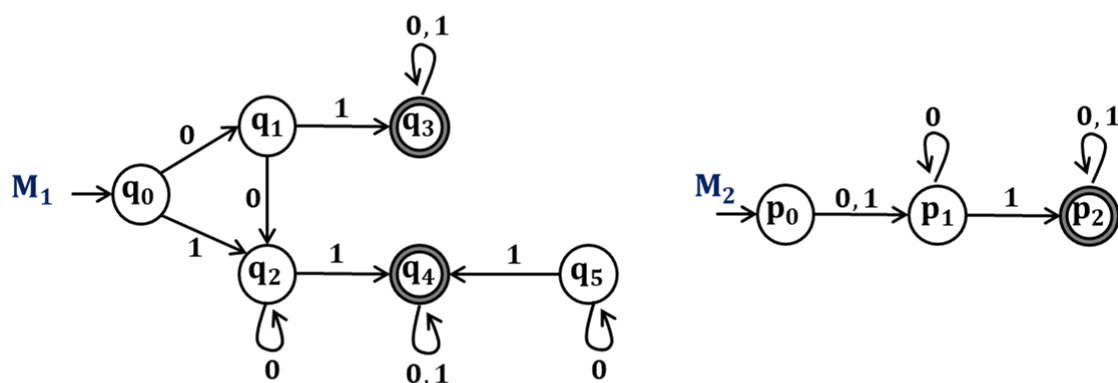
فصل دوم: ماشین متناهی

جلسه پنجم بخش دوم :

کاهش حالات ماشین متناهی قطعی - DFA کمینه

مثال (Peter Linz)

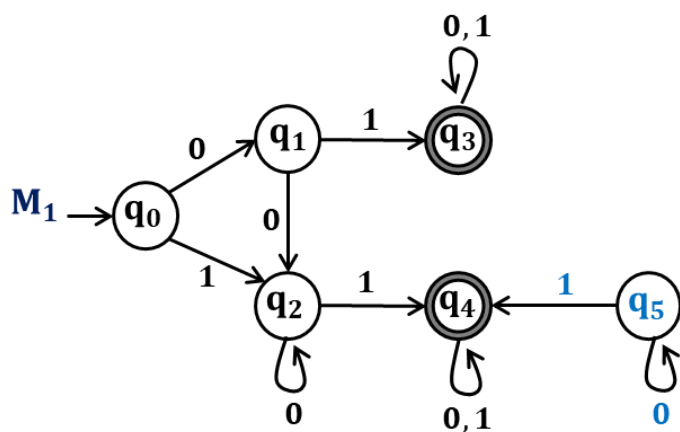
دو ماشین متناهی قطعی، با تعداد حالات مختلف، ممکن است با هم معادل باشند.
برای مثال ماشین‌های قطعی M_1 و M_2 معادلند زیرا $L(M_1) = L(M_2)$.



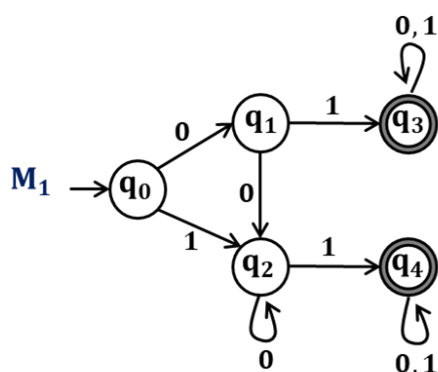
$$L(M_1) = L(M_2) = 00^*1\Sigma^* + 10^*1\Sigma^* = (0 + 1) 0^*1\Sigma^*$$

پس تعدادی از حالات ماشین M_1 قابل کاهش یا حذف هستند.

حالت q_5 زائد است زیرا غیرقابل دسترس است. از q_0 به آن مسیری وجود ندارد. بنابراین قابل حذف است.



حالات تشخیص ناپذیر (ادغام پذیر) (Indistinguishable states)



از بین حالات باقی مانده برخی جفت حالات غیر قابل تشخیص (غیر قابل تمایز) هستند. مثلاً به q_1 و q_2 توجه کنید. وقتی رشته های مختلف را با شروع از q_1 و شروع از q_2 بررسی می کنیم، متوجه می شویم هرگاه $\delta^*(q_1, w)$ حالت پذیرش باشد $\delta^*(q_2, w)$ نیز حالت پذیرش است. هرگاه $\delta^*(q_1, w)$ حالت پذیرش نباشد $\delta^*(q_2, w)$ نیز حالت پذیرش نیست. به عبارتی:

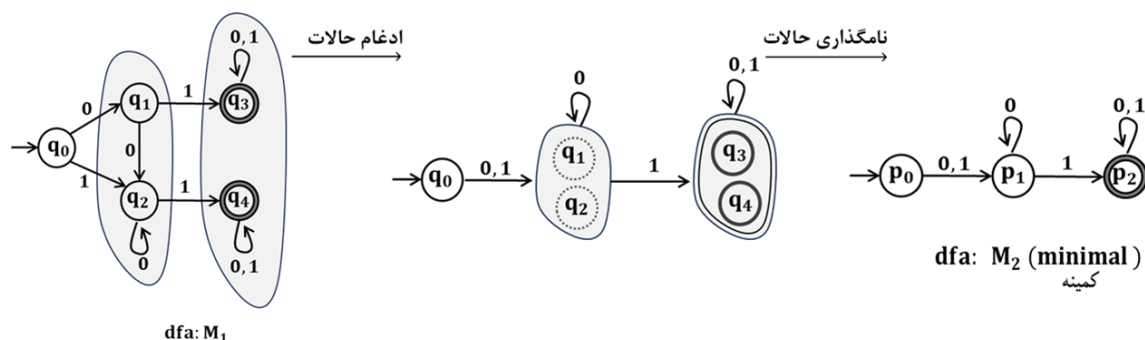
$$\delta^*(q_1, w) \in F \Leftrightarrow \delta^*(q_2, w) \in F$$

$$\delta^*(q_1, w) \notin F \Leftrightarrow \delta^*(q_2, w) \notin F$$

می گوئیم این دو حالت تشخیص ناپذیر (ادغام پذیر) هستند. ادغام پذیری یک رابطه هم ارزی است.

در این ماشین حالت های نهایی q_3 و q_4 نیز تشخیص ناپذیر هستند. برای هر رشته w حاصل $\delta^*(q_3, w)$ و $\delta^*(q_4, w)$ حالت پذیرش است. پس این حالات ادغام پذیرند.

با ادغام حالات ادغام‌پذیر، به dfa معادل اما با کمترین تعداد حالات ممکن می‌رسیم.
 dfa، M_2 کمینه است.



الگوریتم کمینه کردن DFA:

این الگوریتم که به الگوریتم علامت‌گذاری (Mark) معروف است تمام جفت حالات تشخیص‌پذیر را طی چند مرحله متناهی شناسایی می‌کند و در پایان جفت‌های علامت‌گذاری نشده، رابطه ادغام‌پذیری را نشان می‌دهند که یک رابطه هم‌ارزی است و کلاس‌های هم‌ارزی، هر کدام یک حالت از dfa کمینه هستند.

الف: در صورت وجود حالات غیرقابل دسترس، آنها را حذف می‌کنیم. سپس مجموعه تمام زوج مرتب‌های $(p, q) \in Q \times Q$ را در نظر بگیرید. قصد داریم با علامت‌گذاری زوج‌های تشخیص‌پذیر (ادغام‌ناپذیر) کاری کنیم که زوج‌های تشخیص‌ناپذیر (ادغام‌پذیر) معلوم شوند.

ب: هر حالت نهایی $p \in F$ و هر حالت غیرنهایی $q \notin F$ ، ادغام‌ناپذیرند. پس (p, q) و (q, p) را علامت‌گذاری کنیم.

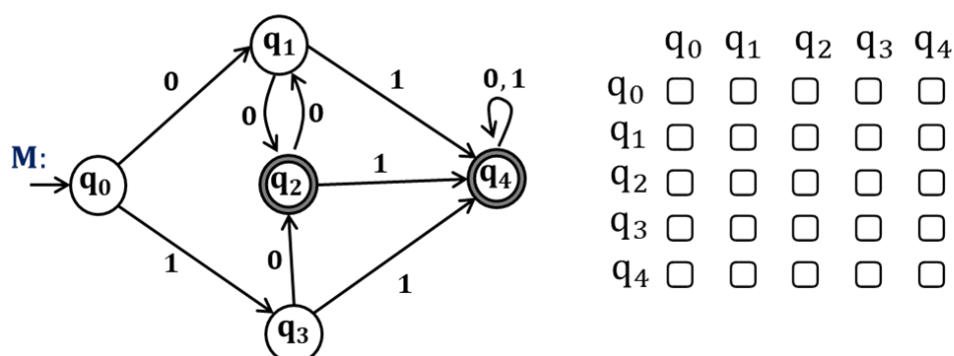
ج: برای هر الفبای $a \in \Sigma$ و هر جفت حالت (p, q) اگر $\delta(p, a) = p_a$ و $\delta(q, a) = q_a$ و (p_a, q_a) علامت‌گذاری شده‌اند، آنگاه زوج (p, q) را علامت‌گذاری کنیم.

مرحله ج را آنقدر تکرار کنیم که دیگر هیچ علامت‌گذاری جدیدی رخ ندهد.

مثال (Peter Linz)

با استفاده از الگوریتم علامت گذاری، ماشین قطعی کمینه را بیابید.

مرحله الف: حالت غیرقابل دسترس نداریم. مجموعه تمام زوج مرتب ها را در نظر بگیریم:



مرحله ب: علامت گذاری همه جفت هایی که یکی حالت پذیرش و دیگری غیرپذیرش است:

	q_0	q_1	q_2	q_3	q_4
q_0	☐	☐	×	☐	×
q_1	☐	☐	×	☐	×
q_2	×	×	☐	×	☐
q_3	☐	☐	×	☐	×
q_4	×	×	☐	×	☐

$$q_2, q_4 \in F$$

$$q_0, q_1, q_3 \notin F$$

مرحله ج: یافتن زوج های ادغام ناپذیر جدید

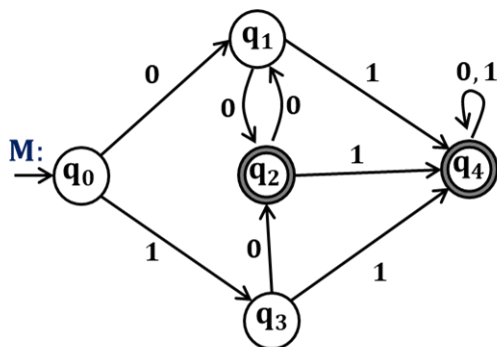
$$\begin{aligned} \times \quad & \delta(q_0, 0) = q_1 \\ & \delta(q_3, 0) = q_2 \end{aligned}$$

$$\begin{aligned} \times \quad & \delta(q_4, 0) = q_4 \\ & \delta(q_2, 0) = q_1 \end{aligned}$$

$$\begin{aligned} \times \quad & \delta(q_0, 0) = q_1 \\ & \delta(q_1, 0) = q_2 \end{aligned}$$

	q_0	q_1	q_2	q_3	q_4
q_0	$\square$	$\times$	$\times$	$\times$	$\times$
q_1	$\times$	$\square$	$\times$	$\square$	$\times$
q_2	$\times$	$\times$	$\square$	$\times$	$\times$
q_3	$\times$	$\square$	$\times$	$\square$	$\times$
q_4	$\times$	$\times$	$\times$	$\times$	$\square$

در پایان زوج مرتب های علامت گذاری نشده رابطه ادغام پذیری را نشان می دهند که یک رابطه هم ارزی است:



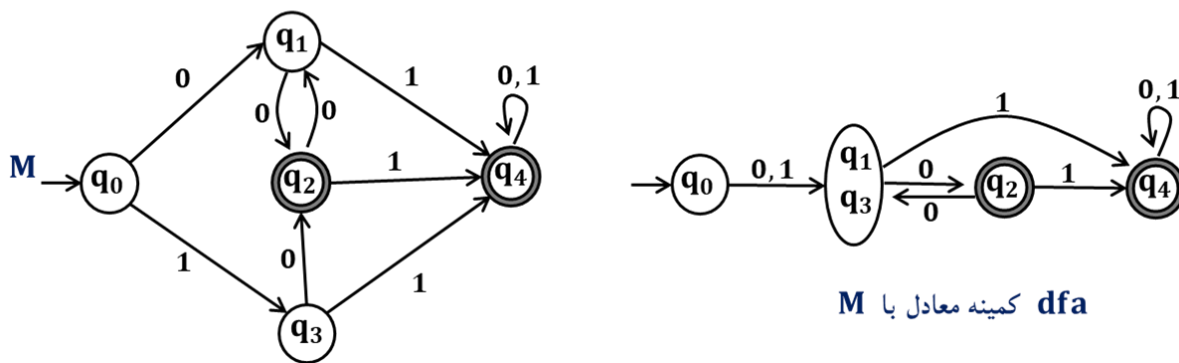
	q_0	q_1	q_2	q_3	q_4
q_0	$\blacksquare$	$\times$	$\times$	$\times$	$\times$
q_1	$\times$	$\blacksquare$	$\times$	$\blacksquare$	$\times$
q_2	$\times$	$\times$	$\blacksquare$	$\times$	$\times$
q_3	$\times$	$\blacksquare$	$\times$	$\blacksquare$	$\times$
q_4	$\times$	$\times$	$\times$	$\times$	$\blacksquare$

رابطه ادغام پذیری:

$$R = \{(q_0, q_0), (q_1, q_1), \dots, (q_4, q_4), (q_1, q_3), (q_3, q_1)\}$$

کلاس های هم ارزی عبارتند از: $\{q_0\}$, $\{q_1, q_3\}$, $\{q_2\}$, $\{q_4\}$ پس dfa کمینه ۴ حالت دارد.

در dfa کمینه هر کلاس هم‌ارزی به یک حالت تبدیل شده است.



۴۸- تعداد حالات ماشین متناهی قطعی (DFA) کمینه زبان $L = \{a^n b^n \mid n \leq 50\}$ چندتا است؟

مهندسی کامپیوتر ۱۴۰۲

(۱) 50

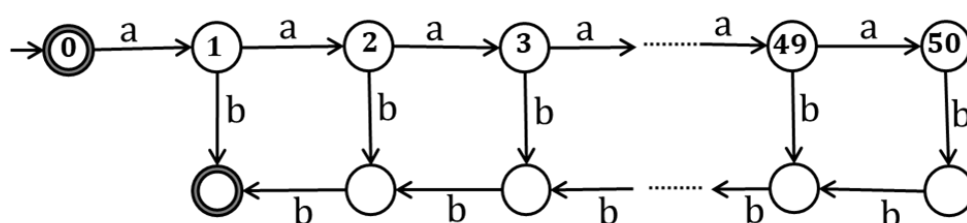
(۲) 51

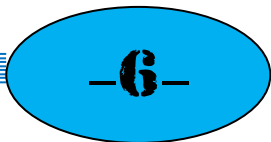
(۳) 100

(۴) 102

پاسخ: گزینه ۴

زبان L متناهی است پس در **DFA** کمینه، هیچ دور یا طوقی در مسیر از حالت اولیه تا حالت پذیرش وجود ندارد. لازم است ترتیب a و b و تساوی تعداد آنها را به ازای $0 \leq n \leq 50$ توسط ماشین محقق شود. رشته تهی عضو زبان است پس حالت اولیه حالت پذیرش است. با این توضیحات شکل زیر به دست می‌آید. در این شکل حالت تله رسم نشده است. تمام انتقال‌های رسم نشده به حالت تله می‌روند. با در نظر گرفتن آن ۱۰۲ حالت خواهیم داشت و هیچکدام با دیگری ادغام پذیر نیست.





6

فصل دوم: ماشین متناهی

جلسه ششم بخش یک: نکات تکمیلی ماشین های متناهی

رابطه روابط ماشین متناهی و زبان منظم

سوال: چند پذیرنده متناهی قطعی (DFA) با 2 حالت و الفبای $\{0,1\}$ وجود دارد؟
(مهندسی کامپیوتر 97)

- 16(1)
26(2)
32(3)
64(4)

پاسخ: گزینه 4 صحیح است.

تعداد ماشین های متناهی قطعی متمایز با n حالت و m الفبا برابر است با: $2^n n^m$.

توضیح:

ببینیم چند پذیرنده متناهی قطعی (DFA) با n حالت و m الفبای $\{a_1, a_2, \dots, a_m\}$ وجود دارد؟
فرض کنید حالات ماشین $q_0, q_1, \dots, q_{n-1}$ باشند. هر حالت می تواند نهایی یا غیرنهایی باشد. (برای هر حالت دو انتخاب). در هر کدام از حالات با الفبای a_1 به n حالت می توان انتقال تعریف کرد. با الفبای a_2 به n حالت می توان انتقال تعریف کرد و به همین ترتیب برای هر الفبا n انتخاب داریم.

$$\underbrace{2 \times 2 \times \dots \times 2}_{\text{به تعداد حالات}} \times \underbrace{n^m \times n^m \times \dots \times n^m}_{\text{به تعداد حالات}} = 2^n n^m$$

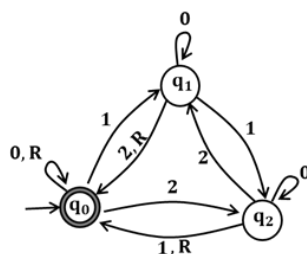
$$\text{گزینه صحیح} = 2^2 \times 2^{2 \times 2} = 64$$

مساله: (Michael sipser)

مجموعه الفبای $\Sigma = \{0, 1, 2, \text{Reset}\}$ را در نظر بگیرید. زبان L مجموعه تمام رشته‌های متناهی است که مجموع رقم‌هایش به 3 بخشپذیر باشد با این شرط که Reset باعث صفر شدن مجموع ارقام قبل از خود شود. نشان دهید این زبان منظم است. ماشین متناهی قطعی کمینه آن، چند حالت دارد؟

- 1- دو حالت 2- سه حالت 3- چهار حالت 4- پنج حالت

پاسخ: گزینه 2 صحیح است.



الفبای Reset را با R نشان داده‌ایم. رشته تهی عضو زبان L است پس q_0 حالت پذیرش است. الفبای Reset در انتهای هر رشته ای بیاید آن را به رشته عضو L تبدیل می‌کند بنابراین از هر حالتی با این الفبا به حالت پذیرش منتقل می‌شویم. رقم 0 در مجموع ارقام بی‌تاثیر است پس باعث تغییر حالت نمی‌شود. سایر انتقال‌ها نیز طوری تعریف شده که با رسیدن به مجموع 3 به حالت پذیرش برگردیم.

هیچ کدام از جفت حالات ماشین، ادغام‌پذیر نیستند. پس این dfa کمینه است.

مساله: (Michael sipser)

زبان $L \subseteq \{0, 1\}^*$ شامل تمام رشته هایی است که رقم 1 سومین الفبای آنها از سمت راست باشد. dfa کمینه پذیرنده L چند حالت دارد؟

1- سه حالت 2- پنج حالت 3- شش حالت 4- هشت حالت

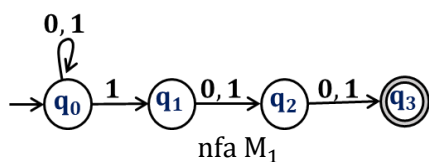
پاسخ: گزینه 4 صحیح است.

$$L = \{a_1 \dots a_{n-2}a_{n-1}a_n \in (0+1)^* \mid a_{n-2} = 1\}$$

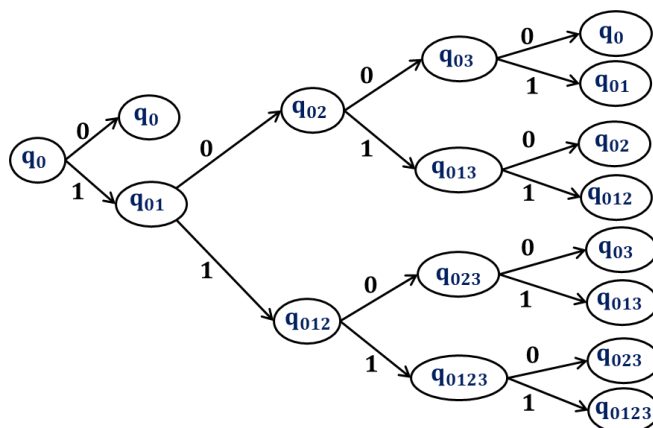
رشته های عضو L به این فرم هستند:

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

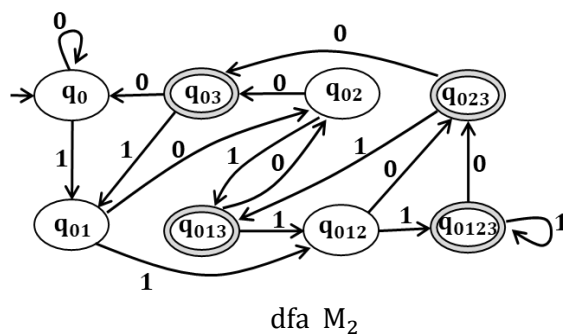
می توانیم nfa پذیرنده برای این رشته ها طراحی کنیم. سپس آن را به dfa معادل تبدیل کنیم.



درخت انتقال در nfa چنین است:



منظورمان از q_{012} مجموعه $\{q_0, q_1, q_2\}$ است. با توجه به این درخت، ماشین dfa معادل به دست می آید. حالاتی که شامل q_3 باشند حالت پذیرش هستند:



آیا این dfa ، کمینه است؟ باید ببینیم آیا حالات ادغام‌پذیر در M_2 وجود دارند یا خیر؟ با کمی دقت متوجه می‌شویم هیچکدام از جفت حالات پذیرش، ادغام‌پذیر نیستند. هیچکدام از جفت حالات غیرپذیرش نیز ادغام‌پذیر نیستند. در این جدول رشته ای که ادغام ناپذیری جفت حالات را مشخص می‌کند آمده است.

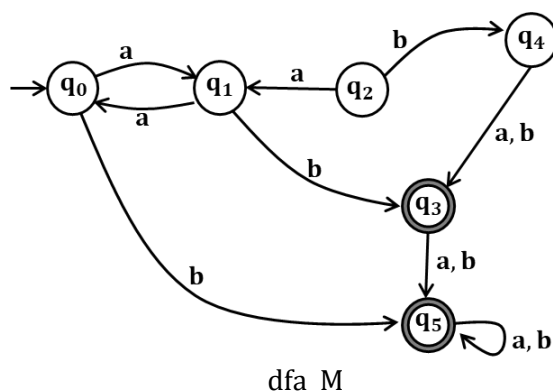
جفت حالت	رشته
$q_{023} \quad q_{03}$	0
$q_{013} \quad q_{03}$	00
$q_{0123} \quad q_{03}$	1
$q_{023} \quad q_{013}$	1
$q_{0123} \quad q_{013}$	1
$q_{023} \quad q_{03}$	0

جفت حالت	رشته
$q_{01} \quad q_0$	11
$q_{02} \quad q_0$	0
$q_{012} \quad q_0$	1
$q_{02} \quad q_{01}$	0
$q_{012} \quad q_{01}$	1
$q_{01} \quad q_0$	11

مثلا q_{023} و q_{03} را در نظر بگیرید. از اولی با مصرف 0 به حالت پذیرش می‌رسیم. از دومی با مصرف 0 به حالت غیرپذیرش می‌رسیم. پس این دو حالت ادغام‌ناپذیرند. با این توضیحات مشخص می‌شود که M_2 کمینه است. ماشین قطعی کمینه L دارای 8 حالت است.

مسئله: (peter linz)

ماشین متناهی M داده شده است. ماشین متناهی قطعی کمینه معادل با M دارای چند حالت است؟



1- دو حالت

2- سه حالت

3- چهار حالت

4- پنج حالت

پاسخ: گزینه 1 صحیح است.

دو حالت پذیرش داریم که ادغام پذیرند زیرا از هر دوی آنها با هر رشته ای به حالت پذیرش می‌رسیم:

$$\forall w (\delta^*(q_5, w) \in F \text{ and } \delta^*(q_3, w) \in F)$$

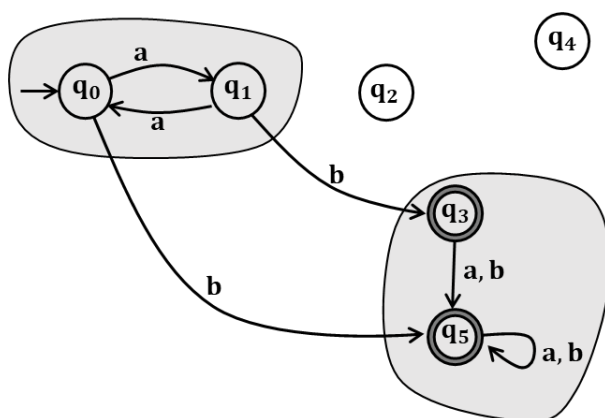
پس q_5 و q_3 ادغام پذیرند.

در مورد حالات غیر نهایی پیش از بررسی ادغام‌پذیری، توجه کنیم که q_2 و q_4 غیرقابل دسترس هستند پس به سادگی حذف می‌شوند. برای هر w داریم:

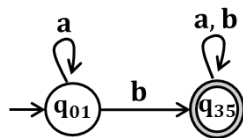
$$\delta^*(q_1, w) \in F \Leftrightarrow \delta^*(q_0, w) \in F$$

یا معادل با آن:

$$\delta^*(q_1, w) \notin F \Leftrightarrow \delta^*(q_0, w) \notin F$$



با این توضیحات ماشین قطعی کمینه، دارای 2 حالت است:

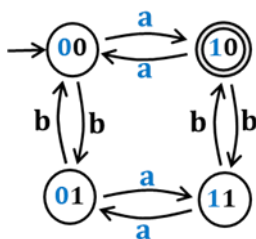
dfa M_1

M_1 ماشین کمینه معادل با M است. زبانی که توسط این ماشین ها پذیرفته می شود مجموعه رشته های است که حداقل یک b دارند. می توان گفت این زبان مکمل مجموعه رشته های است که فقط از الفبای a تولید می شوند:

$$L(M) = L(M_1) = \{w | n_b(w) \geq 1\} = \overline{\{\lambda, a, a^2, \dots\}}$$

یادآوری:

پیش از این برای زبان $L = \{w | n_a(w) \bmod 2 = 1 \text{ and } n_b(w) \bmod 2 = 0\}$ ماشین متناهی پذیرنده طراحی کرده ایم:

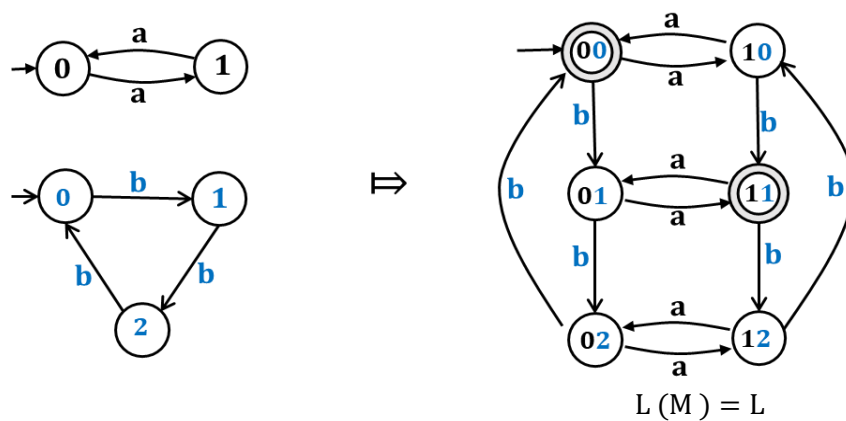


با هر کدام از الفباهای یک دور به طول 2 ایجاد کردیم و این دورها را با هم ترکیب کردیم. حالا همین تکنیک را در یک مثال دیگر به کار بگیریم.

تمرین: فرض کنیم $\Sigma = \{a, b\}$. یک ماشین متناهی قطعی پذیرنده زبان L طراحی کنید:

$$L = \{w \in (a + b)^* | n_a(w) \bmod 2 = n_b(w) \bmod 3\}$$

پاسخ: با الفبای b دور به طول 3 و با الفبای a دور به طول 2 می سازیم حالا آنها را ترکیب می کنیم. مطابق شکل، در هر کدام از حالات رقم اول $n_a(w) \bmod 2$ و رقم دوم $n_b(w) \bmod 3$ را نشان می دهد.



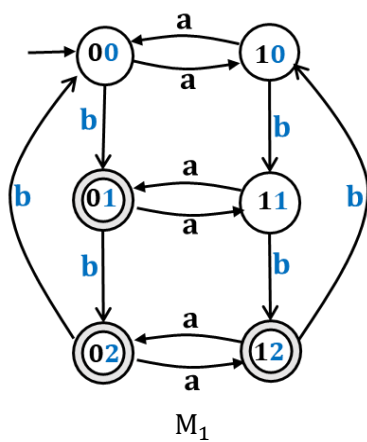
در حالات 00 و 11 این ارقام برابرند پس این دو حالت، حالات پذیرش هستند.

توجه: حالات پذیرش با توجه به نیاز ما تعیین می‌شوند. مثلاً برای پذیرش این زبان:

$$L_1 = \{w \in (a + b)^* \mid n_a(w) \bmod 2 < n_b(w) \bmod 3\}$$

حالات 01 و 02 و 12.

را حالت پذیرش می‌گیریم که در آنها عدد اول کوچکتر از عدد دوم است.

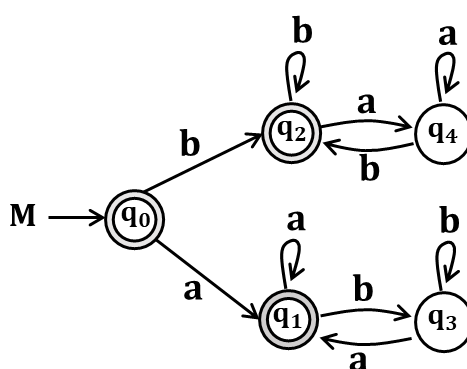


$$L(M_1) = L_1$$

پذیرنده متناهی برای $\bar{L}$:

اگر M یک dfa پذیرنده زبان L باشد، با تبدیل حالات پذیرش به غیرپذیرش و حالات غیرپذیرش به پذیرش ماشین dfa پذیرنده زبان $\bar{L}$ به دست می‌آید.

مثال: فرض کنیم $\Sigma = \{a, b\}$ و A مجموعه تمام رشته‌هایی باشد که تعداد تکرار های ab و ba در آنها برابر نیست. می‌خواهیم پذیرنده dfa برای A پیدا کنیم. در بخش های پیشین dfa پذیرنده $L = \{w \mid n_{ab}(w) = n_{ba}(w)\}$ را یافته‌ایم:

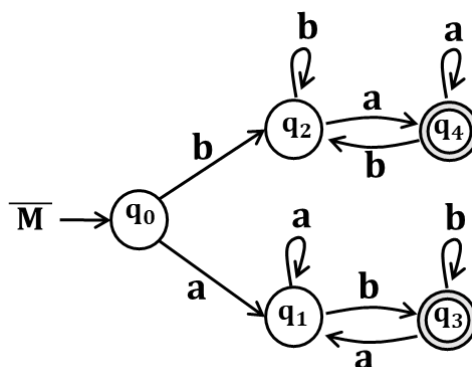


$$L(M) = L = \{w \mid n_{ab}(w) = n_{ba}(w)\}$$

زبان A مکمل L است:

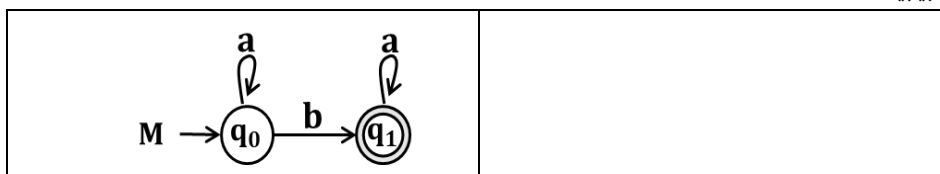
$$A = \bar{L} = \{w \mid n_{ab}(w) \neq n_{ba}(w)\}$$

پس با تبدیل حالات پذیرش به غیرپذیرش و بالعکس، dfa پذیرنده آن به دست می‌آید.

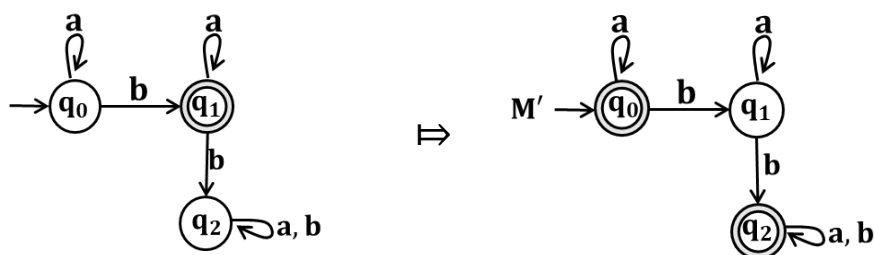


$$L(\bar{M}) = \bar{L} = \{w \mid n_{ab}(w) \neq n_{ba}(w)\}$$

مساله: فرض کنیم $\Sigma = \{a, b\}$. ماشین متناهی M داده شده است. ماشین متناهی پذیرنده $\overline{L(M)}$ را بیابید.



پاسخ: M ماشین قطعی نیست. نمیتوان با راهکار پیشین، از ماشین nfa مکمل گرفت. ابتدا باید dfa معادل را یافت. با کامل کردن انتقالها و افزودن حالت تله M را به dfa تبدیل کنیم. سپس با تغییر حالات پذیرش به غیرپذیرش و بالعکس، ماشین پذیرنده $\overline{L(M)}$ به دست می آید.



در این مساله:

$$L(M) = \{w | n_b(w) = 1\}, \quad \overline{L(M)} = \{w | n_b(w) = 0 \text{ or } n_b(w) \geq 2\}$$

فصل دوم: ماشین متناهی

جلسه ششم بخش دو:

نکات تکمیلی ماشین های متناهی + ویژگی ها و روابط ماشین متناهی و زبان منظم

مساله: (Michael sipser)

فرض کنیم $\Sigma = \{0\}$. زبان L شامل تمام رشته های به فرم 0^k است که k مضرب 2 یا 3 باشد. پذیرنده dfa کمینه L چند حالت دارد؟

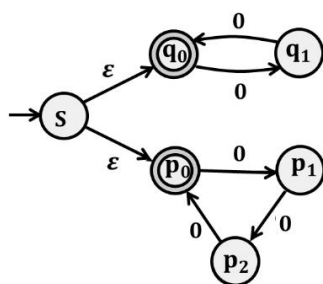
- 1- دو حالت 2- سه حالت 3- چهار حالت 4- پنج حالت

پاسخ:

$$L = \{0^k \mid k \bmod 2 = 0 \text{ or } k \bmod 3 = 0\}$$

$$= \{0^k \mid k \bmod 2 = 0\} \cup \{0^k \mid k \bmod 3 = 0\}$$

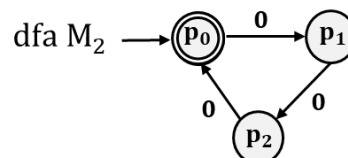
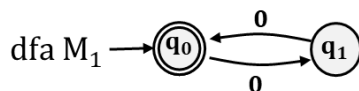
با توجه به فرم رشته های عضو L یافتن nfa پذیرنده ساده است:



می توانیم nfa را به dfa معادل تبدیل کنیم. اما اینجا می خواهیم روش جدیدی را معرفی کنیم. زبان L اجتماع دو زبان منظم است:

$$L = \underbrace{\{0^k \mid k \bmod 2 = 0\}}_{L_1} \cup \underbrace{\{0^k \mid k \bmod 3 = 0\}}_{L_2}$$

برای هریک از زبان های L_1 و L_2 پذیرنده dfa به سادگی به دست می آید. M_1 و M_2 ماشین های قطعی پذیرنده L_1 و L_2 هستند:



ماشین قطعی M را با مجموعه حالات $Q_1 \times Q_2$ در نظر می گیریم.

$$Q_1 \times Q_2 = \{(q_0, p_0), (q_0, p_1), (q_0, p_2), (q_1, p_0), (q_1, p_1), (q_1, p_2)\}$$

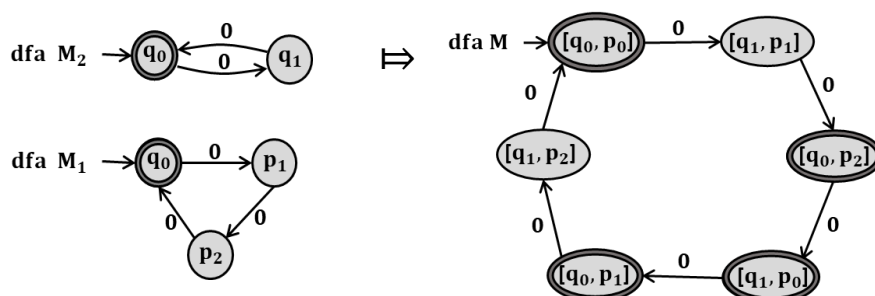
تابع انتقال در چنین تعریف می شود:

$$\delta((q, p), a) = (q', p') \Leftrightarrow \delta_1(q, a) = q' \text{ and } \delta_2(p, a) = p'$$

برای آن که M پذیرنده ی $L_1 \cup L_2$ باشد حالات پذیرش را به این صورت انتخاب می کنیم:

$$q \in F_1 \vee p \in F_2 \Rightarrow (q, p) \in F$$

حالت (q, p) به شرطی حالت پذیرش است که حداقل یکی از مولفه هایش، حالت پذیرش باشد.

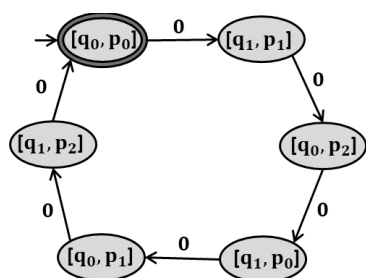


توجه:

در مساله قبل اگر بخواهیم M پذیرنده $L_1 \cap L_2$ باشد، حالاتی را حالت پذیرش می گیریم که هر دو مولفه آنها از حالات پذیرش ماشین های M_1 و M_2 باشند.

$$L_1 \cap L_2 = \{0^k \mid k \bmod 2 = 0 \text{ and } k \bmod 3 = 0\}$$

$$(q, p) \in F \Leftrightarrow q \in F_1 \wedge p \in F_2$$



به عبارتی برای پذیرش $L_1 \cap L_2$ مجموعه حالات پذیرش را $F = F_1 \times F_2$ تعریف می کنیم.

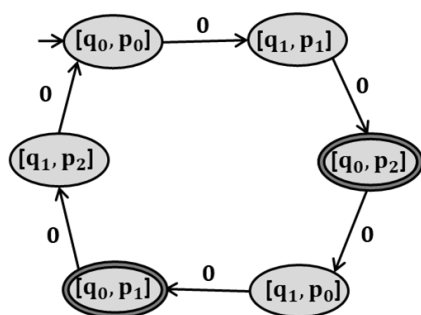
توجه: فرض کنیم M_1 و M_2 ماشین های متناهی قطعی پذیرنده L_1 و L_2 باشند. ماشین M با مجموعه حالات $Q_1 \times Q_2$ و تابع انتقال δ را معرفی کردیم.

$$\delta((q, p), a) = (\delta_1(q, a), \delta_2(p, a))$$

در این جدول انواع حالات پذیرش (پایانی) ماشین M و زبانی که توسط آن پذیرفته می شود آمده است:

تعداد حالات پایانی	$L(M)$	مجموعه حالات پایانی F
$ F_1 F_2 $	$L_1 \cap L_2$	$F_1 \times F_2$
$ F_1 Q_2 - F_2 $	$L_1 - L_2$	$F_1 \times (Q_2 - F_2)$
$ Q_1 - F_1 F_2 $	$L_2 - L_1$	$(Q_1 - F_1) \times F_2$
$ F_1 Q_2 - F_2 + Q_1 - F_1 F_2 $	$L_1 \Delta L_2$	$F_1 \times (Q_2 - F_2) \cup (Q_1 - F_1) \times F_2$
$ F_1 Q_2 + Q_1 F_2 - F_1 F_2 $	$L_1 \cup L_2$	$(F_1 \times Q_2) \cup (Q_1 \times F_2)$

مثال: اگر بخواهیم ماشین M پذیرنده $L_1 - L_2$ باشد حالات پذیرش را $F = F_1 \times (Q_2 - F_2)$

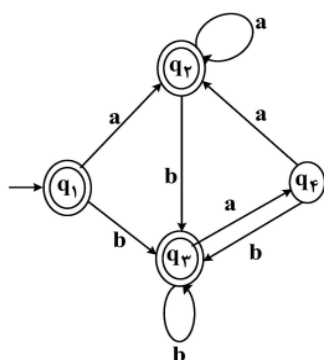


در نظر می‌گیریم. حالت (q, p) حالت

پذیرش است اگر و تنها اگر q در M_1 پذیرش

باشد اما p در M_2 پذیرش نباشد.

(علوم کامپیوتر 1404)

۹۰- فرض کنید L زبان dfa زیر باشد:فرض کنیم $w \in L$ رشته‌ای از زبان L با طول حداقل 1403 باشد. کدام مورد، همواره درست است؟الف - اگر w با a شروع شود رشته wa^{1403} حتماً در L خواهد بود.ب - اگر w با b شروع شود رشته wba^{1403} حتماً در L خواهد بود.

(۱) فقط «الف»

(۲) فقط «ب»

(۳) هر دو

پاسخ: گزینه ۲

الف نادرست است. برای مثال $w = ab^{1402}$ با طول عضو L است. اما $x = a^{1403}wa$ عضو L نیست.رشته w در q_3 پذیرفته می‌شود. اما با مصرف رشته x به حالت q_4 می‌رسیم که غیرپذیرش است.

$$w = ab^{1402} \Rightarrow x = a^{1403}wa = a^{1403}ab^{1402}a \Rightarrow \delta^*(q_1, x) = q_4$$

ب درست است. زیرا در این dfa تمام انتقال‌های b به حالت پذیرش q_3 وارد می‌شوند. پس هر رشته‌ایکه به b ختم شود پذیرفته می‌شود.

روابط بین NFA و DFA و زبان منظم:

- 1-** هر ماشین NFA (چه انتقال λ داشته باشد چه نداشته باشد) دارای DFA معادل است.
- 2-** L منظم است $\Leftrightarrow L$ گرامر منظم دارد $\Leftrightarrow L$ پذیرنده DFA دارد $\Leftrightarrow L$ پذیرنده NFA دارد.
- 3-** ماشین های DFA و NFA پذیرنده زبان منظم L ، منحصر به فرد نیستند.
- 4-** اگر NFA دارای n حالت باشد حتما دارای DFA معادلی با حداکثر 2^n حالت است.

در تکمیل آخرین مورد، می دانیم که هر زیرمجموعه از حالات NFA می تواند یک حالت از DFA معادل باشد. با در نظر گرفتن تمام آنها، یک DFA معادل با دقیقا 2^n حالت وجود دارد. برخی از این حالات ممکن است قابل حذف یا ادغام پذیر باشند. پس یک DFA معادل با حداکثر 2^n حالت وجود دارد. در ضمن اضافه کردن حالات به ماشین متناهی به سادگی امکان پذیر است پس DFA معادل با تعداد حالات بیش از 2^n نیز وجود دارد.

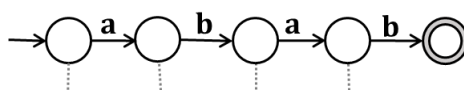
روابط مهم بین زبان منظم و DFA پذیرنده آن:

1-	اگر M یک DFA با n حالت باشد و رشته‌ای به طول حداقل n را بپذیرد، آنگاه گراف انتقال دارای دور است و زبان $L(M)$ نامتناهی خواهد بود.
2-	اگر DFA دارای n حالت باشد و $L(M)$ تهی نباشد، طول کوچکترین رشته‌ای که پذیرفته می‌شود، کمتر از n است.
3-	اگر اندازه رشته‌ها در زبان $L(M)$ حداقل k باشد، تعداد حالات DFA حداقل $k + 1$ است.
4-	اگر زبان $L(M)$ نامتناهی باشد آنگاه: حداقل یک رشته با اندازه $0 \leq w < n$ دارد، حداقل یک رشته با اندازه $n \leq w < 2n$ دارد.

مثال برای توضیح مورد 3:

$$L = \{abab, b^2a^3, b^8, a^5b^3, \dots\}$$

در این زبان کوچکترین رشته به طول 4 است. مسیر پذیرش این رشته باید مسیری ساده فاقد دور یا طوق باشد در غیر این صورت با حذف آن دور، رشته کوچکتری نیز پذیرفته می‌شود. مسیر ساده به طول 4 دارای 5 گره است. پس حداقل 5 حالت در dfa پذیرنده L وجود دارد.

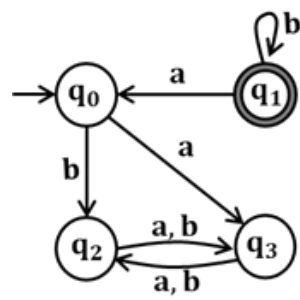


در توضیح موارد 1 تا 4:

فرض کنیم M یک DFA با n حالت باشد. زبان منظم $L(M)$ در یکی از این سه حالت قرار دارد:

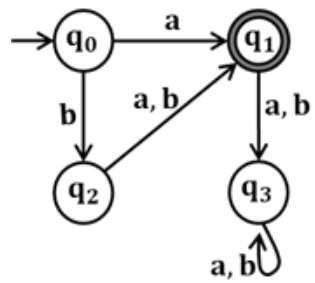
- حالت اول:** $L(M)$ تهی است و هیچ رشته‌ای ندارد.
- حالت دوم:** $L(M)$ متناهی است، فقط دارای رشته‌های با طول کمتر از n است.
- حالت سوم:** $L(M)$ نامتناهی است، هم رشته با طول کمتر از n و هم رشته با طول بیشتر از n دارد.

به ازای $n = 4$ مثالی از هر حالت را مشاهده کنید:



حالت اول

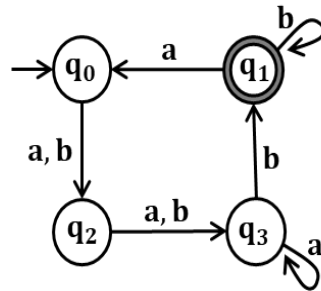
$$L(M) = \{ \}$$



حالت دوم

$$L(M) = \{a, ba, bb\}$$

حالت سوم

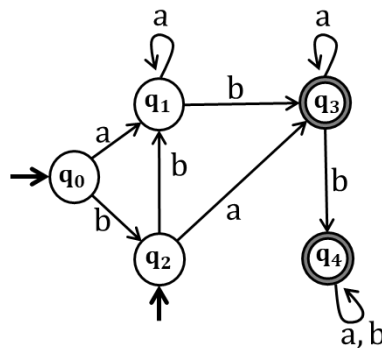


$$L(M) = \{b^3, ab^2, b^4, bab^2, a^2b^2, a^2b^3, b^5, \dots\}$$

5-	در DFA: $q_0 \Leftrightarrow \lambda \in L(M)$ حالت پذیرش است. در NFA: $q_0 \Rightarrow \lambda \in L(M)$ حالت پذیرش است.
6-	اگر DFA کمینه L دارای m حالت باشد DFA کمینه L^R دارای n حالت است که: $[\log m] \leq n \leq 2^m$
7-	هر NFA را می توان به NFA معادل با فقط یک حالت آغاز و یک حالت پایانی (پذیرش) تبدیل کرد. اما در مورد DFA ممکن است نتوان تعداد حالات پذیرش را به یک حالت رساند.

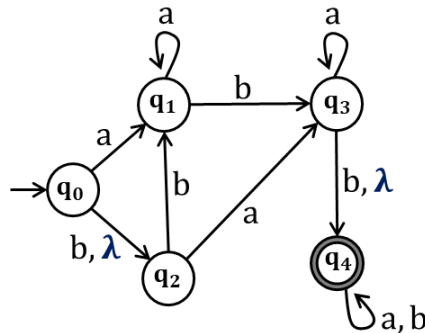
توضیح مورد 7:

فرض کنید M ماشین متناهی (قطعی یا غیر قطعی) با چند حالت اولیه و چند حالت پایانی باشد:

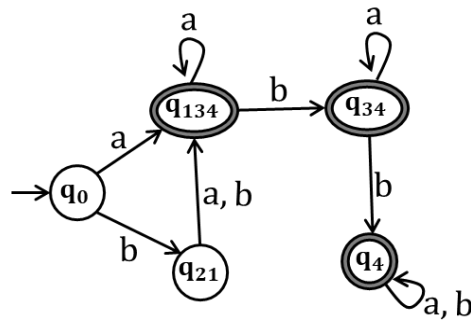


M

با کمک انتقال λ می‌توان M را به NFA معادلی که یک حالت اولیه و یک حالت پایانی دارد تبدیل کرد.



حال الگوریتم تبدیل NFA به DFA را اجرا می‌کنیم. DFA معادلی داریم که تنها یک حالت اولیه دارد اما تعداد حالات نهایی ممکن قابل کاهش به یک حالت نباشد:



DFA پذیرنده L^R :

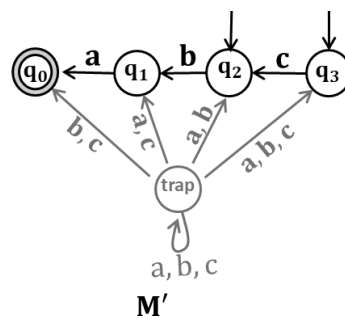
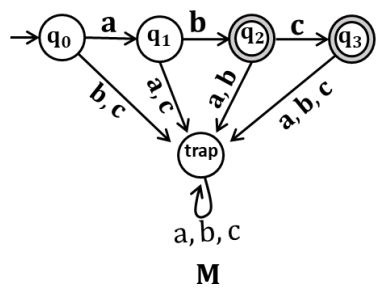
فرض کنید M یک ماشین DFA پذیرنده L باشد. در این ماشین انتقال ها را معکوس کنیم حالت اولیه را به حالت پذیرش و حالات پذیرش را به حالات اولیه تبدیل کنیم. اگر، چند حالت اولیه به وجود آمد با انتقال λ تعدادشان را به یک حالت کاهش می‌دهیم. به این ترتیب یک NFA پذیرنده L^R به دست می‌آید. حال الگوریتم تبدیل NFA به DFA را اجرا می‌کنیم، یک ماشین قطعی پذیرنده L^R به دست می‌آید.

یک نمونه ساده را در این شکل مشاهده می‌کنید. ماشین متناهی M پذیرنده $L = \{ab, abc\}$

است. با تغییر جهت انتقال ها و تبدیل حالت اولیه به حالت پذیرش و بالعکس، به M'

می‌رسیم که پذیرنده قطعی برای $L^R = \{ba, cba\}$ است. حالت تله در M' به حالتی

غیرقابل دسترس تبدیل شده و قابل حذف است. می‌توانیم با تبدیل غیرقطعی به قطعی
DFA پذیرنده L^R به دست آوریم.







-7-

فصل سوم: گرامر منظم - عبارت منظم - زبان منظم

جلسه هفتم بخش یک: عبارات منظم

تعریف عبارات منظم:

فرض کنیم Σ مجموعه الفبا باشد. عبارات منظم با سه قانون معرفی می‌شوند:

الف: $\emptyset$ و λ و هر الفبای $a \in \Sigma$ را عبارت منظم اولیه می‌نامیم.

ب: اگر r_1 و r_2 دو عبارت منظم باشند آنگاه $r_1 + r_2$ و $r_1 r_2$ و r_1^* و (r_1) نیز منظم هستند.

ج: فقط عباراتی منظم هستند که از عبارات منظم اولیه و تعدادی متناهی به کار بستن قانون (ب) به دست آمده باشند.

مثال: فرض کنیم $\Sigma = \{a, b, c\}$. این عبارات منظم هستند:

$$(b + \lambda)(a + bc)^*$$

$$a^+ = aa^*$$

$$\Sigma^2 = (a + b + c)^2$$

تعریف زبان $L(r)$ برای عبارات منظم r :

برای هر عبارت منظم r یک زبان منظم $L(r)$ وجود دارد. برای معرفی $L(r)$ از این قواعد استفاده می‌کنیم:

الف: برای عبارات منظم اولیه:

$$L(\emptyset) = \{ \}$$

$$L(\lambda) = \{ \lambda \}$$

$$L(a) = \{ a \}$$

ب: در مورد عملگرها و پرانتز:

$$L((r)) = L(r)$$

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

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

$$L(r^*) = (L(r))^*$$

عبارت منظم r	زبان $L(r)$
$\emptyset$	$\{ \}$
λ	$\{\lambda\}$
a	$\{a\}$
$a + b$	$\{a\} \cup \{b\}$
$(a + b)$	$\{a\} \cup \{b\}$
$r_1 + r_2$	$L(r_1) \cup L(r_2)$
$r_1 r_2$	$L(r_1)L(r_2)$
(r)	$L(r)$
r^*	$(L(r))^*$

مثال: محاسبه گام به گام $L(a^*(b + c))$:

$$\begin{aligned}
 L(a)^*L(b + c) &= L(a^*(b + c)) = L(a^*)L(b + c) = \\
 &= \{a\}^*(L(b) \cup L(c)) = \{a\}^*({\{b\} \cup \{c\}}) = \\
 &= \{\lambda, a, a^2, a^3, \dots\}\{b, c\} = \{b, c, ab, ac, a^2b, a^2c, \dots\}
 \end{aligned}$$

اولویت عملگرها:

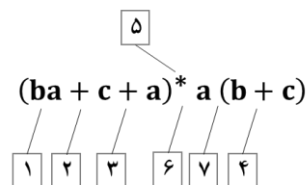
1 پرانتز

2 بستار

3 الحاق

4 جمع

5 چپ به راست



تمرینات بخش 1-3. (peter linz)

برای این زبان‌ها عبارت منظم معادل بیابید.

4. $L = \{a^n b^m \mid n \geq 3, m \text{ زوج باشد}\}$

5. $L = \{a^n b^m \mid n + m \text{ زوج باشد}\}$

6. $L_1 = \{a^n b^m \mid n \geq 4, m \leq 2\}$

7. $\overline{L_1}$

پاسخ:

4. $L = L(a^* a^3 (b^2)^*)$

5. $L = L((a^2)^* (\lambda + ab) (b^2)^*)$

اگر $n + m$ زوج باشد یا هر دو زوجند $((a^2)^* (\lambda) (b^2)^*)$ یا هر دو فردند $((a^2)^* (ab) (b^2)^*)$.

6. $L_1 = L(a^* a^4 (\lambda + b + b^2))$

شرط $n \geq 4$ را به فرم $a^* a^4$ یا $a^4 a^*$ بیان می‌کنیم. شرط $m \leq 2$ نشان می‌دهد که این قسمت به فرم b^2 یا b^1 یا λ خواهد بود.

7. $\overline{L_1} = L(\lambda + (b + ab + a^2 b + a^3 b + a^4 a^* b^3) \Sigma^* + a^4 a^* (\lambda + b + b^2) \Sigma^+)$

توجه کنید که عبارت منظم معادل با هر زبان منحصر به فرد نیست و جواب می‌تواند به فرم‌های دیگر هم نوشته شود.

تمرین:

عبارت منظم معادل با این زبان‌ها را بنویسید. ($\Sigma = \{a, b\}$)

الف: L_1 : مجموعه رشته‌های به طول زوج.

ب: L_2 : مجموعه رشته‌هایی که طول آنها $3k + 1$ است.

پاسخ:

الف: همه رشته‌های به طول زوج با کنار هم گذاشتن تعداد دلخواهی قطعه به طول دو تولید می‌شوند.

$$L_1 = L((aa + ab + ba + bb)^*) = L\left(\left(\Sigma^2\right)^*\right)$$

ب: با کنار هم گذاشتن تعداد دلخواهی قطعه به طول سه همه رشته‌های به طول $3k$ تولید می‌شوند.

کافیست یک حرف به انتها (یا ابتدای) این رشته‌ها اضافه کنیم تا اندازه شان $3k + 1$ شود.

$$L_2 = L((aaa + aab + abb + \dots + bbb)^*(a + b)) = L\left(\left(\Sigma^3\right)^* \Sigma\right)$$

منتخب تمرینات بخش 3-1. (peter linz)

13. یک عبارت منظم برای زبان $\{vwv \mid w \in (a + b)^*, |v| = 2\}$ بیابید.

14. یک عبارت منظم برای زبان $\{vwv \mid w \in (a + b)^*, |v| \leq 3\}$ بیابید.

پاسخ:

13: در این زبان تعداد الگوهای v متناهی است (چهار رشته به طول دو داریم) اما w دلخواه است.

$$r_1 = aa(a + b)^*aa + ab(a + b)^*ab + bb(a + b)^*bb + ba(a + b)^*ba$$

14: در L_2 رشته v در حالت خاص می‌تواند λ باشد. پس L_2 شامل تمام $(a + b)^*$ است:

$$r_2 = (a + b)^*$$

16. عبارات منظم معادل با این زبان‌ها را بنویسید. ($\Sigma = \{a, b, c\}$)

1. رشته‌هایی که دارای دقیقاً یک a هستند.
2. رشته‌هایی که دارای حداقل یک a هستند.
3. رشته‌هایی که دارای دقیقاً دو a هستند.
4. رشته‌هایی که دارای حداکثر دو a هستند.
5. رشته‌هایی که از هر الفبا حداقل یکی داشته باشند.
6. رشته‌هایی که Run به طول بیشتر از 2 از الفبای a نداشته باشد.
7. رشته‌هایی که در آنها طول هر Run از a مضرب 3 باشد.

توجه: هر توالی از الفبای a را یک Run از a می‌نامیم.

پاسخ:

1. رشته‌هایی که دارای دقیقاً یک a هستند.

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

2. رشته‌هایی که دارای حداقل یک a هستند.

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

3. رشته‌هایی که دارای دقیقاً دو a هستند.

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

4. رشته‌هایی که دارای حداکثر دو a هستند.

$$(b+c)^* (\lambda + a) (b+c)^* (\lambda + a) (b+c)^*$$

5. رشته‌هایی که از هر الفبا حداقل یکی داشته باشند.

$$\Sigma^* a \Sigma^* b \Sigma^* c \Sigma^* + \Sigma^* b \Sigma^* c \Sigma^* a \Sigma^* + \dots + \Sigma^* c \Sigma^* a \Sigma^* b \Sigma^*$$

6. رشته‌هایی که Run به طول بیشتر از 2 از الفبای a نداشته باشد.

$$(b+c+ab+ac+aab+aac)^* (\lambda + a + aa)$$

7. رشته‌هایی که در آنها طول هر Run از a مضرب 3 باشد.

$$(b+c+a^3)^*$$

17. برای هر زبان عبارت منظم بیابید

- الف. رشته هایی که به 01 ختم می شوند.
 ب. رشته هایی که به 01 ختم نمی شوند.
 ج. رشته هایی که تعداد زوجی 0 دارند..
 د. رشته هایی که حداقل دو زیر رشته 00 دارند.

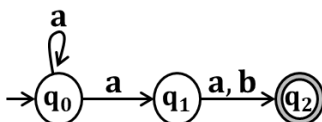
پاسخ:

- الف. $(0 + 1)^* 01$
 ب. $(0 + 1)^* (00 + 01 + 11) + \lambda + 0 + 1$
 ج. $(00 + 1)^*$
 د. $(0 + 1)^* (000)(0 + 1)^* + (0 + 1)^* 00 (0 + 1)^* 00 (0 + 1)^*$

(تکمیلی)

26. یک nfa پذیرنده برای زبان $L(aa^*(a + b))$ بیابید.

پاسخ: با توجه به عبارت منظم داده شده طراحی nfa را انجام می دهیم:



ویژگی های عبارات منظم:

در این روابط x و y دو رشته دلخواه هستند.

$$(x+y)^* = (x^*+y)^* = (x^*+y^*)^* = (x^*y^*)^* = (y^*x^*)^* = (x^*y^*)^+ = (y^*x^*)^+ \quad 1$$

$$(x+y)^* = x^*(yx^*)^* = y^*(xy^*)^* \quad 2$$

$$(x+y)^* = (x^*y)^* + (y^*x)^* \quad 3$$

$$x^*y^* \cap y^*x^* = x^* \cup y^* \quad 4$$

$$(x^*)^R = (x^R)^* \quad 5$$

$$(x^*)^+ = (x^*)^* = x^*x^*x^* \quad 6$$

$$(xx)^* \neq x^*x^*x^*, \quad (xyz)^* \neq x^*y^*z^* \quad 7$$

تمرین: این عبارت منظم را ساده کنید:

$$x^*y^*(x+y)^*x^*y^* = ?$$

پاسخ: ترکیب $\lambda (x+y)^* \lambda$ نیز با این عبارت منظم تولید می شود در نتیجه تمام ترکیبات دلخواه از x و y تولید می شوند.

$$x^*y^*(x+y)^*x^*y^* = (x+y)^*$$

مساله: (علوم کامپیوتر 1400)

91. فرض کنیم L زبان عبارت منظم زیر باشد:

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

به ازای کدام مقدار برای x عبارت $w = 0101111 \times 0$ در L قرار می گیرد؟

- (1) فقط 0 (2) هر دوی 0 و 1 (3) فقط 1 (4) هیچکدام از 0 و 1

پاسخ: گزینه (1) صحیح است.

در همه رشته های عضو $L(R)$ دو الفبای صفر با فاصله $3k$ از یکدیگر وجود دارد:

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

برای $x = 0$ داریم $w = 010111100$ که دو صفر با فاصله 6 از یکدیگر دارد پس عضو $L(R)$ است.

برای $x = 1$ داریم $w = 010111110$ که شرط عضویت را ندارد.

مساله: (مهندسی کامپیوتر 1399)

51- زبان های $L_1 = \Phi$ و $L_2 = \{a\}$ را در نظر بگیرید. پاسخ $L_1 L_2^* \cup L_1^*$ برابر با کدام مورد است؟

- (1) Φ (2) a^* (3) $\{\epsilon\}$ (4) $\{\epsilon, a\}$

پاسخ: گزینه (3) صحیح است.

یادآوری کنیم که همواره $\epsilon \in L^*$ است و برای زبان تهی داریم: $\Phi^* = \{\epsilon\}$.

$$L_1^* = \Phi^* = \{\epsilon\}$$

$$L_2^* = \{a\}^* = \{\epsilon, a, a^2, \dots\}$$

$$L_1 L_2^* = \Phi \{a\}^* = \Phi$$

$$L_1 L_2^* \cup L_1^* = \Phi \cup \{\epsilon\} = \{\epsilon\}$$



8-

فصل سوم: گرامر منظم - عبارت منظم - زبان منظم

جلسه هشتم: رابطه عبارات منظم و ماشین متناهی

حل چند تمرین و تست

(مهندسی کامپیوتر 1399)

– زبان $L = \{ab, aa, baa\}$ را در نظر بگیرید. کدام رشته ها در L^* وجود دارند؟

الف: abaabaaabaa

ب: aaaabaaaa

ج: baaaaabaaaaab

د: baaaaabaa

5) الف، ب، ج (6) الف، ب، د (7) الف، ج، د (8) ب، ج، د

پاسخ: گزینه (2) صحیح است.

به جز (ج) سایر رشته ها با الحاق رشته های ab, aa, ba قاب تولیدند:

الف: abaabaaabaa

ب: aaaabaaaa

ج: baaaaabaaaaab

د: baaaaabaa

تمرین:

عبارت منظم معادل با مجموعه ذیل کدام است؟

$$L = \{a^n b^m \mid n + m \text{ زوج باشد}\}$$

$$(1) \quad a^*(ab)b^* \quad (2) \quad (aa)^*(\epsilon + ab)(bb)^*$$

$$(3) \quad (aa + bb)^*(\epsilon + ab) \quad (4) \quad (aa)^*(ab)(bb)^*$$

پاسخ: گزینه (2) صحیح است.

برای آنکه $n + m$ زوج باشد باید هردو زوج باشند یا هردو فرد باشند.

$$(a^2)^*(\epsilon)(b^2)^* + (a^2)^*(ab)(b^2)^* = (a^2)^*(\epsilon + ab)(b^2)^*$$

مساله: (مهندسی کامپیوتر 85)

کدام عبارت منظم زبان زیر را توصیف می کند؟

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

$$\begin{aligned} & a^* b^* b^* b^* c^* c^* \quad (2) \\ & aa^* bbbb^* b^* b^* ccc^* c^* \quad (4) \end{aligned}$$

$$\begin{aligned} & a^* (bbb)^* (cc)^* \quad (1) \\ & a a^* (bbb)^* bbb (cc)^* cc \quad (3) \end{aligned}$$

پاسخ: گزینه (3) صحیح است.

اگر $m, n, k \geq 0$ بودند جواب به فرم $a^* (b^3)^* (c^2)^*$ بود. اما با توجه شرط $m, n, k \geq 1$ جواب $aa^* (b^3)^* b^3 (c^2)^* c^2$ است.

مساله: (علوم کامپیوتر 1403)

اگر L زبان عبارت منظم $1^* (0+10)^*$ باشد و $\alpha = 1001010110111$ و $\beta = 0101000110011$ کدام مورد درست است؟

$$\begin{aligned} & \alpha \notin L, \beta \in L \quad (2) \\ & \alpha \notin L, \beta \notin L \quad (4) \end{aligned}$$

$$\begin{aligned} & \alpha \in L, \beta \in L \quad (1) \\ & \alpha \in L, \beta \notin L \quad (3) \end{aligned}$$

پاسخ: گزینه (4) صحیح است.

رشته های تولید شده توسط $(0+10)^*$ دارای ارقام 1 متوالی نیستند. پس هر گاه دو رقم 1 متوالی داشتیم باید تا انتها همه ارقام 1 باشند. یعنی توسط 1^* تولید شده باشند. در نتیجه هیچکدام از رشته های α و β در L قرار ندارند.

$$\begin{aligned} & 0111 \underline{11} 1001010\alpha = \\ & 0011 \underline{11} 0101000\beta = \end{aligned}$$

مسئله: (مهندسی کامپیوتر 1402)

اگر $L_1 = L(0^*1^*) \subseteq \{0,1\}^*$ و اگر $L_2 = \{w_1w_2 \mid w_2w_1 \in L_1\}$ آنگاه L_2 به وسیله کدام یک از عبارات منظم (regular expression) زیر تعریف می شود؟

$$\begin{array}{ll} 0^*1^*0^* + 1^*0^*1^* & (1) \\ 1^*0^* + 0^*1^* & (2) \\ 1^*0^*1^* & (4) \\ 0^*1^*0^* & (3) \end{array}$$

پاسخ: گزینه (1) صحیح است.

فرم کلی رشته های $L_1 = L(0^*1^*)$ را می دانیم. مثلاً رشته 00000111111 را در نظر بگیریم. این رشته به این فرم ها می تواند تفکیک شود:

$$\begin{aligned} w_1w_2 &= \underbrace{00111111}_{w_1} \underbrace{000}_{w_2} \Rightarrow w_2w_1 = \underbrace{000}_{w_2} \underbrace{00111111}_{w_1} \\ w_1w_2 &= \underbrace{111111}_{w_1} \underbrace{00000}_{w_2} \Rightarrow w_2w_1 = \underbrace{00000}_{w_2} \underbrace{111111}_{w_1} \\ w_2w_1 &= \underbrace{0000011}_{w_2} \underbrace{1111}_{w_1} \Rightarrow w_1w_2 = \underbrace{1111}_{w_1} \underbrace{0000011}_{w_2} \end{aligned}$$

پس L_2 رشته هایی با این فرم دارد:

$$L_2 = L(0^*1^*0^* + 1^*0^* + 1^*0^*1^*)$$

اما $1^*0^*1^*$ حالت خاصی از $1^*0^*1^*$ است. پس:

$$L_2 = L(0^*1^*0^* + 1^*0^*1^*)$$

قضیه:

زبان L منظم است اگر و تنها اگر زبان تولید شده توسط یک عبارت منظم باشد.

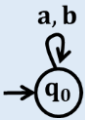
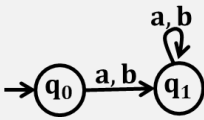
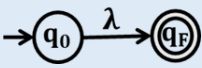
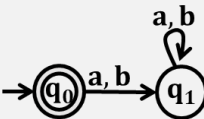
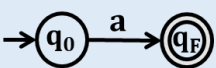
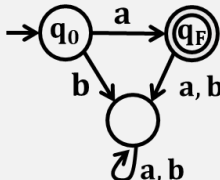
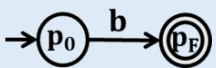
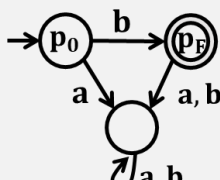
در واقع در ادامه درس متوجه می‌شویم دو حکم زیر برقرارند:

الف: برای هر عبارت منظم r ماشین متناهی M دارد وجود دارد که $L(M) = L(r)$.

ب: برای هر ماشین متناهی M عبارت منظم r دارد وجود دارد که $L(M) = L(r)$.
(گاه از علامت $M(r)$ برای این ماشین استفاده می‌کنیم).

هر عبارت منظم دارای یک ماشین متناهی معادل است:

از عبارات منظم اولیه آغاز کنیم. برای ϕ, λ, a, b ماشین nfa پذیرند به سادگی به دست می‌آید. به علاوه ما dfa پذیرنده را هم نشان داده‌ایم:

	nfa	dfa
ϕ		
λ		
a		
b		

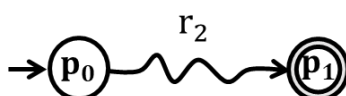
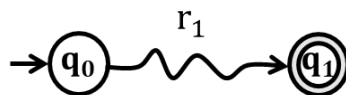
ماشین پذیرنده b مشابه پذیرنده a است.

حالا با داشتن nfa پذیرنده a و b ماشین های پذیرنده a^* , ab, $a + b$ را طراحی می کنیم.

	nfa
$a + b$	
ab	
a^*	

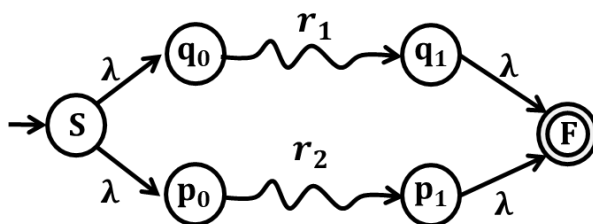
حالت کلی:

فرض کنیم برای عبارات منظم r_1 و r_2 ماشین پذیرنده متناهی داشته باشیم:

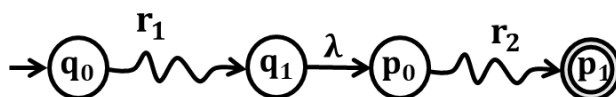


اکنون ماشین منتهای پذیرنده برای جمع، الحاق، بستار آنها به این صورت قابل رسم است:

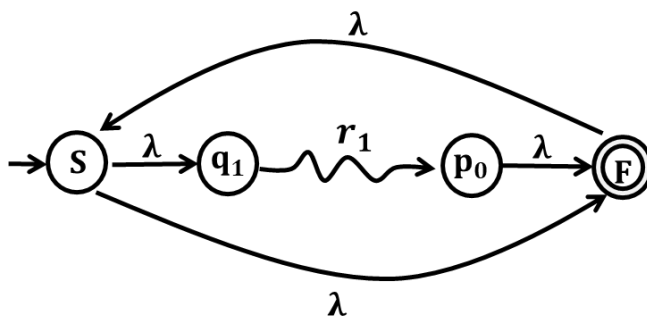
■ برای $r_1 + r_2$:



■ برای $r_1 r_2$:



■ برای r_1^* :

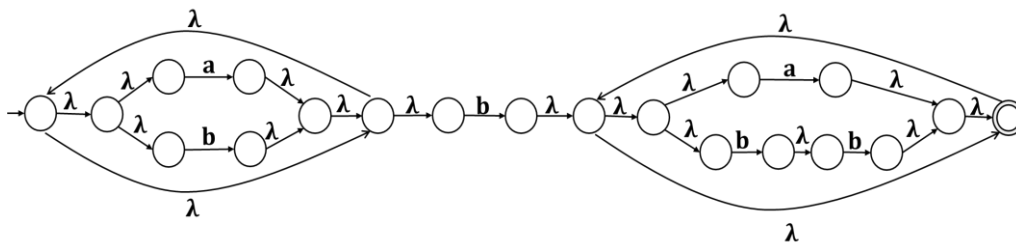


تمرین 3 بخش 3.2 (peter linz)

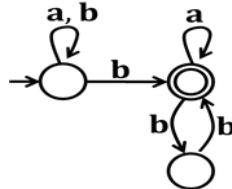
- یک nfa پذیرنده برای زبان $L((a+b)^*b(a+bb)^*)$ طراحی کنید.

پاسخ:

با ترکیب ماشین های پذیرنده این عبارات به این طرح می رسمیم:



البته این ماشین می تواند ساده تر شود. طراحی ساده تر ماشین nfa برای $L((a+b)^*b(a+bb)^*)$ چنین است:

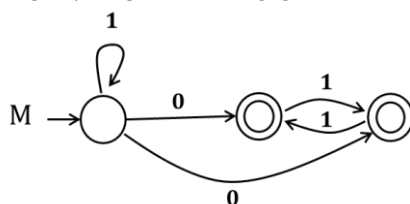


تبدیل nfa به عبارت منظم:

از گراف انتقال تعمیم یافته "(Generalized Transition Graphs)" برای یافتن عبارت منظم معادل با ماشین متناهی استفاده می کنیم.

در گراف انتقال ساده ، هر یال نشان دهنده انتقال با یک الفبا یا انتقال λ بود. اما در GTG، هر یال نشان دهنده یک رشته یا عبارت منظم است. برای شروع کار به تعداد حالات پذیرش ماشین توجه کنیم. اگر چند حالت پذیرش داشته باشیم می توانیم با انتقال λ آنها را به یک حالت کاهش دهیم یک راه هم ن است که مساله را هربار برای یکی از حالات پذیرش حل کنیم و سپس عبارات به دست آمده را جمع کنیم. در هر صورت شروع به حذف حالت های میانی می کنیم و با حذف هر حالت، انتقال هایی که به واسطه آن حالت قابل انجام بود را روی یال مربوطه می نویسیم. با ادامه این کار به مگراف ساده ای می رسمیم که عبارت منظم مربوط به آن کاملاً واضح باشد. به تمرین زیر توجه کنید.

تمرین- ماشین متناهی M داده شده است. زبان $L(M)$ معادل کدام عبارت منظم است؟



$$(0 + 1)^* 0^* 11^* \quad (2)$$

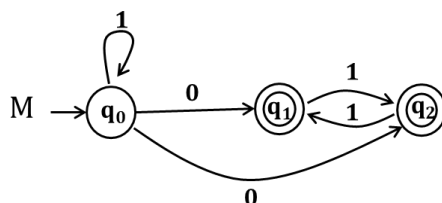
$$1^* 01^* \quad (1)$$

$$1^* 01^* + 010 \quad (4)$$

$$(0 + 01)^* 0^* \quad (3)$$

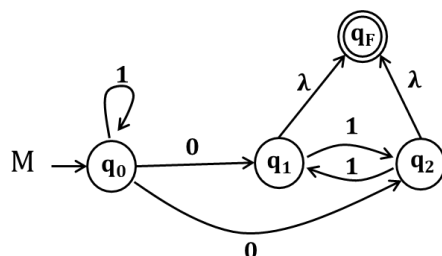
پاسخ: گزینه (1) صحیح است.

حالات را نامگذاری می‌کنیم. ماشین M دو حالت پذیرش دارد. به دو روش می‌توان این مشکل را حل کرد.

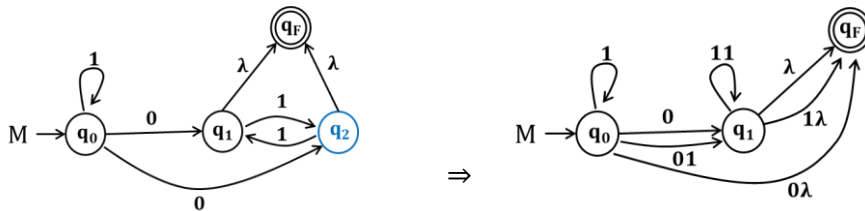


روش اول: هر بار فقط یکی از آنها را پذیرش در نظر بگیریم و مساله را برای آن حل کنیم سپس جوابها را جمع کنیم: $(r = r_1 + r_2)$.

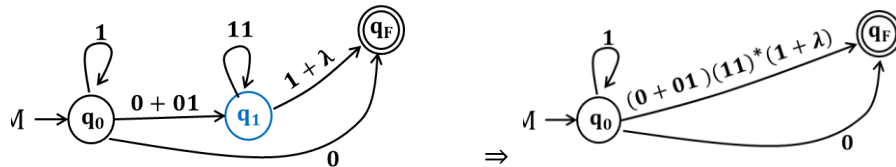
روش دوم: از انتقال λ برای آوردن هر دوی آنها به یک حالت پذیرش استفاده کنیم. ما روش دوم را به کار می‌گیریم:



حالا شروع به حذف حالات میانی کنیم. ابتدا q_2 را حذف کرده و انتقال هایی که با واسطه q_2 انجام می شد به صورت یک یال مستقیم نمایش دهیم. مسیر $q_0 q_2 q_F$ (مصرف 0λ) تبدیل به یال مستقیم از q_0 به q_F می شود. مسیر $q_0 q_2 q_1$ (مصرف 01) تبدیل به یال مستقیم از q_0 به q_1 می شود. مسیر $q_1 q_2 q_F$ (مصرف 1λ) تبدیل به یال مستقیم از q_1 به q_F می شود. مسیر $q_1 q_2 q_1$ (مصرف 11) تبدیل به طوق از q_1 به q_1 می شود.



(البته $0\lambda = 0$ و $1\lambda = 1$). به جای دو یال از q_0 به q_1 یک یال با مجموع $0 + 01$ را در نظر می گیریم. به جای دو یال از q_1 به q_F یک یال با مجموع $1 + \lambda$ را در نظر می گیریم. اکنون q_1 را حذف می کنیم:



به جای دو یال از q_0 به q_F یک یال با مجموع عبارات را در نظر می گیریم. در ضمن طوق 1 روی q_0 به صورت 1^* نوشته می شود. در نهایت به این عبارت منظم می رسیم:

$$M \rightarrow q_0 \xrightarrow{1^* (0 + (0 + 01)(11)^* (1 + \lambda))} q_F$$

عبارت منظم معادل با این ماشین $r = 1^* (0 + (0 + 01)(11)^* (1 + \lambda))$ است. با کمی دقت می توانیم این عبارت را ساده کنیم. برای شروع، به $(11)^* (1 + \lambda)$ دقت کنید. $(11)^* (\lambda)$ یعنی تعداد زوجی 1 و $(11)^* (1)$ یعنی تعداد فردی 1. پس مجموع آنها به معنای تعداد دلخواهی 1 است:

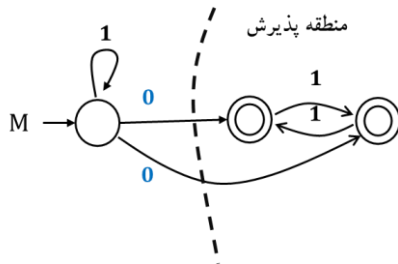
$$(11)^*(1 + \lambda) = 1^*$$

تا اینجا $r = 1^*(0 + (0 + 01)1^*)$ است. به سادگی می‌بینیم که $(0 + 01)1^*$ معادل 01^* است پس $r = 1^*(0 + 01^*)$ رشته‌های 1^*0 زیرمجموعه 1^*01^* هستند پس مجموع آنها (اجتماع آنها) همان 1^*01^* است:

$$\begin{aligned} r &= 1^*(0 + (0 + 01)(11)^*(1 + \lambda)) \\ &= 1^*(0 + (0 + 01)1^*) \\ &= 1^*(0 + 01^*) \\ &= 1^*01^* \end{aligned}$$

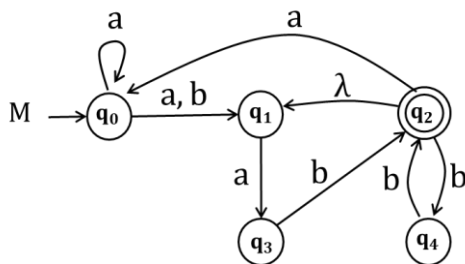
راه حل کوتاهتر:

این ماشین با هر تعداد الفبای 1 در حالت اولیه باقی می‌ماند. با اولین الفبای 0 وارد حالت پذیرش می‌شود و پس از آن هر تعداد دلخواه الفبای 1 ماشین را در پذیرش نگه می‌دارد. بنابراین عبارت منظم آن 1^*01^* است.



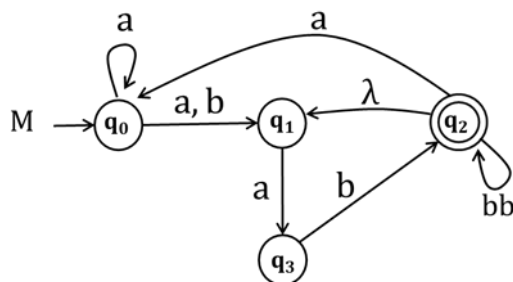
تمرین 8 بخش 3.2 (peter linz)

- عبارت منظم معادل با $L(M)$ را بیابید.

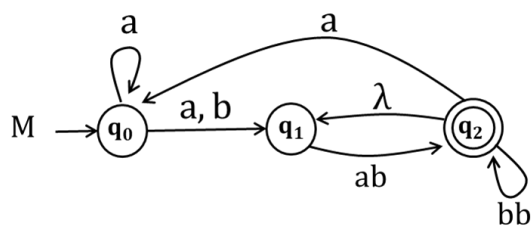


پاسخ:

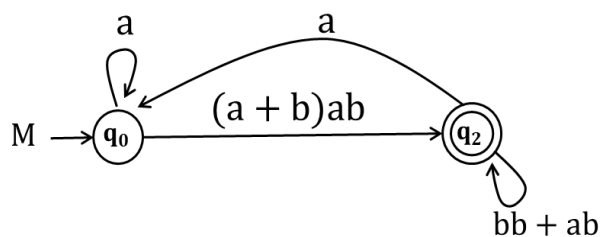
ابتدا q_4 را حذف می‌کنیم. مسیر $q_2q_4q_2$ (مصرف bb) تبدیل به q_2 به q_2 می‌شود.



حالا q_3 را حذف کنیم. مسیر $q_1q_3q_2$ تبدیل به یال مستقیم ab از q_1 به q_2 می‌شود.



حالا q_1 را حذف کنیم. مسیر $q_0q_1q_2$ تبدیل به یال مستقیم $(a + b)ab$ از q_0 به q_2 می‌شود. مسیر $q_2q_1q_2$ تبدیل به طول λab از q_2 به q_2 می‌شود یعنی این عبارت منظم به عبارت منظم bb اضافه می‌شود.



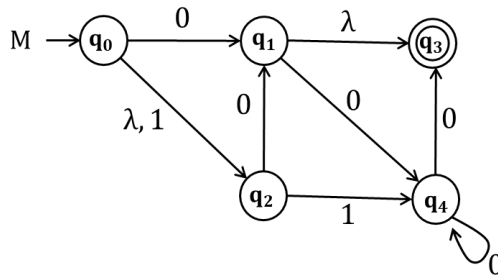
ابتدا با رشته $a^*(a+b)ab$ وارد حالت پذیرش می‌شویم. سپس با رشته های bb یا ab یا $aa^*(a+b)ab$ رفت و برگشت به پذیرش را داریم.

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

توجه: در پاسخ دادن به سوالات این مبحث، در آزمون معمولا رد گزینه های غلط سریعتر از یافتن گزینه صحیح است.

تست-ماشین متناهی M داده شده است. زبان $L(M)$ با کدام عبارت منظم معادل است؟

- (1) $0 + (00 + 1 + 11)0^*$ (2) $0 + (00 + 1 + 11)0^*$
(3) $(00 + 1 + 11)0^*$ (4) $0 + (00 + 1 + 11)00^*$

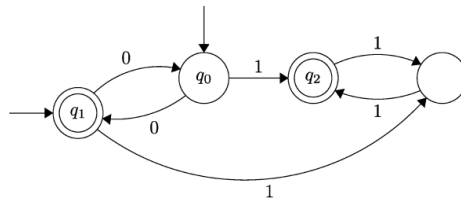


پاسخ: گزینه 4

در این ماشین هر رشته ای که پذیرفته می‌شود آخرین الفبایش حتما صفر بوده. پس در گزینه صحیح باید همه رشته ها به صفر ختم شده باشند. فقط گزینه 4 این را تضمین می‌کند.

مهندسی کامپیوتر 1402

۴۶- عبارت منظم (regular expression) توصیف کننده زبان پذیرنده متناهی زیر کدام است؟ (منظور از ϵ رشته به طول صفر است.)



- (۱) $(00)^*(0+1)(11)^*$
(۲) $(00)^*(\epsilon+0+1)(11)^*$
(۳) $(00)^*(\epsilon+01)(11)^*$
(۴) $(00)^*(\epsilon+0+1+01)(11)^*$

پاسخ: گزینه 4 صحیح است.

گزینه ها از مجموع چند عبارت منظم تشکیل شده اند که پذیرش یا عدم پذیرش هریک را در این جدول بررسی کرده ایم:

عبارت منظم	رشته تولید شده	پذیرش یا عدم پذیرش در $L(M)$
$(00)^* \varepsilon (11)^*$	زوج 1 زوج 0	بله با شروع از q_1 پذیرفته می شود
$(00)^* 0 (11)^*$	زوج 1 فرد 0	بله با شروع از q_0 پذیرفته می شود
$(00)^* 1 (11)^*$	فرد 1 زوج 0	بله با شروع از q_0 پذیرفته می شود
$(00)^* 01 (11)^*$	فرد 1 فرد 0	بله با شروع از q_1 پذیرفته می شود

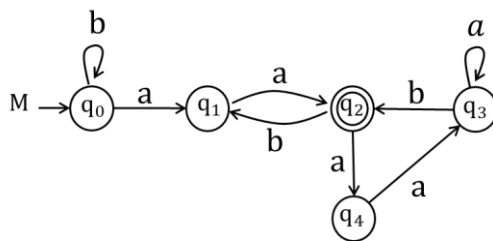
می بینیم که هر چهار نوع رشته مورد پذیرش قرار می گیرند پس پاسخ صحیح گزینه 4 است. درواقع گزینه 4 شامل تمام رشته های به فرم $0^* 1^*$ است:

$$(00)^* (\varepsilon + 0 + 1 + 01) (11)^* = 0^* 1^*$$

توجه: در بسیاری از ماشین های منتهای استفاده از اروش زیر می تواند به نوشتن عبارت منظم معادل با ماشین کمک کند. فرض کنید q_F حالت پذیرش باشد ابتدا رشته r_0 را که با آن برای اولین بار به حالت پذیرش می رسیم پیدا کنیم. حالا تمام حالات رفت و برگشت به حالت پذیرش را در نظر بگیرید اگر با عبارات r_1 و r_2 رفت و برگشت به انجام می شود، عبارت منظم معادل با ماشین به فرم $r_0(r_1 + r_2)^*$ است.

مثال:

عبارت منظم معادل با M را بیابید.



فقط q_2 حالت پذیرش است، با عبارت $b^* a^2$ برای اولین بار به حالت پذیرش وارد می شویم. اکنون با رشته های ba و $a^2 a^* b$ رفت و برگشت به q_2 انجام می شود. عبارت منظم معادل با ماشین به این فرم است.

$$r = b^* a^2 (ba + a^2 a^* b)^*$$

فصل سوم: گرامر منظم - عبارت منظم - زبان منظم
جلسه هشتم بخش دوم: رابطه گرامر منظم و ماشین متناهی

گرامر منظم (regular grammar)

گرامر G خطی راست گفته می شود که همه قواعد تولید به این فرم باشند:

$$\begin{aligned} A &\rightarrow xB & (x \in T^*) \\ A &\rightarrow x \end{aligned}$$

گرامر G خطی چپ گفته می شود که همه قواعد تولید به این فرم باشند:

$$\begin{aligned} A &\rightarrow Bx & (x \in T^*) \\ A &\rightarrow x \end{aligned}$$

گرامر خطی راست یا خطی چپ را **گرامر منظم** گوئیم.

مثال:

به گرامرهای G_1 و G_2 توجه کنید:

$$\begin{aligned} G_1: \\ S &\rightarrow abS \mid a \end{aligned}$$

$$\begin{aligned} G_2: \\ S &\rightarrow Aab \\ A &\rightarrow Aab \mid B \\ B &\rightarrow a \end{aligned}$$

G_1 و G_2 هر دو منظم هستند. G_1 خطی راست و G_2 خطی چپ است.

$$L(G_1) = L((ab)^*a) \quad L(G_2) = L(aab(ab)^*) = L(a(ab)^+)$$

مثال:

گرامر G_3 خطی است اما منظم نیست. چون برخی قواعدش خطی راست و برخی خطی چپ هستند.

$$\begin{aligned} G_3: \\ S &\rightarrow A \\ A &\rightarrow aB \mid \lambda \\ B &\rightarrow Ab \end{aligned}$$

$$L(G_3) = \{a^n b^n \mid n \geq 0\}$$

توجه:

اگر گرامر G منظم باشد، زبان $L(G)$ منظم است.

اگر گرامر G نامنظم باشد، زبان $L(G)$ ممکن است منظم باشد یا نباشد.

اگر $L(G)$ منظم باشد گرامر منظم G' وجود دارد که $L(G) = L(G')$.

در مثال فوق G_3 نامنظم است و زبان $L(G_3)$ هم نامنظم است. پس این گرامر، معادل منظم ندارد.

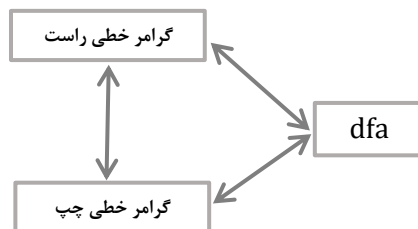
قضیه: هر زبان منظم گرامر خطی راست دارد.

اثبات: می دانیم که:

زبان L منظم است $\Leftrightarrow$ ماشین متناهی قطعی M وجود داشته باشد که $L = L(M)$.

پس کفایت نشان دهیم برای هر ماشین متناهی قطعی M ، گرامر خطی راست G وجود دارد که:
 $L(G) = L(M)$

در حقیقت در ادامه درس خواهیم دید که این تبدیلات قابل انجام است:

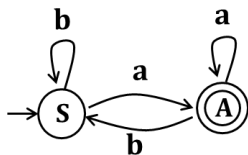


یافتن گرامر خطی راست معادل dfa:

هر ماشین dfa یک گرامر خطی راست معادل دارد. مطابق جدول زیر انتقالها و حالات ماشین قطعی به قواعد تولید گرامر خطی راست تبدیل می شوند:

حالات و انتقالات ماشین dfa	متغیرها و قواعد گرامر خطی راست
$\rightarrow S$ حالت اولیه	S متغیر آغاز گرامر
$A \xrightarrow{a} B$	$A \rightarrow aB$
$A \xrightarrow{a} A$	$A \rightarrow aA$
$\textcircled{B}$ حالت پذیرش	$B \rightarrow \lambda$

مثال: گرامر خطی راست معادل با این ماشین متناهی را بنویسید.

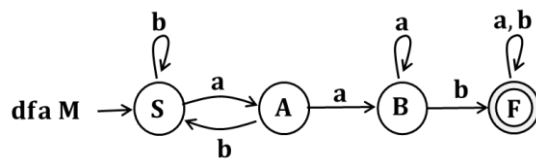


پاسخ: این ماشین dfa است. پس گرامر معادل آن چنین است:

G:

$$\begin{aligned} S &\rightarrow bS \mid aA \\ A &\rightarrow aA \mid bS \mid \lambda \end{aligned}$$

مثال: در جلسات پیشین برای زبان L که شامل تمام رشته های دارای زیر رشته aab است، ماشین پذیرنده dfa یافته ایم. با توجه به این ماشین، یک گرامر منظم برای این زبان پیدا کنید.



$$L = L(M) = \Sigma^* aab \Sigma^*$$

پاسخ: این dfa قابل تبدیل به گرامر خطی راست G است. حالت S اولیه است پس با متغیر S در گرامر آغاز می کنیم. هر انتقال را مطابق الگو به یک قاعده تولید تبدیل می کنیم. حالت F پذیرش است پس انتقال $F \rightarrow \lambda$ را به گرامر اضافه کنیم.

G:

$$\begin{aligned} S &\rightarrow bS \mid aA \\ A &\rightarrow aB \mid bS \\ B &\rightarrow aB \mid bF \\ F &\rightarrow aF \mid bF \mid \lambda \end{aligned}$$

یافتن ماشین متناهی معادل با گرامر خطی راست:

همانطور که ماشین dfa به گرامر خطی راست قابل تبدیل است. گرامر خطی راست نیز به ماشین متناهی nfa و dfa تبدیل می‌شود. جدول زیر به شکل کاملتری جزئیات این کار را نشان می‌دهد.

گرامر خطی راست	توضیحات	ماشین متناهی
S متغیر آغاز گرامر		$\rightarrow$ حالت اولیه (S)
$A \rightarrow aB$		$(A) \xrightarrow{a} (B)$
$A \rightarrow aA$		$(A) \xrightarrow{a} (A)$
$B \rightarrow \lambda$		حالت پذیرش (B)
$A \rightarrow abaB$	$A \rightarrow aX$ $X \rightarrow bY$ $Y \rightarrow aB$	$(A) \xrightarrow{a} (X) \xrightarrow{b} (Y) \xrightarrow{a} (B)$
$A \rightarrow a$	$A \rightarrow aZ$ $Z \rightarrow \lambda$	$(A) \xrightarrow{a} (Z)$

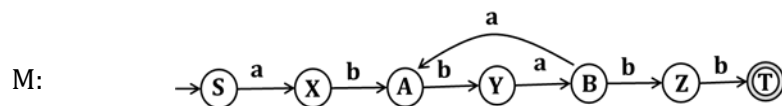
مثال: گرامر G داده شده است. ماشین متناهی M را پیدا کنید که $L(G) = L(M)$ باشد.

G :
 $S \rightarrow abA$
 $A \rightarrow baB$
 $B \rightarrow aA \mid bb$

پاسخ: می‌توانیم ابتدا گرامر را به گرامر معادل G' تبدیل کنیم.

G' :
 $S \rightarrow aX$
 $X \rightarrow bA$
 $A \rightarrow bY$
 $Y \rightarrow aB$
 $B \rightarrow aA \mid bZ$
 $Z \rightarrow bT$
 $T \rightarrow \lambda$

متغیر آغاز S به حالت اولیه ماشین تبدیل می‌شود. قاعده $S \rightarrow aX$ به انتقال $\delta(S, a) = X$ تبدیل می‌شود. به همین ترتیب ادامه می‌دهیم و در نهایت چون متغیر T اشتقاق $T \rightarrow \lambda$ دارد پس حالت T حالت پذیرش (نهایی) است.



گرامر های G و G' و ماشین متناهی غیرقطعی M معادل هستند:

$$L(G) = L(G') = L(M) = \{ab(baa)^nbabb \mid n \geq 0\}$$

توجه: وقتی گرامر خطی راست را به ماشین متناهی تبدیل می‌کنیم ماشین به دست آمده ممکن است nfa باشد. سپس می‌توانیم آن را به dfa معادل تبدیل کنیم.

قضیه: هر گرامر خطی چپ یک گرامر معادل خطی راست دارد.

اثبات: $L(G)$ منظم است. پس پذیرنده dfa دارد. این dfa دارای گرامر معادل خطی راست است. پس یک گرامر خطی راست معادل با G وجود دارد.

توجه: فرض کنیم G گرامر خطی چپ باشد. اگر از همه قواعد تولید، وارون بگیریم، گرامر $\hat{G}$ به دست می‌آید که خطی راست است و زبان تولید شده توسط آن $L(G)^R$ است:

$$L(\hat{G}) = L(G)^R$$

$G:$	$\hat{G}:$
$A \rightarrow Bx$	$A \rightarrow x^R B$
$A \rightarrow x$	$A \rightarrow x^R$

مثال: به گرامر G توجه کنید. هنگامی که از قواعد تولید آن وارون بگیریم، گرامر $\hat{G}$ به دست می‌آید. به وضوح زبان $L(\hat{G})$ وارون زبان $L(G)$ است.

$G:$	$\hat{G}:$
$S \rightarrow Sab A$	$S \rightarrow baS A$
$A \rightarrow Ac c$	$A \rightarrow cA c$

$$L(G) = \{c^n (ab)^{m+1} \mid n \geq 0\}$$

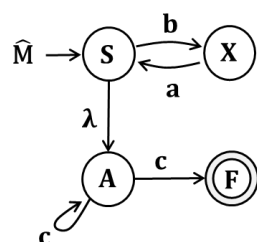
$$L(\hat{G}) = \{(ba)^{m+1} c^n \mid n \geq 0\}$$

$$L(\hat{G}) = L(G)^R \text{ می‌بینیم که:}$$

تمرین: در ادامه مثال قبل به عنوان تمرین ماشین‌های متناهی پذیرنده $L(G)$ و $L(\hat{G})$ را رسم می‌کنیم. $\hat{G}$ گرامر خطی راست است. ابتدا قواعد تولید آن را به این صورت بازنویسی می‌کنیم:

$$\begin{aligned} \hat{G}: \\ S &\rightarrow bX \mid \lambda A \\ X &\rightarrow aS \\ A &\rightarrow cA \mid cF \\ F &\rightarrow \lambda \end{aligned}$$

ماشین متناهی معادل با این گرامر را رسم کنیم:



برای یافتن ماشین متناهی معادل با گرامر خطی چپ G ، ابتدا رشته های عضو $L(G)$ تشخیص می دهیم. طراحی می کنیم.

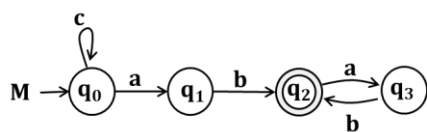
G :

$S \rightarrow Sab|A$
 $A \rightarrow Ac|c$

عبارت منظم معادل با $L(G)$ چنین است:

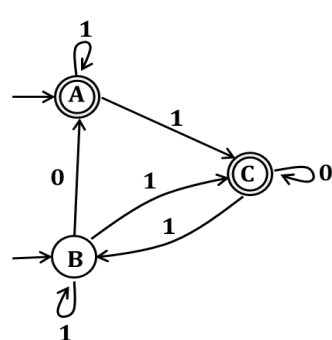
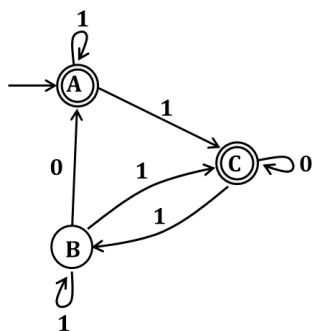
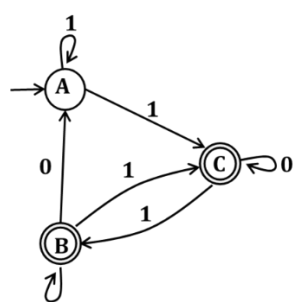
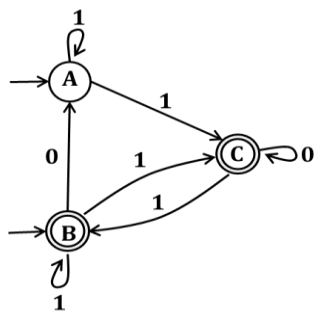
$$r = c^*ab(ab)^* = c^*(ab)^+$$

ماشین متناهی پذیرنده این عبارت را رسم می کنیم:



مسئله: گرامر $G = (\{0,1\}, \{S, A, B, C\}, S, P)$ داده شده است. کدام ماشین متناهی زیر پذیرنده $L(G)$ است؟

G :
 $S \rightarrow A \mid B \mid \lambda$
 $A \rightarrow 1A \mid 1$
 $B \rightarrow 1C \mid 0A \mid 1B \mid 1$
 $C \rightarrow 1B \mid 0C \mid 0 \mid 1$



پاسخ: گزینه 2 صحیح است.

برخی رشته های تولید شده توسط گرامر این ها هستند.

λ
 11^*
 011^*
 $011^* 11^*$

100^*1

برای مثال λ و 1000001 و 111 باید پذیرفته شوند پس گزینه 2 درست است.



فصل چهارم: ویژگیهای بستاری زبانهای منظم
جلسه نهم بخش یک: ویژگی های بستاری زبان منظم

-9-

قضیه:

اگر L_1 و L_2 منظم باشند آنگاه این زبانها منظم هستند:

$$L_1 \cup L_2, L_1 \cap L_2, L_1 - L_2, \overline{L_1} = L_1^c, L_1 L_2, L_1^n, L_1^*, L_1^R, h(L_1)$$

زبانهای منظم تحت اجتماع، اشتراک، تفاضل، مکمل، الحاق، بستار، وارون و همریختی بسته هستند.
در ادامه این موارد را توضیح می دهیم.

تمرین 26 بخش 4.1 (peter linz)

گرامرهای منظم G_1 و G_2 را داریم.

$$G_1: \\ S_1 \rightarrow aS_1 \mid bA \mid \lambda \\ A \rightarrow bA \mid ba$$

$$G_2: \\ S_2 \rightarrow bA \\ A \rightarrow aS_2 \mid a$$

گرامرهایی برای تولید این این سه زبان ارائه دهید. آیا می توانید گرامر منظم برای آنها ارائه دهید؟

الف: $L(G_1) \cup L(G_2)$

ب: $L(G_1)L(G_2)$

ج: $L(G_1)^*$

پاسخ: ابتدا نام متغیرهای مشترک را تغییر دهیم تا گرامرها متغیر مشترک نداشته باشند. فرض کنیم متغیرهای آغازین در G_1 و G_2 به ترتیب S_1 و S_2 باشند.

$$\begin{array}{ll} \mathbf{G_1:} & \mathbf{G_2:} \\ S_1 \rightarrow aS_1 \mid bA \mid \lambda & S_2 \rightarrow bA \\ A \rightarrow bA \mid ba & A \rightarrow aS_2 \mid a \end{array}$$



$$\begin{array}{ll} \mathbf{G_1:} & \mathbf{G_2:} \\ S_1 \rightarrow aS_1 \mid bA \mid \lambda & S_2 \rightarrow bB \\ A \rightarrow bA \mid ba & B \rightarrow aS_2 \mid a \end{array}$$

حالا گرامرهای تولید کننده اجتماع، الحاق، و بستار را تعریف می‌کنیم:

$$\text{الف: } L(G_1) \cup L(G_2)$$

فرض کنیم گرامر G همه قواعد تولید گرامرهای G_1 و G_2 را دارد اما متغیر آغاز S و قاعده $S \rightarrow S_1 \mid S_2$ را به آن اضافه می‌کنیم. مجموعه قواعد تولید G برابر است با:

$$P = P_1 \cup P_2 \cup \{S \rightarrow S_1 \mid S_2\}$$

به وضوح این گرامر $L(G_1) \cup L(G_2)$ را تولید می‌کند. اگر G_1 و G_2 هر دو خطی راست باشند G نیز خطی راست است. یعنی اجتماع دو زبان منظم، منظم است.

$$\begin{array}{l} \mathbf{G:} \\ \mathbf{S \rightarrow S_1 \mid S_2} \\ S_1 \rightarrow aS_1 \mid bA \mid \lambda \\ A \rightarrow bA \mid ba \\ S_2 \rightarrow bB \\ B \rightarrow aS_2 \mid a \end{array}$$

ب: $L(G_1)L(G_2)$

فرض کنیم گرامر G همه قواعد G_1 و G_2 را همراه متغیر آغاز S و قاعده $S \rightarrow S_1S_2$ داشته باشد.

$$P = P_1 \cup P_2 \cup \{S \rightarrow S_1S_2\}$$

این گرامر $L(G_1)L(G_2)$ را تولید می‌کند:

G :

$$S \rightarrow S_1S_2$$

$$S_1 \rightarrow aS_1 \mid bA \mid \lambda$$

$$A \rightarrow bA \mid ba$$

$$S_2 \rightarrow bB$$

$$B \rightarrow aS_2 \mid a$$

$$L(G) = L(G_1)L(G_2)$$

اما قاعده $S \rightarrow S_1S_2$ گرامر G را غیرخطی و نامنظم می‌کند. برای یافتن یک گرامر منظم به جای S از S_1 برای متغیر آغاز استفاده می‌کنیم اما در تمام قواعد پایانی گرامر G_1 ، متغیر S_2 را الحاق می‌کنیم. حالا G خطی راست و در نتیجه منظم است.

G :

$$S_1 \rightarrow aS_1 \mid bA \mid \lambda S_2$$

$$A \rightarrow bA \mid baS_2$$

$$S_2 \rightarrow bB$$

$$B \rightarrow aS_2 \mid a$$

$$L(G) = L(G_1)L(G_2)$$

نتیجه: الحاق دو زبان منظم، منظم است.

ج: $L(G_1)^*$

گرامر G شامل همه قواعد G_1 باشد به علاوه‌ی متغیر شروع S و قواعد تولید $S \rightarrow SS \mid S_1 \mid \lambda$. این گرامر $L(G_1)^*$ را تولید می‌کند.

G :

$S \rightarrow SS \mid S_1 \mid \lambda$

$S_1 \rightarrow aS_1 \mid bA \mid \lambda$

$A \rightarrow bA \mid \lambda$

$$L(G) = L(G_1)^*$$

البته قاعده $S \rightarrow SS$ گرامر G را غیرخطی و نامنظم می‌کند. می‌توانیم به جای آن، به سمت راست همه قواعد پایانی، متغیر S را الحاق کنیم. حالا یک گرامر منظم برای $L(G_1)^*$ داریم که خطی راست است.

G :

$S \rightarrow S_1 \mid \lambda$

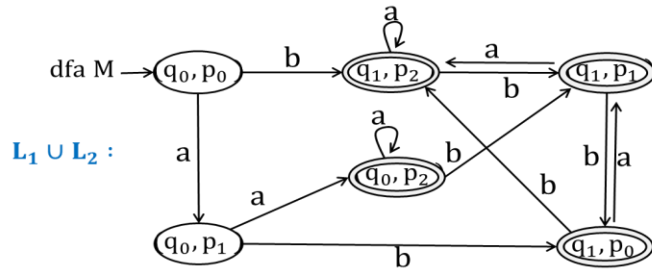
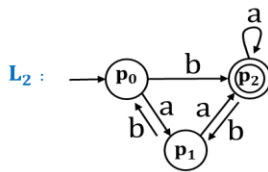
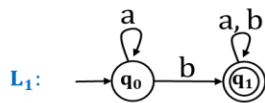
$S_1 \rightarrow aS_1 \mid bA \mid \lambda S$

$A \rightarrow bA \mid \lambda S$

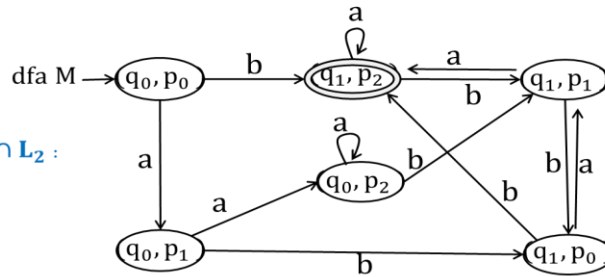
$$L(G) = L(G_1)^*$$

نتیجه: بستار یک زبان منظم، منظم است.

یادآوری: در جلسات پیشین دیدیم که اگر ماشین منتهای پذیرنده داشته باشند می‌توانیم برای اجتماع، اشتراک، مکمل، وارون، تفاضل و بستار آنها ماشین منتهای پذیرنده یافت.

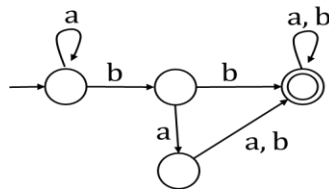


$L_1 \cap L_2$:



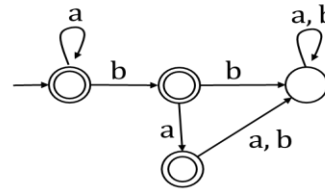
مثال برای وارون و مکمل:

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

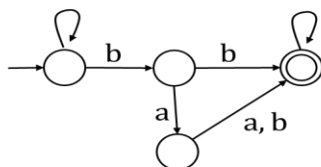


dfa M: $L(M) = L$

مکمل

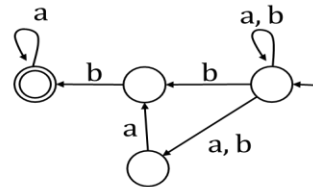


dfa M: $L(M') = \bar{L}$



dfa M: $L(M) = L$

وارون



nfa M'': $L(M'') = L^R$

پیشوندها و پسوندها: (prefixes and suffixes)

برای هر رشته $w = a_1a_2 \cdots a_n$ ، پیشوندها و پسوندها عبارتند از:

پیشوندها	پسوندها
λ	λ
a_1	a_n
a_1a_2	$a_{n-1}a_n$
$a_1a_2a_3$	$a_{n-2}a_{n-1}a_n$
$a_1a_2 \cdots a_n$	$a_1a_2 \cdots a_{n-1}a_n$

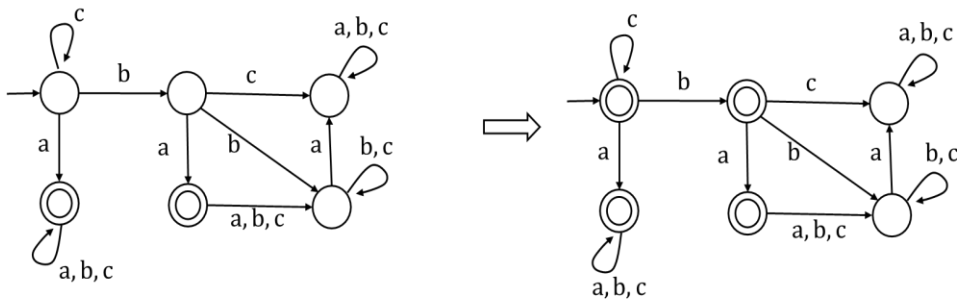
مجموعه همه پیشوندهای رشته های L : $\text{pre}(L) = \text{head}(L)$

مجموعه همه پسوندهای رشته های L : $\text{suff}(L) = \text{tail}(L)$

اگر L منظم باشد این دو زبان هم منظم هستند. ماشین پذیرنده L را M بنامیم. میخواهیم ماشین پذیرنده برای پیشوندها و پسوندها را معرفی کنیم.

ماشین پذیرنده $\text{pre}(L)$: فرض کنیم M ماشین متناهی پذیرنده L باشد. ماشین M_1 همان ماشین است با این تفاوت که حالات پذیرش آن به این صورت انتخاب شوند: هر حالتی که از آن مسیری برای رسیدن حالت پذیرش M وجود دارد، حالت پذیرش M_1 باشد. در این صورت:

$$L(M_1) = \text{pre}(L)$$



ماشین متناهی پذیرنده L

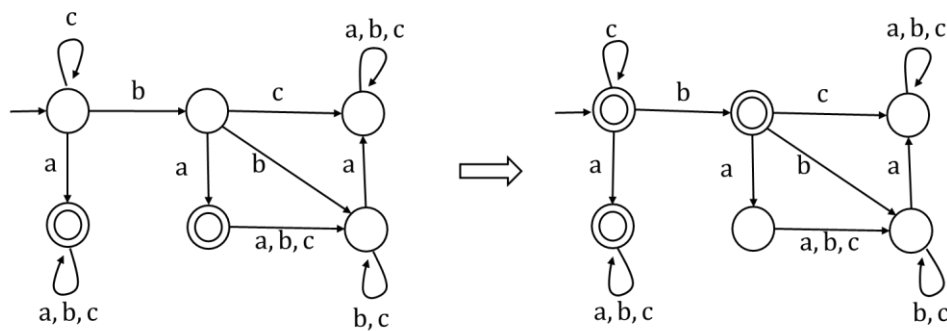
$$L(M) = L$$

ماشین متناهی پذیرنده $\text{pre}(L)$

$$L(M_1) = \text{pre}(L)$$

ماشین پذیرنده $\text{deletlast}(L)$: زبان $\text{deletlast}(L)$ مجموعه همه رشته‌هاییست که از حذف یک الفبا از انتهای اعضای L به دست می‌آیند. در واقع $\text{deletlast}(L) = \frac{L}{\Sigma^*}$. فرض کنیم M ماشین متناهی پذیرنده L باشد. ماشین M_2 همان ماشین M است با این تفاوت که حالات پذیرش آن به این صورت انتخاب شوند: هر حالتی که از آن با مصرف دقیقاً یک الفبا به حالت پذیرش M برسیم حالت پذیرش M_2 باشد. در این صورت:

$$L(M_2) = \text{deletlast}(L)$$



ماشین متناهی پذیرنده L

ماشین متناهی پذیرنده $\text{deletlast}(L)$

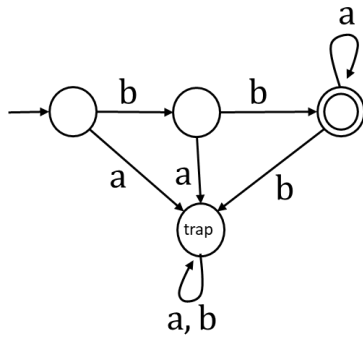
ماشین پذیرنده‌ی $\text{suff}(L)$: فرض کنیم M ماشین متناهی پذیرنده L باشد. ماشین M_3 همان ماشین M است با این تفاوت که از حالت اولیه به سایر حالات، انتقال λ وجود دارد. در این صورت:

$$L(M_3) = \text{suff}(L)$$

همچنین می‌توان از این نکته استفاده کرد:

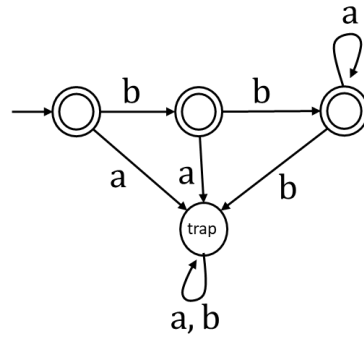
$$\text{suff}(L) = \left(\text{pre}(L^R) \right)^R$$

مثال: ماشین متناهی پذیرنده‌ی L و ماشین متناهی پذیرنده پیشوندهای L :



dfa for L

$$L(M) = \{b^2a^n \mid n \geq 0\}$$



dfa for $\text{pre}(L)$

فصل چهارم: ویژگیهای بستاری زبانهای منظم

جلسه نهم بخش دو: خارج قسمت راست و چپ

خارج قسمت راست:

فرض کنید L_1 و L_2 دو زبان دلخواه باشند. خارج قسمت راست آنها چنین تعریف می‌شود:

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

خارج قسمت چپ نیز به صورت مشابه تعریف می‌شود و در آن باید $yx \in L_1$ باشد.

$$L_1 \backslash L_2 = \{x \mid yx \in L_1 \text{ for some } y \in L_2\}$$

دو نمونه از خارج قسمت راست:

$$\begin{array}{ll} L_1 = \{abb, bbba, baa\} & \\ L_2 = \{ba, a, aa\} & \Rightarrow L_1/L_2 = \{bb, bbb, ba, b\} \end{array}$$

$$\begin{array}{ll} L_1 = \{ab^n a^n \mid n \geq 1\} & \\ L_2 = \{a^n \mid n \geq 1\} & \Rightarrow L_1/L_2 = \{ab^n a^m \mid n \geq 1, m \leq n-1\} \end{array}$$

مثال ۱: خارج قسمت راست و چپ را بیابید.

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

$$L_2 = \{a^n \mid n \geq 1\}$$

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

پاسخ: هیچکدام از اعضای L_2 پسوند هیچ عضایی از L_1 نیستند. پس خارج قسمت راست تهی است.

$$L_1/L_2 = \emptyset$$

البته اعضای L_2 پیشوند رشته های L_1 هستند. پس خارج قسمت چپ تهی نیست:

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

مثال ۲: خارج قسمت راست را بیابید.

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

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

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

پاسخ: با توجه به اعضای L_1 و L_2 :

$$L_1: \quad ab \quad a^2b^2 \quad a^3b^3 \quad a^4b^4 \quad \dots$$

$$L_2: \quad \lambda \quad b \quad b^2 \quad b^3 \quad b^4 \quad \dots$$

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

توجه:

$$\lambda \in L_2 \Rightarrow L_1 \subseteq \frac{L_1}{L_2}$$

مثال ۳: خارج قسمت راست را بیابید. با کدام مجموعه برابر است؟ $\frac{L}{\Sigma^*} = ?$

پاسخ: چون Σ^* شامل تمام رشته‌های متناهی است پس تمام پسوندهای اعضای L در آن قرار دارند و با حذف آنها مجموعه تمام پیشوندها به دست می‌آید.

$$\frac{L}{\Sigma^*} = \text{pre}(L) = \text{head}(L)$$

مثال ۴: peter linz

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

$$L_2 = \{b^m \mid m \geq 1\}$$

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

پاسخ: با توجه به فرم اعضای دو زبان داریم:

L_1 :	ba	a	a^2	a^3
		ab^1	a^2b^1	a^3b^1
		ab^2	a^2b^2	a^3b^2
		ab^3	a^2b^3	a^3b^3
L_2 :				
	b^1			
	b^2			
	b^3			
	b^4			

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

در این مثال دیدیم که گاهی $\frac{L_1}{L_2} \subseteq L_1$ است.**قضیه:** اگر L_1 منظم باشد، $\frac{L_1}{L_2}$ منظم است (حتی اگر L_2 منظم نباشد).

اگر L_1 منظم باشد ماشین متناهی پذیرنده دارد. فرض کنیم M ماشین متناهی پذیرنده L_1 باشد. ماشین M' همان ماشین M باشد با این تفاوت که حالات پذیرش آن چنین انتخاب شوند: هر حالتی که از آن با مصرف رشته‌ای از L_2 به حالت پذیرش M می‌رسیم حالت پذیرش M' باشد. در اینصورت:

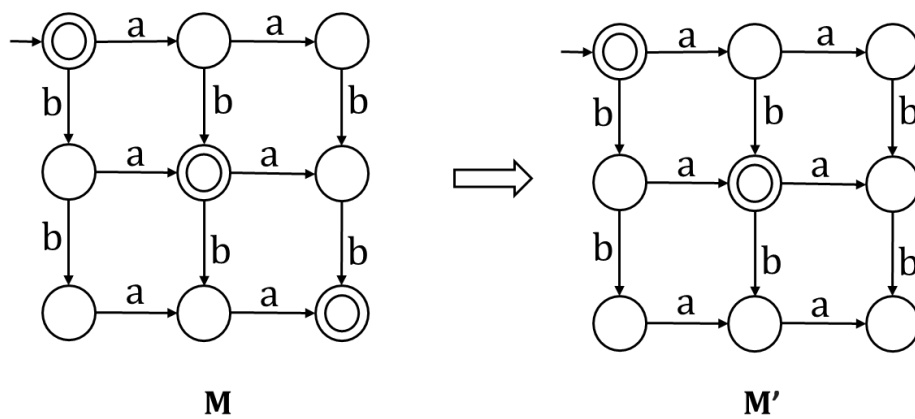
$$L(M') = \frac{L_1}{L_2}$$

مثال: زبان های L_1 و L_2 داده شده اند. ماشین متناهی M پذیرنده L_1 است. ماشین متناهی پذیرنده L_1/L_2 را بیابید. از آنجا اعضای خارج قسمت را بنویسید.

$$L_1 = \{w \mid n_a(w) = n_b(w) \leq 2\}$$

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

پاسخ:



$$L(M') = \frac{L_1}{L_2} = \{\lambda, ab, ba\}$$

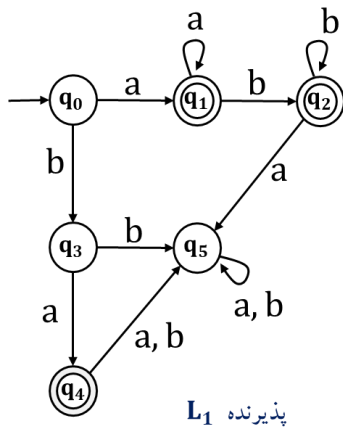
تمرین: مجموعه $\frac{L_1}{L_2}$ را با رسم ماشین پذیرنده آن بیابید.

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

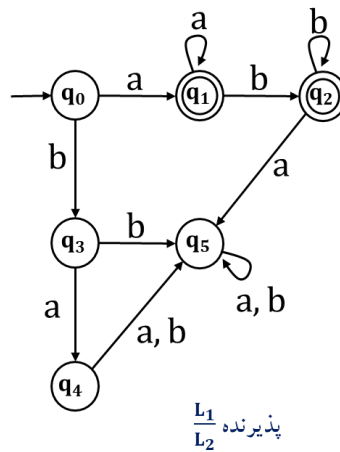
$$L_2 = \{b^m \mid m \geq 1\}$$

پاسخ:

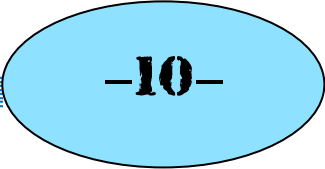
ابتدا ماشین پذیرنده L_1 را طراحی کنیم. سپس حالتی که از آنها با مصرف یک رشته‌ی L_2 به پذیرش می‌رسیم را در ماشین جدید حالت پذیرش فرض کنیم.



$\Rightarrow$



$$\Rightarrow \frac{L_1}{L_2} = \{a^n b^m \mid n \geq 1, m \geq 0\}$$



–10–

فصل چهارم: ویژگیهای بستاری زبان‌های منظم

جلسه دهم بخش یک: تکمیل بحث خارج قسمت + همریختی

سایر خارج قسمت ها:

خارج قسمت راست چنین تعریف می شود:

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

خارج قسمت چپ چنین تعریف می شود:

$$L_1 \backslash L_2 = \{x \mid yx \in L_1 \text{ for some } y \in L_2\}$$

در نتیجه:

$$L/\Sigma^* = \text{pre}(L)$$

$$L \backslash \Sigma^* = \text{suf}(L)$$

$$L/\Sigma = \text{deletlast}(L)$$

$$L \backslash \Sigma = \text{deletfirst}(L)$$

مثال: peter linz

خارج قسمت راست $\frac{L_1}{L_2}$ را تعیین کنید و برایش ماشین متناهی پذیرنده بیابید.

$$L_1 = L(a^*baa^*)$$

$$L_2 = L(a b^*)$$

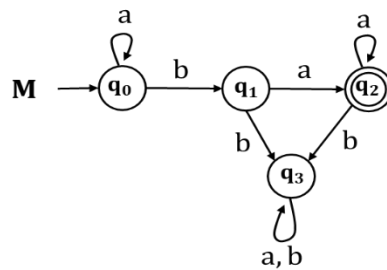
پاسخ:

اولا با توجه به آنکه $a \in L_2$ است با حذف آن از انتهای اعضای L_1 داریم:

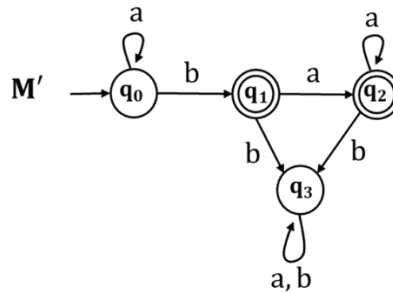
$$\frac{L_1}{L_2} = L(a^*b a^*)$$

ماشین پذیرنده L_1 را با توجه به عبارت منظم طراحی کنیم:

$$L(M) = L_1$$



ماشین M' دارای همان حالات و همان تابع انتقال باشد اما حالات پذیرش آن به این صورت انتخاب شوند: هر حالتی که از آن با مصرف یک رشته از L_2 به حالت پذیرش M می‌رسیم حالت پذیرش M' باشد. a عضو L_2 است. در ماشین M از q_1 با مصرف a همچنین از q_2 با مصرف a می‌توان به پذیرش رسید پس این دو حالت در حالت پذیرش هستند:



$$L(M') = \frac{L_1}{L_2} = L(a^* b a^*)$$

چند ویژگی خارج قسمت راست:

$$\lambda \in L_2 \Rightarrow L_1 \subseteq \frac{L_1}{L_2} \quad \text{نکته ۱:}$$

البته عکس آن درست نیست:

$$\lambda \in L_2 \not\Rightarrow L_1 \subseteq \frac{L_1}{L_2}$$

مثال نقض:

برای زبان‌های $L_2 = \{a\}$, $L_1 = L(a^*)$ داریم $L_1 \subseteq \frac{L_1}{L_2}$ در حالیکه $\lambda \notin L_2$.

$$L_2 \cap \text{suff}(L_1) = \emptyset \Leftrightarrow \frac{L_1}{L_2} = \emptyset \quad \text{نکته ۲:}$$

دقت کنید اگر یکی از زبان های L_1 یا L_2 تهی باشد، خارج قسمت تهی است. اما برای تهی بودن خارج قسمت لازم نیست حتما یکی از زبان ها تهی باشد.

مثلا برای زبان های ناتهی $L_1 = L(a^*)$ و $L_2 = L(b^+)$ نیز خارج قسمت تهی است: $\frac{L_1}{L_2} = \emptyset$

$$L_2 \cap L_1 \neq \emptyset \Leftrightarrow \lambda \in \frac{L_1}{L_2} \quad \text{نکته ۳:}$$

تست: کدام گزینه در مورد خارج قسمت راست نادرست است؟ (مهندسی کامپیوتر ۹۰)

$L_1 \subseteq \frac{L_1}{L_2} \Leftrightarrow \lambda \in L_2$ (۱)
$L_1 = \{\} \vee L_2 = \{\} \Leftrightarrow \frac{L_1}{L_2} = \{\}$ (۲)
$\lambda \in \frac{L_1}{L_2} \Leftrightarrow L_1 \cap L_2 \neq \{\}$ (۳)
$\frac{L_1}{L_2} = \{\} \Leftrightarrow L_2 = \{\}$ (۴)

پاسخ: گزینه ۲

برای تهی بودن خارج قسمت لازم نیست حتما یکی از زبان ها تهی باشد. مثلا اگر $L_1 = \{ab\}$ و $L_2 = \{aa\}$ آنگاه $\frac{L_1}{L_2} = \emptyset$.

جایگشت رشته های زبان منظم:

فرض کنید f تابعی باشد که برای هر رشته $w = a_1 a_2 \cdots a_n$ جایگشت مشخصی از آن را تولید می کند. برای مثال:

$$f(w) = f(a_1 a_2 \cdots a_{n-1} a_n) = a_n a_1 a_2 \cdots a_{n-1}$$

اگر L منظم باشد $f(L)$ نیز منظم است.

توجه:

فرض کنیم L_1 و L_2 منظم باشند و رشته های L_3 از بُر خوردن رشته های $w_1 \in L_1$ و $w_2 \in L_2$ به دست آمده اند آنگاه L_3 منظم است. (موضوع تست علوم کامپیوتر ۱۴۰۳ در همین جلسه).

$$L_1 = \{ \dots, \text{babac}, \dots \}$$

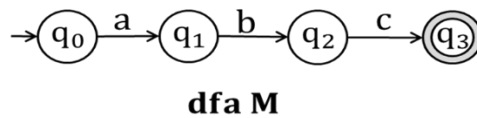
$$L_3 = \{ \dots, \text{abacbac}, \dots \}$$

$$L_2 = \{ \dots, \text{ac}, \dots \}$$

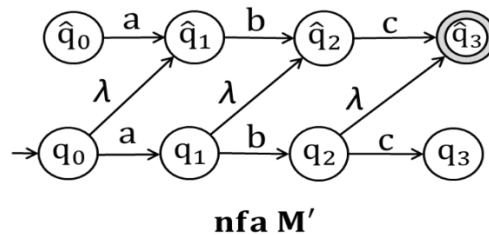
drop(L):

فرض کنید $\text{drop}(L)$ زبانی باشد که رشته هایش با حذف یک الفبای دلخواه از رشته های عضو L به دست آمده اند. برای مثال اگر $abc \in L$ آنگاه $ab, ac, bc \in \text{drop}(L)$. اگر L منظم باشد $\text{drop}(L)$ نیز منظم است. ماشین متناهی پذیرنده $\text{drop}(L)$ به صورت زیر به دست می آید:

$$L = \{abc\}$$



$$\text{drop}(L) = \{ab, ac, bc\}$$



حل تست:

۸۴- فرض کنیم زبان های $L_A \subseteq \Sigma^*$ و $L_B \subseteq \Sigma^*$ توسط DFA های $A = (Q_A, \Sigma, \delta_A, q_0, F_A)$ و $B = (Q_B, \Sigma, \delta_B, q'_0, F_B)$ داده شده اند و $\Sigma = \{0, 1, 2\}$. اتوماتای C را با زبان L_C به صورت زیر در نظر می گیریم:

$$C = (Q_A \times Q_B, \Sigma, \delta_C, (q_0, q'_0), F_A \times F_B)$$

$$\delta_C((p, q), a) = \{(p', q) : p' \in \delta_A(p, a)\} \cup \{(p, q') : q' \in \delta_B(q, a)\}$$

دو عبارت زیر، مفروض است.

I: اگر $010 \in L_A$ و $210 \in L_B$ آنگاه $021100 \in L_C$

II: اگر $010 \in L_A$ و $210 \in L_B$ آنگاه $210010 \in L_C$

کدام موارد فوق، همواره درست است؟

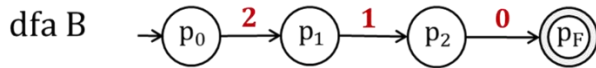
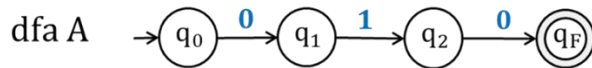
علوم کامپیوتر ۱۴۰۳

- (۲) فقط II
(۴) نه I و نه II

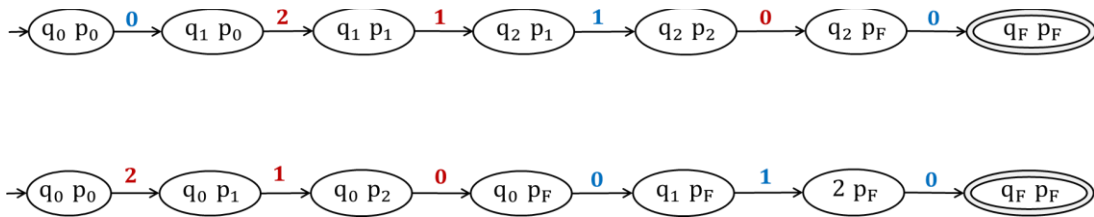
- (۱) فقط I
(۳) هم I و هم II

پاسخ: کلید اولیه گزینه یک بوده است. اما در این مثال هر دو رشته پذیرفته می‌شوند:
فرض کنیم:

$$L_A = \{010\}, L_B = \{210\}$$



ماشین C غیرقطعی است. ما در این شکل ها فقط مسیر (walk) منتهی به پذیرش رشته‌ها را نشان داده‌ایم:



همربختی (همومورفیسم):

فرض کنید Σ_1 و Σ_2 دو مجموعه الفبا و $h: \Sigma_1^* \rightarrow \Sigma_2^*$ یک تابع باشد. هرگاه برای هر رشته $w = a_1 a_2 \dots a_n$ تابع h بتواند روی الحاق توزیع شود یعنی داشته باشیم:

$$h(w) = h(a_1)h(a_2) \dots h(a_n)$$

آن را یک همومورفیسم می‌نامیم.

برای هر زبان L مجموعه $h(L)$ را تصویر L تحت h می‌نامیم. اگر L منظم باشد، $h(L)$ منظم است.

مثال ۱:

در این مثال با آن که L نامنظم است $h(L)$ منظم است.

$$h: \{a, b, c\} \rightarrow \{a, b, c\}^*$$

$$h(a) = b, h(b) = b, h(c) = \lambda$$

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

$$h(a^n b^n c^n) = h(a)^n h(b)^n h(c)^n = (b)^n (b)^n \lambda^n = b^{2n}$$

$$h(L) = \{b^{2n} \mid n \geq 0\}$$

مثال ۲:

در این مثال L منظم است و $h(L)$ نیز منظم است.
تصویر همریختی هر زبان منظم، حتما منظم است.

$$h: \{a, b\} \rightarrow \{a, b, c\}^*$$

$$h(a) = b^2, h(b) = ca$$

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

$$h(a^n b^{2m}) = h(a)^n h(b)^{2m} = (b^2)^n (ca)^{2m} = b^{2n} (ca)^{2m}$$

$$h(L) = \{b^{2n} (ca)^{2m} \mid n, m \geq 0\}$$

تصویر معکوس همریختی:

اگر L منظم باشد است و $h^{-1}(L)$ نیز منظم است.
 البته منظور از $h^{-1}(L)$ بزرگترین وارونی است که می‌توان در نظر گرفت. زیرا ممکن است تابع h یک به یک نباشد و h^{-1} تابع نباشد مثلاً $h^{-1}(a)$ چندمقداری باشد.

$h(L)$ منظم است $\longrightarrow L$ منظم باشد

$h^{-1}(L)$ منظم است $\longrightarrow L$ منظم باشد

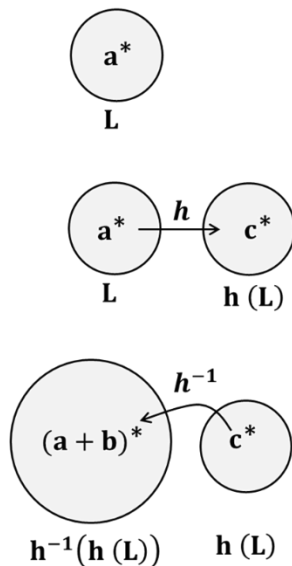
توجه:

$$L \subseteq h^{-1}(h(L))$$

در حالت کلی داریم:

$$h(h^{-1}(L)) \subseteq L$$

یک مثال ساده: فرض کنید $h(a) = c$, $h(b) = c$, $h(c) = c$ حالا مطابق شکل از $L = a^*$ شروع کنیم ابتدا تصویر و سپس معکوس تصویر آن را به دست آوریم:



$$L = a^* \Rightarrow h(L) = c^* \Rightarrow h^{-1}(h(L)) = (a + b)^*$$

$$L \subseteq h^{-1}(h(L)) \text{ می‌بینیم که:}$$

چگونه نامنظم بودن یک زبان ثابت می‌شود؟

تا اینجا روشهای اثبات و شرایط کافی برای منظم بودن یک زبان را مطالعه کردیم. حالا فرض کنید بخواهیم نامنظم بودن یک زبان را ثابت کنیم. شرط کافی برای نامنظم بودن یک زبان چیست؟

فصل چهارم: ویژگیهای بستاری زبان‌های منظم

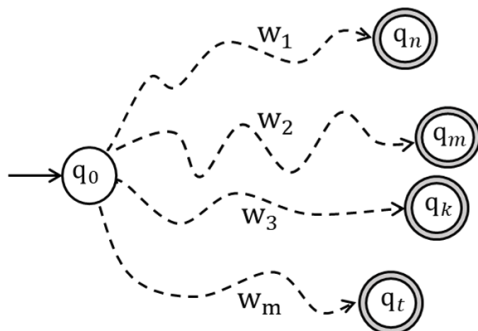
جلسه دهم بخش دو:

لم تزریق زبان‌های منظم - شرط کافی برای نامنظم بودن

چگونه نامنظم بودن یک زبان ثابت می‌شود؟

تا اینجا شرایط کافی برای اثبات منظم بودن یک زبان را مطالعه کردیم:

- زبان L متناهی باشد $\leftarrow$ زبان L منظم است.
- عبارت منظم r موجود باشد که $L = L(r)$ $\leftarrow$ زبان L منظم است.
- ماشین متناهی M موجود باشد که $L = L(M)$ $\leftarrow$ زبان L منظم است.
- گرامر منظم G موجود باشد که $L = L(G)$ $\leftarrow$ زبان L منظم است.



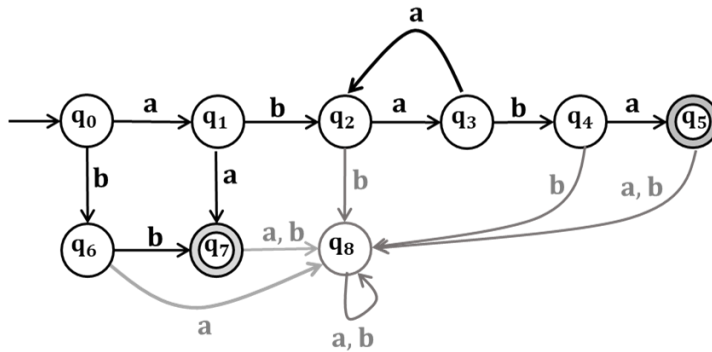
مثلا زبان متناهی $L = \{w_1, w_2, w_3, \dots, w_m\}$ دستکم یک ماشین متناهی پذیرنده مطابق شکل دارد.

حالا فرض کنید بخواهیم نامنظم بودن یک زبان را ثابت کنیم. **شرط کافی برای نامنظم بودن یک زبان چیست؟** برای یافتن این شرط کافی، **لم تزریق زبان های منظم** را مطالعه می‌کنیم.

اگر زبان نامتناهی L منظم باشد، آنگاه یک dfa پذیرنده دارد که شامل یک دور یا حلقه در مسیر از حالت اولیه تا حالت پذیرش است. رشته های عضو L از یک اندازه به بعد، دارای دور هستند و آن دور قابل تکرار یا حذف است (قسمت قابل تزریق).

مثال: به dfa پذیرنده زبان L توجه کنید.

$$L = \{a^2, b^2, ababa, aba^3ba, aba^5ba, aba^7ba, \dots\}$$



در این مثال رشته های $a^2, b^2, ababa$ بدون دور به پذیرش رسیده اند اما رشته های بزرگتر دارای تکرارهایی از حلقه a^2 هستند.

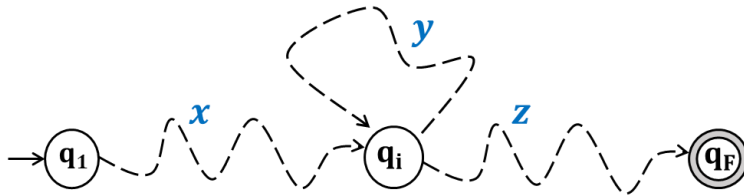
لم تزریق زبان منظم:

اگر L زبان منظم (منظم نامتناهی) باشد. عدد طبیعی n وجود دارد که هر $w \in L$ با طول $|w| \geq n$ دارای تجزیه به فرم $w = xyz$ است طوری که:

الف: $|y| \geq 1$

ب: $|xy| \leq n$

ج: $\forall k \geq 0 \quad xy^kz \in L$



توجه: عدد n را ثابت تزریق می‌نامند. تعداد حالات ماشین dfa یک مقدار قابل قبول برای n است.

توجه: زبان های متناهی هم در لم تزریق صدق می‌کنند به این صورت که از یک n به بعد هیچ رشته ای با طول $|w| \geq n$ در L وجود ندارد.

لم تزریق زبان منظم به زبان منطقی:

اگر زبان L منظم باشد آنگاه:

$\exists n \quad \forall w \in L \quad |w| \geq n \text{ then } \exists x, y, z \text{ s.t.}$

$$w = xyz$$

$$1. |y| \geq 1$$

$$2. |xy| \leq n$$

$$3. xy^kz \in L \quad k = 0, 1, 2, \dots$$

شرط کافی برای نامنظم بودن یک زبان:

اگر عکس و نقیض گزاره بالا را بنویسیم، یک شرط کافی برای نامنظم بودن زبان به دست می‌آید:

$$\forall n \exists w \in L \text{ s.t. } |w| \geq n \text{ and}$$

$$\forall x, y, z \quad w = xyz \text{ then}$$

$$\text{then } \sim(|y| \geq 1 \text{ and } |xy| \leq n \text{ and } xy^kz \in L \quad k \geq 0)$$

نتیجه مهم:

اگر می‌خواهیم نامنظم بودن زبان L ثابت شود باید نشان دهیم برای هر مقدار دلخواه n رشته‌ای با طول حداقل n در L وجود دارد که هر طور آن را به سه بخش xyz تفکیک کنیم، حداقل یک شرط از شرایط $1, 2, 3$ برقرار نشود. مثلاً اگر $|y| \geq 1$ و $|xy| \leq n$ برقرار بود آنگاه شرط 3 برقرار نمی‌شود. یعنی xy^kz برای برخی مقادیر k عضو نیست.

تمرین: ثابت کنید زبان $L = \{0^n 1^n \mid n \geq 0\}$ منظم نیست.

خلاصه‌ی برهان:

برای هر عدد طبیعی N رشته به فرم $w = 0^N 1^N$ را در نظر بگیریم. اندازه‌اش بیشتر از N است. اگر $w = 0^N 1^N = xyz$ و $|y| \geq 1$ و $|xy| \leq N$ آنگاه رشته y رشته ناتهی از ارقام 0 است پس تکرار یا حذف آن باعث نابرابری تعداد 0 و 1 می‌شود.

$$w = 0^N 1^N = \underbrace{0000000}_{xy} 011111111$$

تمرین: ثابت کنید زبان $L = \{w \mid n_a(w) = n_b(w)\}$ منظم نیست.

خلاصه ای از برهان:

چون رشته های به فرم $a^N b^N$ نیز عضو این زبان هستند می توان مانند تمرین قبل از این رشته ها استفاده کرد. برای هر عدد طبیعی N رشته به فرم $w = a^N b^N$ را در نظر بگیریم. اندازه w بیشتر از N است. اگر $w = a^N b^N = xyz$ و $|y| \geq 1$ و $|xy| \leq N$ آنگاه y رشته ناتهی از الفبای a است پس تکرار یا حذف آن باعث نابرابری تعداد a و b می شود.

$$w = a^N b^N = \underbrace{aaaaaa}_{xy} abbbbbbb$$

نمونه های دیگر از زبان های نامنظم

نامنظم بودن این زبان ها نیز با لم تزریق قابل اثبات است.

مثال: این زبان منظم نیست:

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

برای توابع غیر خطی $n^3, n!, f(n) = n^2$ زبان نامتناهی $\{a^{f(n)} \mid n \geq 0\}$ منظم نیست. اما برای تابع خطی مانند $f(n) = an + b$ زبان $\{a^{an+b} \mid n \geq 0\}$ منظم است.

مثال: این زبان‌ها منظم نیستند:

$$L = \{w \mid n_a(w) \leq n_b(w)\}$$

$$L = \{a^i b^j \mid i \leq 2j\}$$

مثال: مجموعه رشته‌های آینه‌ای منظم نیست.

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

مجموعه رشته‌های آینه‌ای به طول زوج منظم نیست:

$$L = \{ww^R \mid w \in (a + b)^*\}$$

مسئله‌های تصمیم‌پذیر در مورد زبان‌های منظم:

مسئله	الگوریتم تصمیم
؟ $w \in L$	زبان منظم ماشین dfa دارد که رشته w به طول n را در زمان $O(n)$ مصرف و چه عضو زبان باشد یا نه، متوقف شده و عضویت یا عدم عضویت را تعیین می‌کند.
؟ $L = \emptyset$	زبان منظم ماشین dfa دارد. الگوریتمی وجود دارد که می‌تواند وجود یا عدم وجود مسیر از حالت اولیه به حالت پذیرش را بررسی کرده و متوقف شود. اگر مسیری وجود داشته باشد زبان L ناتهی است و بالعکس.
L متناهی است ؟	زبان منظم ماشین dfa دارد. الگوریتمی وجود دارد که می‌تواند وجود یا عدم وجود دور در مسیری از حالت اولیه به حالت پذیرش را بررسی کرده و متوقف شود. اگر چنین دوری وجود داشته باشد زبان L نامتناهی است و در غیراینصورت متناهی است.
؟ $L_1 \subseteq L_2$	L_1 و L_2 منظم باشند آنگاه $L_1 - L_2$ منظم است. حالا کفایت به این سوال پاسخ دهیم که آیا $L_1 - L_2 = \emptyset$ است؟ این مسئله تصمیم‌پذیر است.
؟ $L_1 = L_2$	برای تساوی زبان‌ها باید $L_1 \subseteq L_2$ و $L_2 \subseteq L_1$ باشد. این مسئله تصمیم‌پذیر است.

-11-

فصل پنجم: زبان‌های مستقل از متن

جلسه یازدهم بخش یک: گرامرهای مستقل از متن - درخت اشتقاق

یادآوری: گرامر و نحوه تولید رشته توسط آن.

A grammar consists of a collection of *substitution rules*, also called *productions*. Each rule appears as a line in the grammar, comprising a symbol and a string separated by an arrow. The symbol is called a *variable*. The string consists of variables and other symbols called *terminals*. The variable symbols

$$A \rightarrow 0A1$$

$$A \rightarrow B$$

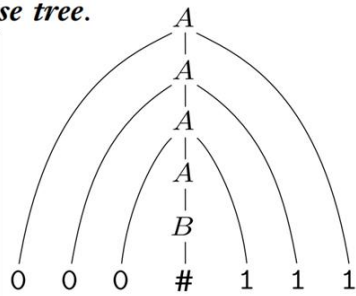
$$B \rightarrow \#$$

تولید رشته‌ی 000#111 :

$$A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000B111 \Rightarrow 000\#111.$$

درخت اشتقاق مربوط به تولید این رشته:

parse tree.



مثال: یک گرامر برای تولید جملاتی در زبان انگلیسی:

```

<SENTENCE> → <NOUN-PHRASE><VERB-PHRASE>
<NOUN-PHRASE> → <CMPLX-NOUN> | <CMPLX-NOUN><PREP-PHRASE>
<VERB-PHRASE> → <CMPLX-VERB> | <CMPLX-VERB><PREP-PHRASE>
<PREP-PHRASE> → <PREP><CMPLX-NOUN>
<CMPLX-NOUN> → <ARTICLE><NOUN>
<CMPLX-VERB> → <VERB> | <VERB><NOUN-PHRASE>
<ARTICLE> → a | the
<NOUN> → boy | girl | flower
<VERB> → touches | likes | sees
<PREP> → with

```

نمونه جملات تولید شده توسط گرامر:

```

a boy sees
the boy sees a flower
a girl with a flower likes the boy

```

```

<SENTENCE> ⇒ <NOUN-PHRASE><VERB-PHRASE>
              ⇒ <CMPLX-NOUN><VERB-PHRASE>
              ⇒ <ARTICLE><NOUN><VERB-PHRASE>
              ⇒ a <NOUN><VERB-PHRASE>
              ⇒ a boy <VERB-PHRASE>
              ⇒ a boy <CMPLX-VERB>
              ⇒ a boy <VERB>
              ⇒ a boy sees

```


گرامر های مستقل از متن:

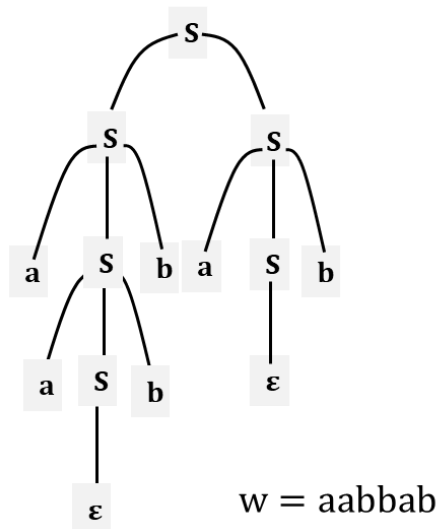
همه قواعد تولید در این نوع گرامر به فرم $A \rightarrow \alpha$ هستند. که $A \in V$ و $\alpha \in (V \cup \Sigma)^*$.

مثال: گرامر G_p

اگر a و b را همان پرانتزهای باز و بسته فرض کنیم این گرامر تمام رشته های قابل تولید خوش ساخت با پرانتز را تولید می کند. درخت اشتقاق رشته $aabbab$ را نشان داده ایم. رشته $aabbab$ معادل با $((()))$ است.

G_p :

$S \rightarrow aSb \mid SS \mid \epsilon$.



اشتقاق چپ و اشتقاق راست:

در اشتقاق چپ، هر بار آخرین متغیر از سمت چپ را اشتقاق می‌دهیم و در اشتقاق راست هر بار آخرین متغیر از سمت راست را اشتقاق می‌دهیم.

مثال: تولید $w = ab^2$ توسط گرامر G را یکبار با اشتقاق چپ و بار دیگر با اشتقاق راست مشاهده می‌کنید:

G :

$S \rightarrow aSb \mid AB \mid \epsilon$

$A \rightarrow AS \mid a$

$B \rightarrow bB \mid b$

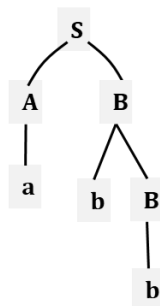
$S \rightarrow AB \rightarrow AbB \rightarrow Abb \rightarrow abb$

اشتقاق چپ:

$S \rightarrow AB \rightarrow aB \rightarrow abB \rightarrow abb$

اشتقاق راست:

درخت اشتقاق:



ابهام در تولید رشته - گرامر مبهم.

اگر برای یک رشته دو درخت اشتقاق متفاوت وجود داشته باشد (w با بیش از یک طریق قابل تولید باشد) می‌گوییم آن رشته دارای تولید مبهم است و آن گرامر را **مبهم** می‌نامیم. به راحتی می‌توان هر گرامر را، به گرامر مبهم معادل تبدیل کرد.

برای نمونه: در مثال قبل گرامر G رشته $w = ab^2$ را به این صورت تولید کرد:

$$S \rightarrow AB \rightarrow AbB \rightarrow Abb \rightarrow abb$$

حالا با افزودن قواعد جدید $S \rightarrow X, X \rightarrow ab^2$ یک مسیر دیگر برای تولید $w = ab^2$ به وجود می‌آوریم و باعث مبهم شدن گرامر می‌شویم.

$G :$

$$S \rightarrow aSb \mid AB \mid \epsilon \mid X$$

$$A \rightarrow AS \mid a$$

$$B \rightarrow bB \mid b$$

$$X \rightarrow ab^2$$

توجه: هر گرامر مستقل از متن، دارای گرامر معادل مبهم هست. پس هر زبان مستقل از متن حداقل یک گرامر مستقل از متن **مبهم** دارد. اما سوال مهم این است که آیا گرامر مستقل از متن غیرمبهم نیز دارد یا خیر؟ این را جلسه بعد مطرح می‌کنیم.

مثال: گرامر غیرمبهم و گرامر مبهم برای عبارات ریاضی

گرامر های زیر تمام عبارات ریاضی خوش ساخت با الفبای $\Sigma = \{a, +, *, (,)\}$ را تولید می‌کنند.

$G_a:$

$$E \rightarrow E + T \mid T$$

$$T \rightarrow T * F \mid F$$

$$F \rightarrow (E) \mid a$$

$G'_a:$

$$E \rightarrow E + E \mid E * E \mid (E) \mid a$$

در G_a قواعد تولید ابتدا عمل جمع، سپس ضرب، و در انتها پرانتز را تولید می‌کنند، (برخلاف ترتیب اولویت این عملگرها) این یک گرامر غیر مبهم است. اما اگر همه این عملگرها در یک سطح تولید شوند گرامری مبهم خواهیم داشت. G_a و G'_a معادل هستند. G'_a مبهم است و G_a غیرمبهم.

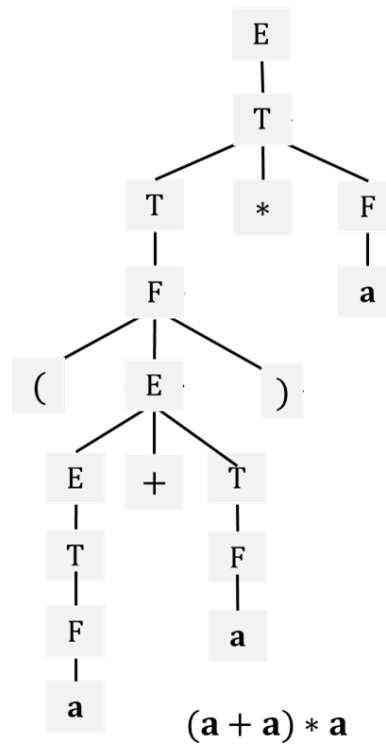
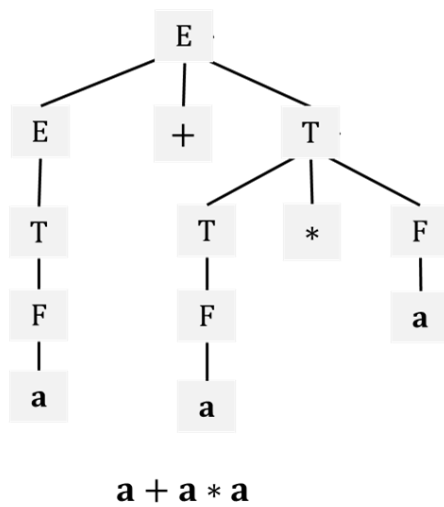
مثال: عبارات ریاضی $a + a * a$ و $(a + a) * a$ هر کدام فقط یک درخت اشتقاق در G_a دارند.

G_a :

$E \rightarrow E + T \mid T$

$T \rightarrow T * F \mid F$

$F \rightarrow (E) \mid a$



چند نمونه گرامر مستقل از متن و زبان های تولید شده توسط آنها:

نمونه ۱: چند گرامر برای تولید رشته‌هایی که $n_a(w) = n_b(w)$ در جلسات پیشین مشاهده کرده‌ایم.

$G1 :$

$S \rightarrow aSbS \mid bSaS \mid \lambda$

$G2 :$

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

$$L(G1) = L(G2) = \{w \in \{a, b\}^* \mid n_a(w) = n_b(w)\}$$

نمونه ۲: برای تولید رشته هایی که در آنها $n_a(w) = 2n_b(w)$ باشد.

G₂ :

$S \rightarrow aSaSbS \mid aSbSaS \mid bSaSaS \mid \lambda$

$$L(G_2) = \{w \in \{a, b\}^* \mid n_a(w) = 2n_b(w)\}$$

به طور مشابه:

G₃ :

$S \rightarrow aSbSbSbS \mid bSaSbSbS \mid bSbSaSbS \mid bSbSbSaS \mid \lambda$

$$L(G_3) = \{w \in \{a, b\}^* \mid n_b(w) = 3n_a(w)\}$$

فصل پنجم: زبان‌های مستقل از متن

جلسه یازدهم بخش دو: گرامرهای مستقل از متن – درخت اشتقاق – parse

نمونه ۳:

$G_{\neq}$:

$$S \rightarrow S_1 a S_1 \mid S_2 b S_2$$

$$S_1 \rightarrow a S_1 b \mid b S_1 a \mid S_1 S_1 \mid a S_1 \mid \lambda$$

$$S_2 \rightarrow a S_2 b \mid b S_2 a \mid S_2 S_2 \mid b S_2 \mid \lambda$$

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

اگر $n_a(w) \neq n_b(w)$ دو نوع داریم: یا $n_a(w) > n_b(w)$ است یا $n_a(w) < n_b(w)$ است. قاعده $S \rightarrow S_1 a S_1$ رشته‌های نوع اول و قاعده $S \rightarrow S_2 b S_2$ رشته‌های نوع دوم را تولید می‌کند.

G_R :

نمونه ۴: رشته‌های آینه‌ای

$$S \rightarrow a S a \mid b S b \mid a \mid b \mid \lambda$$

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

نمونه ۵: متغیرهای S_1 و S_2 هر کدام نوعی از رشته‌ها را تولید می‌کنند.

G :

$$S \rightarrow S_1 \mid S_2$$

$$S_1 \rightarrow a S_1 b \mid A$$

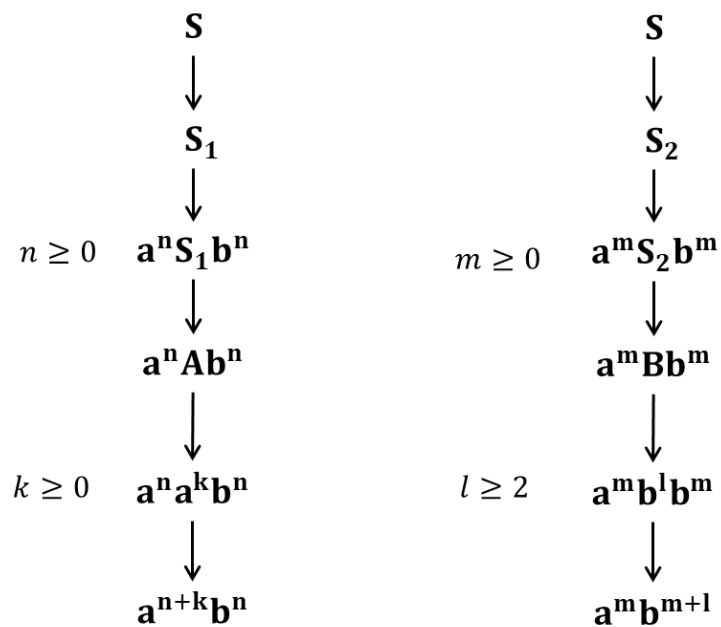
$$S_2 \rightarrow a S_2 b \mid B$$

$$A \rightarrow a A \mid \lambda$$

$$B \rightarrow b B \mid b b$$

$$L(G) = \{a^i b^j \mid j \neq i + 1\} = \{a^i b^j \mid j \leq i \text{ or } j \geq i + 2\}$$

در شکل زیر مراحل تولید این دو نوع رشته را نشان داده‌ایم.



چند الگوی پر کاربرد از قواعد تولید:

قاعده	نتیجه
$S \rightarrow a S b$	$a^n S b^n$
$S \rightarrow a S b b$	$a^n S b^{2n}$
$S \rightarrow a S b b \mid a S b \mid a S$	$a^n S b^m \quad m \leq 2n$
$S \rightarrow a S b b \mid a S b$	$a^n S b^m \quad n \leq m \leq 2n$
$S \rightarrow a S a \mid b S b$	$w S w^R$

توجه: به تفاوت این گرامر ها و زبان تولید شده هر کدام دقت کنید:

G₁:

$$S \rightarrow aSbb \mid aSb \mid \lambda \quad L_1 = \{a^n b^m \mid n \leq m \leq 2n\}$$

G₂:

$$\begin{aligned} S &\rightarrow S_1 \mid S_2 & L_2 &= \{a^n b^m \mid m = n \text{ or } m = 2n\} \\ S_1 &\rightarrow aS_1b \mid \lambda \\ S_2 &\rightarrow aS_2bb \mid \lambda \end{aligned}$$

کدام مورد، زبان تولیدشده توسط گرامر زیر را توصیف می کند؟ (مهندسی کامپیوتر ۱۴۰۴)

$$S \rightarrow aSc \mid aS \mid A$$

$$A \rightarrow bAcc \mid bA \mid bAc \mid \epsilon$$

$$\{a^m b^n c^p \mid m + 2n \geq p\} \quad (۲)$$

$$\{a^m b^n c^p \mid 2m + n \geq p\} \quad (۱)$$

$$\{a^m b^n c^p \mid 2m + 2n \geq p\} \quad (۴)$$

$$\{a^m b^n c^p \mid m + n > p\} \quad (۳)$$

پاسخ: گزینه ۲

با قواعد $S \rightarrow aSc \mid aS$ به عبارت $a^n S c^m$ ($m \leq n$) می رسیم، سپس با $S \rightarrow A$ به $a^n A c^m$

می رسیم. حالا با قواعد $A \rightarrow bAcc \mid bA \mid bAc \mid \epsilon$ به ازای هر b حداکثر دو c افزوده می شود پس

رشته های $a^n b^k c^l c^m$ به دست می آیند که $m \leq n$ و $l \leq 2k$. در نتیجه:

$$L(G) = \{a^n b^k c^{l+m} \mid l + m \leq 2k + n\}$$

که همان گزینه ۲ است.

تجزیه (parse) و تشخیص عضویت w در زبان مستقل از متن:

مقدمه: دیدیم که مساله تعیین عضویت یا عدم عضویت در $L(G)$ برای گرامر منظم در زمان خطی بر حسب اندازه w قابل انجام است. حالا همین مساله را برای گرامر مستقل از متن بررسی کنیم.

الگوریتم جستجوی کامل:

گرامر مستقل از متن G با متغیر آغاز S و مجموعه قواعد تولید P را داریم. رشته w داده شده است و می‌خواهیم تولید یا عدم تولید آن توسط گرامر G را تشخیص دهیم. جستجوی کامل در واقع جستجوی سطح نخست (BFS) در درخت با ریشه‌ی S و فاکتور انشعاب حداکثر $|P|$ است و به صورت سطح به سطح تمام عبارات قابل تولید در عمق ۱، عمق ۲، ... را مشاهده می‌کند تا به رشته w برسد یا مشخص شود که این رشته تولید نخواهد شد.

مرتبه زمانی: برای گرامر مستقل از متن دلخواه G دو حالت خواهیم داشت:

الف: اگر $w \in L(G)$ باشد جستجو در زمان متناهی متوقف می‌شود، و علاوه بر تشخیص عضویت w ، مسیر تولید آن نیز معلوم می‌شود.

ب: اگر $w \notin L(G)$ ممکن است جستجو متوقف نشود و زمان اجرا نامتناهی شود. (البته این مشکل با انتخاب گرامر مناسب برطرف شدنی است)

مثال: جستجوی کامل برای تولید رشته $w = bb$ توسط گرامر G .

G :

$S \rightarrow aSb \mid AB$

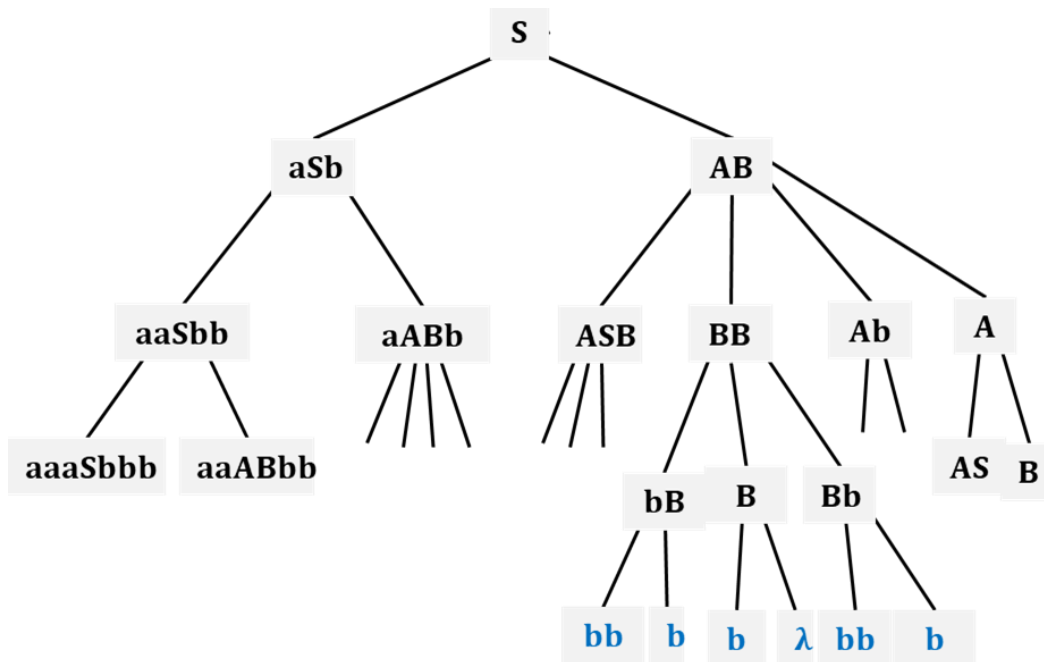
$A \rightarrow AS \mid B$

$B \rightarrow b \mid \lambda$

مطابق شکل پس از چند مرحله انشعاب، در عمق ۴ رشته bb تولید می‌شود و جستجو به پایان می‌رسد و مراحل تولید آن نیز مشخص می‌شود:

$S \rightarrow AB \rightarrow BB \rightarrow bB \rightarrow bb$

اما برای رشته‌هایی که قابل تولید نباشند، جستجوی کامل تا عمق بینهایت ادامه می‌یابد.



توقف جستجوی کامل برای گرامر غیرانقباضی:

گرامر G که فاقد قاعده تولید λ و فاقد قواعد یکانی (مانند $S \rightarrow A$ و $A \rightarrow B$) باشد، گرامر غیرانقباضی نامیده می‌شود. در گرامر غیرانقباضی بعد از هر اشتقاق، طول رشته کاهش نخواهد یافت همچنین طول رشته نمی‌تواند بیش از چند مرحله متناهی، ثابت بماند و در نهایت افزایش خواهد یافت. در این نوع گرامر وقتی جستجوی کامل را برای رشته w به طول n اجرا می‌کنیم، کفایت جستجو را حداکثر تا عمق $2n$ ادامه دهیم زیرا اگر تا این عمق پیش رفتیم و w تولید نشد، دیگر تولید نخواهد شد. (در واقع $2n - 1$ حداکثر عمق ممکن برای تولید این رشته است).

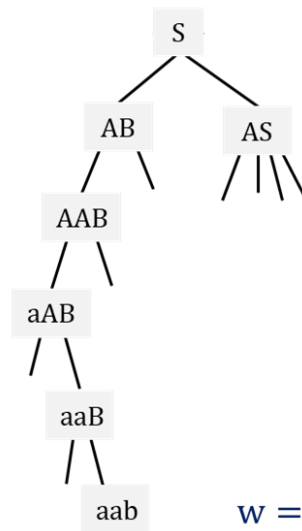
مثال: در گرامر غیر انقباضی G تمام رشته های به طول ۳ که قابل تولید باشند، حداکثر تا عمق ۶ تولید می‌شود. مانند $w = aab$:

G :

$S \rightarrow AB \mid AS$

$A \rightarrow AA \mid aB \mid a$

$B \rightarrow b$



$w = aab \in L(G)$

مرتبۀ زمانی جستجوی کامل برای گرامر غیرانقباضی:

فرض کنیم گرامر مستقل از متن G غیرانقباضی باشد و $|w| = n$. تعداد کل گره‌های درخت جستجو تا

عمق $2n$ برابر است با:

$$1 + |P| + \dots + |P|^{2n} = \frac{|P|^{2n+1} - 1}{|P| - 1}$$

نتیجه: جستجوی کامل برای گرامر مستقل از متن و غیرانقباضی G و برای هر رشته w چه $w \in L(G)$ و

چه $w \notin L(G)$ متوقف خواهد شد و زمان اجرای آن نمایی از مرتبۀ $O(|P|^{2n+1})$ است.

توجه: در ادامه درس به نسخه‌های بهتری از الگوریتم تجزیه و تشخیص عضویت برای زبان‌های مستقل

از متن اشاره خواهیم کرد که در زمان چند جمله‌ای به پایان می‌رسند. برای مثال در حالت خاص اگر

گرامر مورد نظر گرامر ساده باشد الگوریتم از مرتبۀ $O(n)$ است.

-12-

فصل پنجم: گرامر های مستقل از متن

جلسه دوازدهم بخش یک: گرامر مبهم و غیر مبهم

زبان های غیرمبهم و ذاتاً مبهم:

گفتیم که ایجاد ابهام در یک گرامر بسیار ساده است. هر زبان مستقل از متن گرامر مستقل از متن مبهم نیز دارد.

مثال: زبان L_1 دارای گرامر غیرمبهم G_1 است. رشته های به فرم $a^n b^n$ را از طریق S_1 و رشته های به فرم $a^n b^{2n}$ از طریق S_2 تولید می کند و چون $n > 0$ است این دو فرم اشتراک ندارند. یعنی رشته ای وجود ندارد که از دو طریق تولید شود. مثلاً ab فقط از طریق S_1 تولید می شود: $S \rightarrow S_1 \rightarrow ab$. اگر بخواهیم، می توانیم با افزودن متغیر S_3 یک مسیر دیگر هم برای تولید ab ایجاد کنیم تا گرامر G_1 به گرامر مبهم G'_1 تبدیل شود. پس هر گرامر غیرمبهم دارای معادل مبهم است.

$$L_1 = \{a^n b^m \mid m = n \text{ or } m = 2n, n > 0\}$$

G_1 :

$$\begin{aligned} S &\rightarrow S_1 \mid S_2 \\ S_1 &\rightarrow aS_1b \mid ab \\ S_2 &\rightarrow aS_1b^2 \mid ab^2 \end{aligned}$$

G'_1 :

$$\begin{aligned} S &\rightarrow S_1 \mid S_2 \mid S_3 \\ S_1 &\rightarrow aS_1b \mid ab \\ S_2 &\rightarrow aS_1b^2 \mid ab^2 \\ S_3 &\rightarrow ab \end{aligned}$$

اما برخی زبان های مستقل از متن فقط گرامر مبهم دارند. به آنها زبان های ذاتاً مبهم می گوئیم.

مثال:

زبان L_2 ذاتا مبهم است. هر گرامر مولد آن دارای ابهام است. مثلا اگر رشته های $a^n b^m c^k$ در حالت $m = n$ را از طریق S_1 و رشته های حالت $m = k$ را از طریق S_2 تولید کنیم باز هم تعداد بی نهایت رشته وجود دارد که در آنها هم $m = n$ و هم $m = k$ است یعنی رشته های $a^n b^n c^n$ دارای ابهام در تولید هستند. (از هر دو طریق تولید می شوند)

دقت کنید اگر تعداد این رشته های مشترک متناهی بود باز هم می توانستیم مشکل ابهام را برطرف کنیم اما برای بی نهایت رشته مشترک، نمی توان رفع ابهام کرد.

 G_2 : $S \rightarrow S_1 \mid S_2$ $S_1 \rightarrow S_1 c \mid A$ $S_2 \rightarrow a S_2 \mid B$ $A \rightarrow a A b \mid \lambda$ $B \rightarrow b B c \mid \lambda$

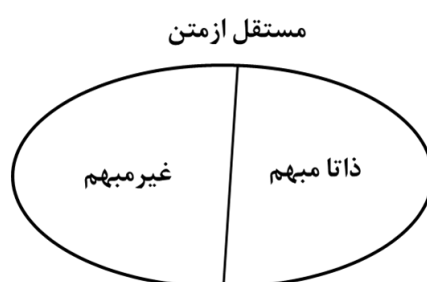
$$L_2 = \{a^n b^m c^k \mid m = n \text{ or } m = k\}$$

$$S \rightarrow S_1 \xrightarrow{*} S_1 c^k \rightarrow A c^k \xrightarrow{*} a^m A b^m c^k \rightarrow a^m b^m c^k$$

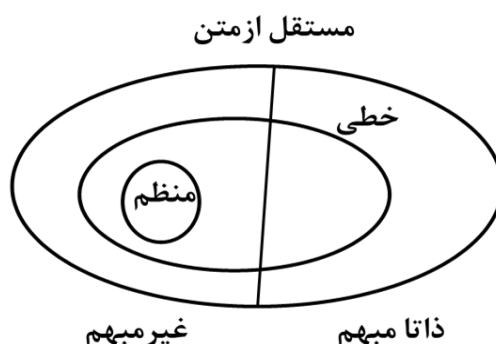
$$S \rightarrow S_2 \xrightarrow{*} a^n S_2 \rightarrow a^n B \xrightarrow{*} a^n b^m B c^m \rightarrow a^n b^m c^m$$

توجه:

زبان های مستقل از متن به دو دسته افراز می شوند. زبان های ذاتا مبهم که هر گرامر مستقل از متن آنها، مبهم است و زبان های غیرمبهم که دارای گرامر غیرمبهم هستند (البته گرامر مبهم نیز دارند).

**توجه:**

هر زبان منظم دارای گرامر منظم است پس دارای گرامر خطی است. همچنین هر گرامر خطی، یک گرامر مستقل از متن است. از طرفی همان طور که در شکل نشان داده ایم، زبان های منظم حتما دارای گرامر غیرمبهم هستند. زبان های خطی، برخی ذاتا مبهم و برخی غیرمبهم هستند. اما زبان های منظم همگی غیر مبهم هستند.



گرامر تولید کننده عبارات جبری:

گرامرهای G_1 و G_2 با هم معادلند و همه عبارات جبری خوش ساخت با مجموعه الفبای

$$\Sigma = \{1, 2, \dots, 9, +, *, (,)\}$$

را تولید می کنند. در این گرامرها اگر همه اعمال را در یک سطح بدون توجه به اولویت ها تولید

کنیم گرامر مبهم به دست می آید. گرامر G_1 مبهم است.

اما اگر جمع، ضرب و پرانتز با توجه به عکس ترتیب اولویت آنها تولید شوند (ترتیب تولید، عکس

ترتیب اولویت باشد) گرامر غیر مبهم خواهیم داشت. گرامر G_2 غیر مبهم است.

در G_2 ابتدا عمل جمع، سپس عمل ضرب و در انتها پرانتز در سه سطح متفاوت تولید می شوند.

G_1 : مبهم

$$E \rightarrow E + E \mid E * E \mid (E) \mid 1 \mid 2 \mid \dots \mid 9$$

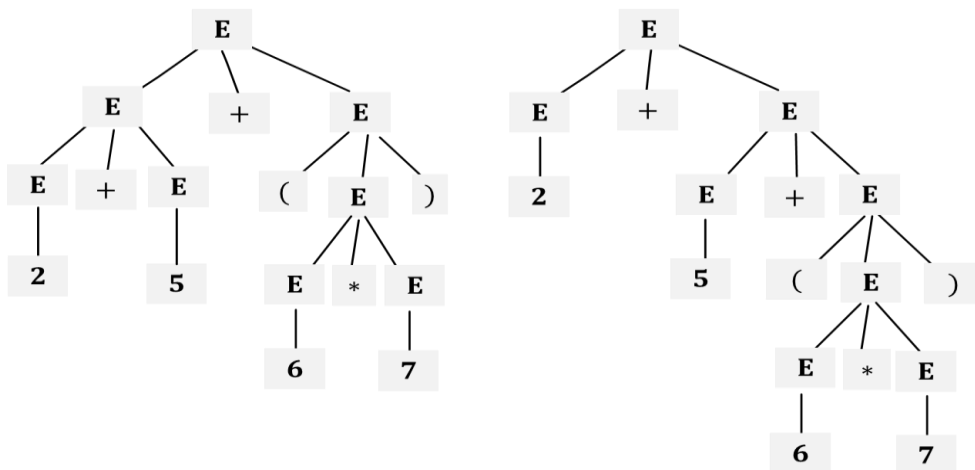
G_2 : غیر مبهم

$$E \rightarrow E + T \mid T$$

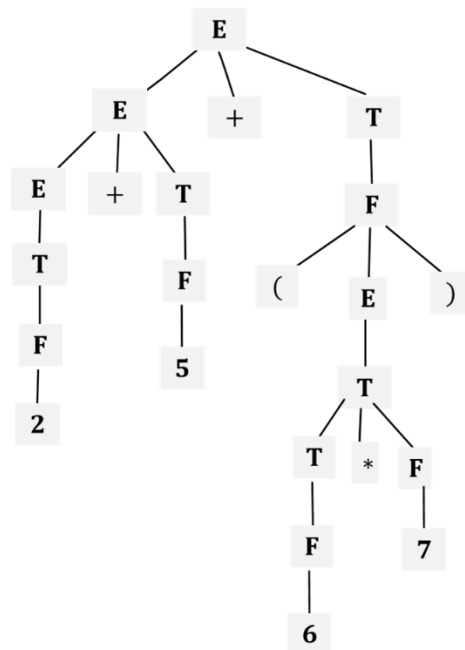
$$T \rightarrow T * F \mid F$$

$$F \rightarrow (E) \mid 1 \mid 2 \mid \dots \mid 9$$

دو روش مختلف برای تولید عبارت $2 + 5 + (6 * 7)$ توسط G_1 :

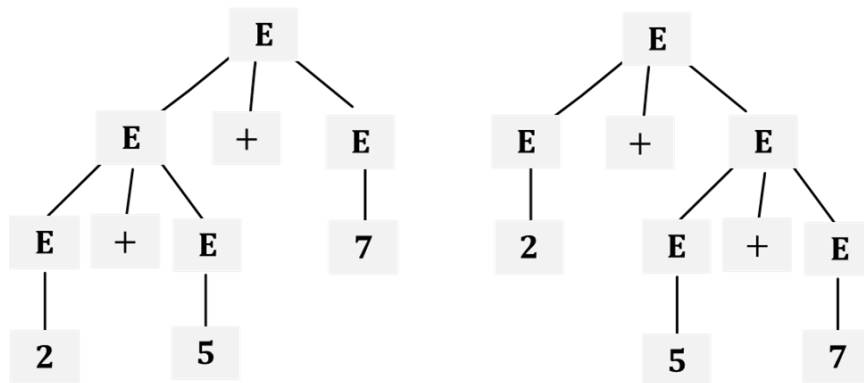


تنها راه تولید عبارت $2 + 5 + (6 * 7)$ توسط G_2 .



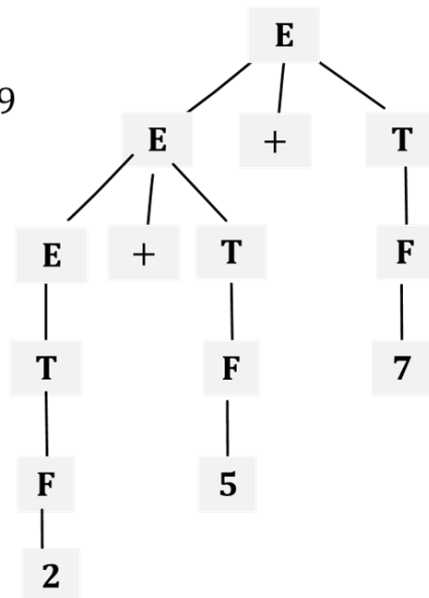
توجه: برای مجموع $2 + 5 + 7$ وقتی پیرانتزبندی نشده باشد، در G_1 دو درخت مختلف وجود دارد زیرا نوشتن این عبارت ممکن است از عمل جمع سمت چپ یا سمت راست آغاز شود. (شرکت پذیری چپ یا راست). اما در گرامر G_2 قاعده $E \rightarrow E + T$ بیان کننده شرکت پذیری چپ است یعنی فقط $2 + 5$ با 7 جمع می شود.

G₁:

$$E \rightarrow E + E \mid E * E \mid (E) \mid 1 \mid 2 \mid \dots \mid 9$$
**G₂:**

$$E \rightarrow E + T \mid T$$

$$T \rightarrow T * F \mid F$$

$$F \rightarrow (E) \mid 1 \mid 2 \mid \dots \mid 9$$


$G_2:$

$$\left\{ \begin{array}{l} \mathbf{E} \rightarrow \mathbf{E} + \mathbf{T} \mid \mathbf{T} \\ \mathbf{T} \rightarrow \mathbf{T} * \mathbf{F} \mid \mathbf{F} \end{array} \right. \text{شرکت پذیری چپ (در قواعد تولید بازگشتی)}$$

$$\mathbf{F} \rightarrow (\mathbf{E}) \mid 1|2|\dots|9$$

$G'_2:$

$$\left\{ \begin{array}{l} \mathbf{E} \rightarrow \mathbf{T} + \mathbf{E} \mid \mathbf{T} \\ \mathbf{T} \rightarrow \mathbf{F} * \mathbf{T} \mid \mathbf{F} \end{array} \right. \text{شرکت پذیری راست (در قواعد تولید بازگشتی)}$$

$$\mathbf{F} \rightarrow (\mathbf{E}) \mid 1|2|\dots|9$$

توجه: زبان های غیر مبهم نسبت به اجتماع بسته نیستند.

در مثال زیر L_1 و L_2 غیرمبهم هستند اما اجتماع آنها (L_3) ذاتا مبهم است.

زبان های غیر مبهم نسبت به اشتراک بسته نیستند.

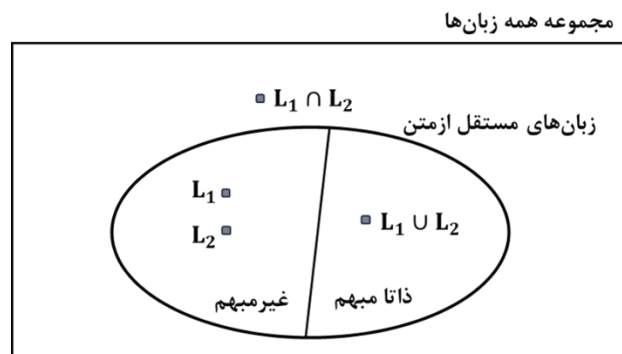
در مثال زیر L_1 و L_2 غیرمبهم هستند اما اشتراک آنها (L_4) اصلا مستقل از متن نیست.

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

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

$$L_3 = L_1 \cup L_2 = \{a^n b^m c^k \mid m = n \text{ or } m = k\}$$

$$L_4 = L_1 \cap L_2 = \{a^n b^m c^k \mid m = n \text{ and } m = k\} = \{a^n b^n c^n \mid n \geq 0\}$$



شکل مربوط به مثال L_1 و L_2

توجه: زبان‌های ذاتا مبهم و زبان‌های غیرمبهم، نسبت به وارون بسته‌اند.

L غیرمبهم است. $\Leftrightarrow L^R$ غیرمبهم است.

L ذاتا مبهم است. $\Leftrightarrow L^R$ ذاتا مبهم است.

گرامر ساده:

گرامر مستقل از متن G را ساده گوییم اگر تمام قواعد تولید آن به فرم $A \rightarrow aX$ باشد که a الفبا و $X \in V^*$ رشته ای دلخواه از متغیرهاست. همچنین هر متغیر A با هر الفبای a حداکثر یک قاعده تولید به فرم $A \rightarrow aX$ داشته باشد.

مثال:

گرامر G_1 ساده نیست چون قاعده $S \rightarrow AB$ دارد. گرامر G_2 ساده نیست چون متغیر A با الفبای b دو قاعده $A \rightarrow bAA \mid bB$ دارد. گرامر G_3 ساده است.

G_1 :

$$S \rightarrow AB \mid aB$$

$$A \rightarrow bA \mid a$$

$$B \rightarrow ba$$

G_2 :

$$S \rightarrow bAB \mid aB$$

$$A \rightarrow bAA \mid bB$$

$$B \rightarrow bB \mid a$$

G_3 :

$$S \rightarrow bAB$$

$$A \rightarrow bSA \mid aB$$

$$B \rightarrow aB \mid b$$

چند ویژگی گرامر ساده:

۱- چون برای هر متغیر $A \in V$ و هر الفبای $a \in \Sigma$ حداکثر یک قاعده تولید $A \rightarrow aX$ دارد پس تعداد قواعد تولید حداکثر به تعداد جفت های (A, a) است: $|P| \leq |V| \times |\Sigma|$
مثلا با سه متغیر S, A, B و دو الفبای a, b حداکثر ۶ قاعده تولید در گرامر ساده وجود دارد.

۲- هر گرامر ساده، غیرمبهم است. (عکس این مطلب درست نیست).

مثلاً رشته $w = aabb$ توسط گرامر ساده G تولید می‌شود. از آنجا که این رشته با a آغاز می‌شود و متغیر S فقط یک قاعده با الفبای a دارد تنها راه برای تولید w استفاده از $S \rightarrow aS$ است. به همین ترتیب در هر مرحله با توجه به الفبایی که نیاز داریم، قاعده اشتقاق به صورت یکتا مشخص می‌شود.

G:

$$\begin{aligned} S &\rightarrow aS \mid bA & S &\rightarrow aS \rightarrow aaS \rightarrow aabA \rightarrow aabb \\ A &\rightarrow aAB \mid b \\ B &\rightarrow b \end{aligned}$$

۳- برای گرامر ساده مرتبه زمانی الگوریتم تجزیه (parse) و تعیین عضویت، $O(n)$ است.

با توجه به توضیحات قسمت ۲، اگر G گرامر ساده باشد، جستجوی کامل و بررسی تمام انشعاب‌های درخت لازم نیست و فقط یک انشعاب از درخت ممکن است بتواند w را تولید کند. پس در زمان $O(n)$ عضویت یا عدم عضویت w در $L(G)$ معلوم می‌شود.

توضیح تکمیلی

برای مثال گرامر ساده G را داریم. می‌خواهیم عضویت یا عدم عضویت رشته $w = bab$ در $L(G)$ را بررسی کنیم. با توجه به اولین الفبا فقط با شروع از قاعده $S \rightarrow bA$ ممکن است w تولید شود. به همین ترتیب در مرحله دوم فقط قاعده $A \rightarrow aB$ می‌تواند دومین حرف را تولید کند. در سومین گام متوجه می‌شویم که قاعده ای به فرم $B \rightarrow bX$ نداریم پس w عضو $L(G)$ نیست. پس احتیاجی به بررسی تمام شاخه های درخت نداریم. در سه گام جستجو متوقف شد. $|w| = 3$

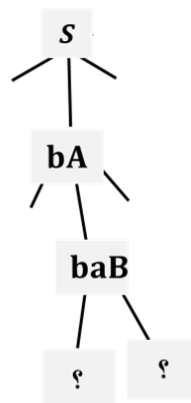
G :

$S \rightarrow aAB \mid bA \mid cBB$

$A \rightarrow aB \mid bD \mid cA$

$B \rightarrow aBD \mid c$

$D \rightarrow cBD \mid b$



توجه:

گرامر ساده ← گرامر غیرمبهم

گرامر ساده → // گرامر غیرمبهم

مثلا G غیرمبهم است اما ساده نیست:

G :

$S \rightarrow aSb \mid \lambda$

فصل پنجم: گرامر های مستقل از متن

جلسه دوازدهم بخش دو: حل تست

علوم کامپیوتر ۹۱

زبان مستقل از متن $L = \{0^n 1^m \mid m \neq n\}$ را در نظر بگیرید. قرار دهید:

$$r = \min\{|V(G)| : G \text{ گرامری مستقل از متن برای } L \text{ است}\}$$

مقدار r کدام است؟

۱ (۱)

۲ (۲)

۳ (۳)

۴ (۴)

پاسخ : گزینه ۳

سوال آن است که هر گرامر مستقل از متن تولید کننده زبان L حداقل چند متغیر خواهد داشت؟ رشته های عضو این زبان در دو دسته کلی قرار دارند: $n > m$ و $m > n$. برای تولید هر کدام به یک متغیر مجزا نیاز داریم. یک متغیر نیز برای شروع لازم داریم. با کمی دقت متوجه می شویم گرامر مستقل از متنی با کمتر از سه متغیر نمی تواند L را تولید کند.

$$\begin{aligned} G: & \quad L(G) = L = \{0^n 1^m \mid m \neq n\} \\ S & \rightarrow 0S1 \mid A \mid B \\ A & \rightarrow 1A \mid 1 \\ B & \rightarrow 0B \mid 0 \end{aligned}$$

فرض کنید زبان گرامر $G(\{a,b\}, V, S, P)$ برابر با تمام رشته هایی از $\{a,b\}^*$ باشد که تعداد حروف a و b آن ها برابر است. گرامر $G' = (\{a,b\}, V \cup \{S', A\}, P \cup P')$ برای زبان تمام رشته هایی که تعداد حروف a از b بیشتر باشد را در نظر بگیرید که $A \notin V$. کدام گزینه در مورد قواعد P' صحیح است؟ علوم کامپیوتر ۹۵

$$P' = \{S' \rightarrow AS'A \mid SS' \mid \lambda, A \rightarrow aA \mid a\} \quad (۱)$$

$$P' = \{S' \rightarrow SAS \mid SS' \mid \lambda, A \rightarrow aA \mid a\} \quad (۲)$$

$$P' = \{S' \rightarrow ASS' \mid SS'A \mid \lambda, A \rightarrow aA \mid a\} \quad (۳)$$

$$P' = \{S' \rightarrow SAS' \mid \lambda, A \rightarrow aA \mid a\} \quad (۴)$$

پاسخ: گزینه ها نادرستند

اولاً λ نباید توسط گرامر P' تولید شود پس قاعده $S' \rightarrow \lambda$ نباید وجود داشته باشد. ثانياً گزینه سه فقط رشته هایی از زبان مورد نظر را تولید می کند که با الفبای a شروع شوند یا به a ختم شوند. در حالی که رشته هایی مانند ba^4b نیز باید تولید شوند یعنی باید بتوانیم تعدادی الفبای a در بین حروف رشته ای که تعداد یکسان a و b دارد اضافه کنیم: پس علاوه بر $S' \rightarrow ASS' \mid SS'A$ باید قاعده $S' \rightarrow SAS'$ نیز داشته باشیم.

۴۹- کدام مورد، درست است؟ مهندسی کامپیوتر ۱۴۰۳

- (۱) الگوریتم پویش (parsing) برای زبان های مستقل از متن، همیشه از مرتبه نمایی است و به فرم گرامر وابسته نیست.
- (۲) اگر گرامر یک زبان مستقل از متن، به فرم نرمال چامسکی باشد، آنگاه می توان الگوریتم پویش (parsing) با مرتبه خطی از طول رشته های آن زبان داشت.
- (۳) اگر گرامر یک زبان مستقل از متن، گرامری ساده (s-grammar) باشد، آنگاه می توان الگوریتم پویش (parsing) با مرتبه خطی از طول رشته های آن زبان داشت.
- (۴) اگر قوانین لامبدا (اپسیلون) (λ -Productions) را از یک گرامر یک زبان مستقل از متن حذف کنیم، آنگاه می توان الگوریتم پویش (parsing) با مرتبه خطی از طول رشته های آن زبان داشت.

پاسخ: گزینه ۳

گزینه ۱ غلط است زیرا هم مرتبه زمانی و هم توقف الگوریتم وابسته به نوع گرامر مستقل از متن است. گزینه ۲ درست نیست (در ادامه خواهیم دید که برای نرمال چامسکی الگوریتمی پویا از مرتبه $O(n^3)$ وجود دارد که n طول رشته است.) گزینه ۳ درست است و علت آن در طول درس گفته شده. گزینه ۴ نادرست است. اگر قوانین لامبدا و همچنین قواعد تولید واحد را حذف کنیم، گرامر غیرانقباضی می شود و از توقف الگوریتم جستجو برای هر w اطمینان می یابیم اما مرتبه زمانی را نمایی است: $O(|P|^{2n+1})$.

زبان گرامر مستقل از متن زیر کدام است؟ مهندسی کامپیوتر ۹۷

$$S \rightarrow aB \mid bA$$

$$A \rightarrow a \mid aS \mid bAA$$

$$B \rightarrow b \mid bS \mid aBB$$

- (۱) شامل تمام رشته‌هایی که دارای تعداد a ها و b های برابر هستند.
- (۲) شامل تمام رشته‌هایی که دارای تعداد a های زوج و تعداد b های فرد هستند.
- (۳) شامل تمام رشته‌هایی که دارای تعداد a های فرد و تعداد b های زوج هستند.
- (۴) شامل تمام رشته‌هایی که دارای تعداد a های زوج و تعداد b های زوج هستند.

پاسخ: گزینه ۱ (البته باید می‌گفت رشته‌های نا تهی که تعداد a و b برابری دارند)

متغیرهای A و B در نهایت به الفباهای a و b تبدیل می‌شوند: $A \rightarrow a$ و $B \rightarrow b$.

اگر با $S \rightarrow aB$ آغاز کنیم در ادامه متغیر B رشته‌هایی را تولید می‌کند که تعداد b یکی بیشتر باشد. پس در نهایت تعداد a و b برابر می‌شود.

اگر با $S \rightarrow bA$ آغاز کنیم در ادامه متغیر A رشته‌هایی را تولید می‌کند که تعداد a یکی بیشتر باشد. پس در نهایت تعداد a و b برابر می‌شود.

تمام رشته‌های نا تهی که تعداد یکسانی a و b دارند تولید می‌شوند.

۶۰. در گرامر مستقل از متن G هیچ سمبل غیر پایانی A وجود ندارد به طوری که $A \xRightarrow{+} UAV$. کدام صحیح است؟

- (۱) یک زبان منظم را معرفی می کند.
- (۲) زبان معادل آن منظم نیست.
- (۳) زبان معادل آن بی پایان و نامنظم است.
- (۴) زبان معادل آن بی پایان ولی منظم است.

مهندسی کامپیوتر ۹۰

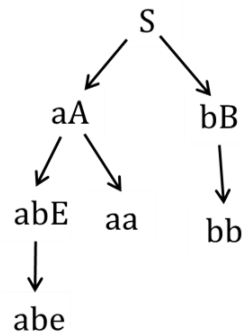
پاسخ : گزینه ۱

گرامر G هیچ متغیری ندارد که حین اشتقاق به صورت بازگشتی به آن برگردیم. بنابراین مجموعه رشته هایی که تولید می کند، متناهی است. $L(G)$ متناهی است و هر زبان متناهی، منظم است.

مثلا: گرامر مقابل تعدادی متناهی رشته تولید می کند. درخت جستجوی کامل تمام رشته های قابل تولید را نشان داده است.

G :
 $S \rightarrow aA \mid bB$
 $A \rightarrow bE \mid a$
 $B \rightarrow b$
 $E \rightarrow e$

$L(G) = \{aa, bb, abe\}$



۵۳- کدام زبان ذاتاً مبهم است؟ مهندسی کامپیوتر ۹۴

$$L_1 = \{a^n b^{2n} c \mid n \geq 0\} \cup \{a^{2n} b^n d \mid n \geq 0\} \quad (۱)$$

$$L_2 = \{a^n b^m c^p \mid n \neq m\} \cup \{a^n b^m c^p \mid p \neq m\} \quad (۲)$$

$$L_3 = \{a^n b a^{2n} \mid n \geq 0\} \cup \{a^{2n} b a^n \mid n \geq 0\} \quad (۳)$$

$$L_4 = \{w \in (a+b)^* \mid w \text{ زیررشته } abab \text{ ندارد}\} \cup \{w \in (a+b)^* \mid w \text{ زیررشته } baba \text{ ندارد}\} \quad (۴)$$

پاسخ : گزینه ۲

گزینه ۴ اجتماع دو زبان منظم است پس منظم است. منظم ها گرامر غیرمبهم دارند.

در گزینه ۱ رشته ها در دو دسته کاملاً مجزا قرار دارند $a^n b^{2n} c$ را با S_1 و $a^{2n} b^n d$ را با S_2 تولید می کنیم و به سادگی یک گرامر غیرمبهم به دست می آید. گزینه ۳ نیز مشابه گزینه ۱ است با این تفاوت که در حالت $n = 0$ رشته b تنها عضو مشترک دو مجموعه است. پس باید این رشته را از یک مسیر جداگانه مثلاً S_3 و سایر رشته را با کمک S_1 و S_1 تولید کنیم. به این ترتیب گزینه ۳ هم گرامر غیرمبهم خواهد داشت.

اما گزینه ۲ ذاتاً مبهم است. دو مجموعه داده شده بی نهایت رشته مشترک دارند. رشته هایی که

در آنها هم $p \neq m$ هم $n \neq m$ تعدادشان بی نهایت است. (رشته هایی مانند $a^2 b^3 c^5$).

این زبان ذاتاً مبهم است.

توضیح تکمیلی مساله قبل: نحوه یافتن گرامر های غیرمبهم برای گزینه های ۱ و ۳:

$$L_1 = \{a^n b^{2n} c \mid n \geq 0\} \cup \{a^{2n} b^n d \mid n \geq 0\}$$

G₁: غیرمبهم

$$S \rightarrow S_1 c \mid S_2 d$$

$$S_1 \rightarrow a S_1 b^2 \mid \lambda$$

$$S_2 \rightarrow a^2 S_2 b \mid \lambda$$

$$L_3 = \{a^n b a^{2n} \mid n \geq 0\} \cup \{a^{2n} b a^n \mid n \geq 0\}$$

G'₃:

$$S \rightarrow S_1 \mid S_2$$

$$S_1 \rightarrow a S_1 a^2 \mid b$$

$$S_2 \rightarrow a^2 S_2 a \mid b$$

G'₃ مبهم است چون b
از دو طریق تولید می شود.

G₃:

$$S \rightarrow S_1 \mid S_2 \mid S_3$$

$$S_1 \rightarrow a S_1 a^2 \mid a b a^2$$

$$S_2 \rightarrow a^2 S_2 a \mid a^2 b a$$

$$S_3 \rightarrow b$$

G₃ غیرمبهم است

۵۴- کدام گزینه درباره گرامرهای G_1 و G_2 صحیح است؟ مهندسی کامپیوتر ۹۳

$$G_1: S \rightarrow SS \mid (S) \mid \lambda$$

$$G_2: S \rightarrow (S)S \mid \lambda$$

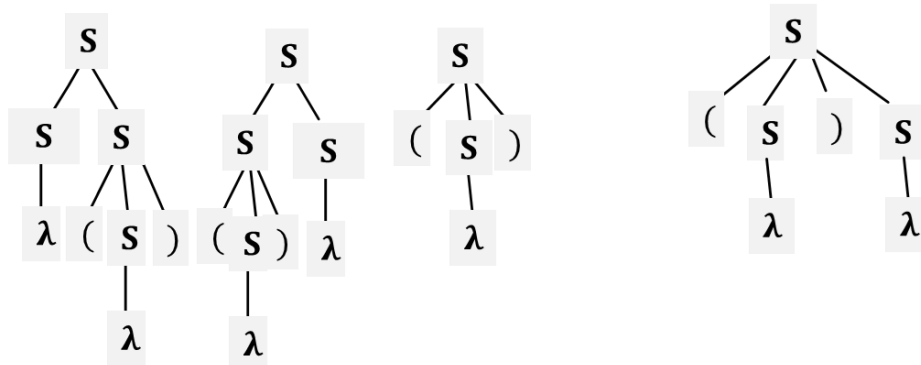
- (۱) هر دو G_1 و G_2 مبهم و با هم معادلند.
 (۲) G_1 مبهم و G_2 غیر مبهم ولی با هم معادلند.
 (۳) هر دو G_1 و G_2 مبهم ولی معادل نیستند.
 (۴) G_1 مبهم و G_2 غیر مبهم هستند و معادل نیستند.

پاسخ: گزینه ۲

هر دو گرامر تمام رشته های خوش ساخت با پرانتزهای باز و بسته را تولید می کنند. خوش ساخت یعنی هر پرانتزی که باز شد حتما بسته شود. پس گرامرها با هم معادل هستند: $L(G_1) = L(G_2)$. در G_2 قاعده $S \rightarrow (S)S$ اولویت تولید پرانتزها را مشخص کرده است. اما در G_1 چنین قاعده ای نداریم. به همین دلیل G_2 غیر مبهم و G_1 مبهم است. در شکل زیر می بینید که رشته $()$ در G_1 به چند روش تولید می شود اما همین رشته در G_2 فقط به یک روش تولید می شود.

$$G_1: S \rightarrow SS \mid (S) \mid \lambda$$

$$G_2: S \rightarrow (S)S \mid \lambda$$





-13-

فصل ششم: ساده سازی گرامر های مستقل از متن - فرم های نرمال
جلسه سیزدهم بخش یک: ساده سازی گرامر مستقل

ساده سازی گرامر مستقل

می‌خواهیم گرامر مستقل از متن را تا حد امکان ساده کنیم تا به گرامر غیرانقباضی تبدیل شود و سپس قابل تبدیل به فرم‌های نرمال باشند. البته اگر $\lambda \in L(G)$ باشد گرامر مستقل از متن G نمی‌تواند کاملاً غیرانقباضی شود زیرا قاعده $S \rightarrow \lambda$ انقباضی است. در این نمونه‌ها اقدامات خود را به $L(G) - \{\lambda\}$ محدود می‌کنیم. متغیر شروع S_0 را با قواعد λ $S_0 \rightarrow S$ به گرامر اضافه می‌کنیم. حالا با شروع از S زبان $L(G) - \{\lambda\}$ تولید می‌شود.

مثلاً برای زبان $\{a^n b^n \mid n = m\}$ گرامر مستقل از متن G را داریم:

$$G: S \rightarrow aSb \mid \lambda$$

تولید رشته تهی را به محول می‌کنیم و با شروع از S رشته‌های غیرتهی را تولید می‌کنیم:

$G:$

$$S_0 \rightarrow S \mid \lambda$$

$$S \rightarrow aSb \mid ab$$

قاعده جایگزینی: یک قاعده ساده داریم که ابزار کار ما در ادامه است:

اگر قاعده تولید به فرم کلی

$$A \rightarrow x_1 B x_2$$

داشته باشیم و متغیر B دارای اشتقاق‌های

$$B \rightarrow y_1 \mid y_2 \dots \mid y_k$$

باشد می‌توانیم با جایگزین کردن y_1 و y_2 و $\dots$ و y_k به جای B قواعدی به فرم

$$A \rightarrow x_1 y_1 x_2 \mid x_1 y_2 x_2 \dots \mid x_1 y_k x_2$$

را به جای قاعده $A \rightarrow x_1 B x_2$ بنویسیم و گرامر معادلی به دست آوریم.

مثال: گراف G را داریم. نتیجه جایگزینی A در قواعد $S \rightarrow aA \mid bAS$ رسیدن به گراف معادل G' است:

$$\begin{array}{ll} G : & G' : \\ S \rightarrow aA \mid bAS \mid bBb & \Rightarrow S \rightarrow aBbc \mid bBbcS \mid aa \mid baS \mid bBb \\ A \rightarrow Bbc \mid a & A \rightarrow Bbc \mid a \\ B \rightarrow b \mid bb & B \rightarrow b \mid bb \end{array}$$

$$L(G') = L(G)$$

ساده سازی گرامر مستقل از متن

برای ساده سازی گرامر مستقل از متن، این اقدامات را به ترتیب انجام می دهیم:

۱- حذف قواعد تولید λ

۲- حذف قواعد تولید واحد

۳- حذف قواعد تولید زائد

توضیح کلی:

گرامر مستقل از متن ممکن است سه مشکل داشته باشد که باعث انقباضی بودن یا پیچیده بودن آن شود:

مشکل (۱) وجود قواعد تولید λ مانند: $A \rightarrow \lambda$ و $B \rightarrow \lambda$

مشکل (۲) وجود قواعد واحد (یکانی) مانند $A \rightarrow B$ و $B \rightarrow E$

مشکل (۳) وجود قواعد غیر مفید و زائد که دو نوعند:

اول: متغیرهایی که با شروع از آنها رشته ای تولید نمی شود (خروجی ندارند).

دوم: متغیرهایی که از S به آنها دسترسی نداریم (ورودی ندارند).

حذف (۱) می تواند باعث ایجاد مشکلات (۲) و (۳) شود.

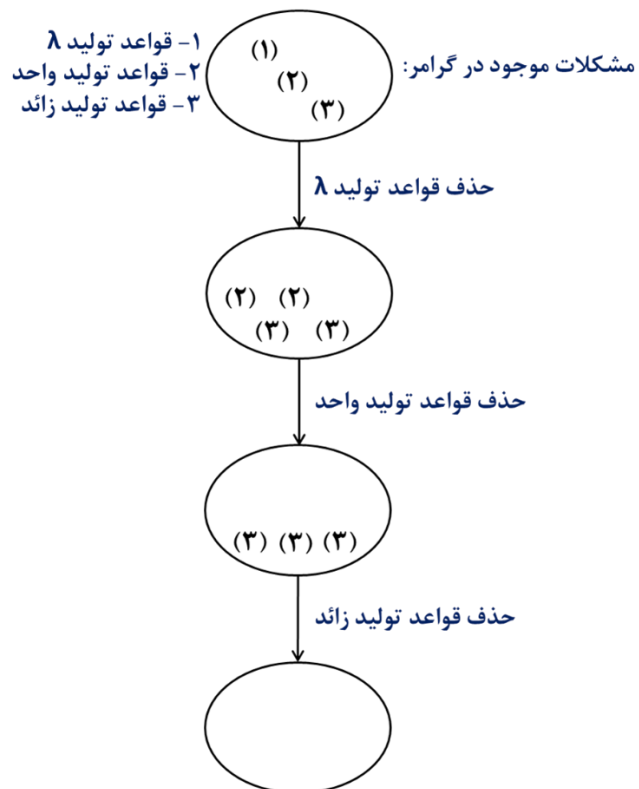
حذف (۲) می تواند باعث ایجاد مشکل (۳) شود.

اما

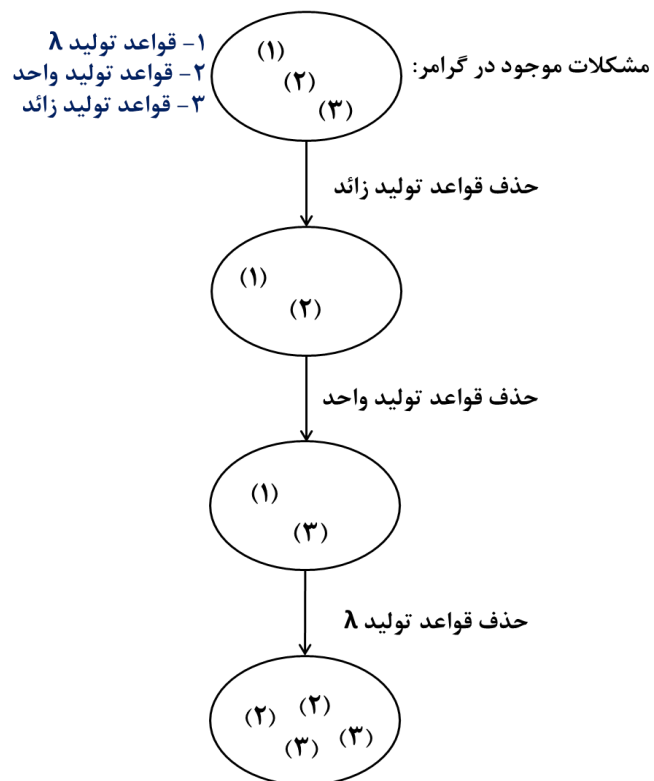
حذف (۳) نمی تواند باعث ایجاد مشکلات (۱) و (۲) شود.

حذف (۲) نمی تواند باعث ایجاد مشکل (۱) شود.

ترتیب درست ساده سازی و حذف مشکلات:



یک ترتیب نادرست ساده سازی و حذف مشکلات:



۱- نحوه حذف قواعد تولید λ

ابتدا متغیرهای پوچ شونده را تشخیص دهیم یعنی متغیرهایی که مستقیم یا غیر مستقیم λ را تولید کنند. سپس جایگزینی λ به جای پوچ شونده‌ها و به دست آوردن قواعد جدید. مثلاً اگر A و B پوچ شونده باشند قاعده $S \rightarrow ABab$ به ۴ قاعده تبدیل می‌شود:

$$S \rightarrow ABab \mid Bab \mid Aab \mid ab$$

در پایان:

اگر S پوچ شونده باشد $\lambda \in L(G)$ خواهد بود در اینصورت از متغیر S_0 با قواعد $\lambda \mid S \rightarrow S_0$ استفاده می‌کنیم تا $\lambda \rightarrow S$ حذف شود.

مثال - حذف قواعد تولید λ در گرامر G

B و C تولید λ دارند. قاعده $A \rightarrow BC$ داریم پس A هم پوچ شونده است. قاعده $S \rightarrow AC$ داریم پس S پوچ شونده است. حالا در سمت راست تمام قواعد به جای متغیرهای پوچ شونده تهی را قرار می‌دهیم.

نکته:

هر قاعده که در سمت راست آن k متغیر پوچ شونده باشند به 2^k قاعده تبدیل می‌شود. در ضمن چون S نیز پوچ شونده است از شروع λ $S_0 \rightarrow S$ استفاده می‌کنیم.

G:	$\Rightarrow$	G':
$S \rightarrow ABab \mid AC$		$S_0 \rightarrow S \mid \lambda$
$A \rightarrow a \mid BC$		$S \rightarrow ABab \mid AC \mid Bab \mid Aab \mid ab \mid A \mid C$
$B \rightarrow b \mid \lambda$		$A \rightarrow a \mid BC \mid B \mid C$
$C \rightarrow E \mid \lambda$		$B \rightarrow b$
$E \rightarrow aEb \mid b$		$C \rightarrow E$
		$E \rightarrow aEb \mid b$

$$L(G') = L(G)$$

مثال - حذف قواعد تولید λ در گرامر G

فقط B پوچ شونده است. با جایگزینی λ به جای B قواعد تازه‌ای به دست می‌آید.

$$\begin{array}{ll}
 G: & G': \\
 S \rightarrow Aab \mid aBb & \Rightarrow S \rightarrow Aab \mid aBb \mid ab \\
 A \rightarrow aS \mid bB & A \rightarrow aS \mid bB \mid b \\
 B \rightarrow bA \mid \lambda & B \rightarrow bA
 \end{array}$$

$$L(G') = L(G)$$

مثال - حذف قواعد تولید λ در گرامر G

در این مثال مشاهده می‌کنید که حذف قواعد تولید λ ممکن است باعث ایجاد قواعد تولید واحد و قواعد زائد (متغیرهای زائد) شود.

حذف قواعد تولید λ $\Leftarrow$ ایجاد قواعد تولید واحد
ایجاد قواعد تولید زائد

$$\begin{array}{ll}
 G: & \Rightarrow G': \\
 S \rightarrow Aa \mid AB & S_0 \rightarrow S \mid \lambda \\
 A \rightarrow aS \mid a \mid \lambda & S \rightarrow Aa \mid AB \mid A \mid B \\
 B \rightarrow \lambda & A \rightarrow aS \mid a
 \end{array}$$

$$L(G') = L(G)$$

۲- شیوه حذف قواعد تولید واحد

ابتدا جدا کردن و خارج کردن قواعد واحد از سایر قواعد گرامر. سپس تشکیل گراف وابستگی با توجه کردن به قواعد واحد. در گام بعد، اگر در گراف وابستگی از A به B می‌توان رسید، تمام قواعد تولید B باید به قواعد تولید A اضافه شوند.

مثال - حذف قواعد λ و قواعد یکانی (واحد) از گرامر G

قاعده تولید λ ندارد.

قواعد واحد $A \rightarrow B$ ، $B \rightarrow D$ و $D \rightarrow B$ در گرامر وجود دارند. گراف وابستگی این است:



قواعد واحد را از گرامر حذف کنیم.

حالا با توجه به گراف وابستگی:

قواعد تولید D و B هر کدام به دیگری اضافه می‌شود.
همه قواعد تولید B و D به قواعد تولید A اضافه می‌شود.

G :

$S \rightarrow Aab \mid aBb$

$A \rightarrow aS \mid B$

$B \rightarrow bA \mid D$

$D \rightarrow AA \mid B \mid aD$

$\Rightarrow$

G' :

$S \rightarrow Aab \mid aBb$

$A \rightarrow aS \mid bA \mid AA \mid aD$

$B \rightarrow bA \mid AA \mid aD$

$D \rightarrow AA \mid aD \mid bA$

مثال - حذف قواعد تولید واحد

در این مثال خواهیم دید که حذف قواعد واحد می‌تواند باعث ایجاد قواعد و متغیرهای زائد شود:

حذف قواعد تولید واحد $\Leftarrow$ ایجاد قواعد تولید زائد

قواعد واحد $A \rightarrow B$ و $B \rightarrow D$ را از گرامر جدا کنیم. گراف وابستگی این است:

$$A \longrightarrow B \longrightarrow D$$

همه قواعد تولید D به قواعد تولید B اضافه می‌شوند. قواعد تولید B و D به قواعد تولید A اضافه می‌شوند.

G :	$\Rightarrow$	G' :
$S \rightarrow Aab$		$S \rightarrow Aab$
$A \rightarrow aS \mid B$		$A \rightarrow aS \mid aa \mid b$
$B \rightarrow D \mid b$		$B \rightarrow b \mid aa$
$D \rightarrow aa$		$D \rightarrow aa$

گرامر G' قاعده واحد ندارد. اما متغیرهای B و D به متغیرهای زائد تبدیل شدند.

۳- شیوه حذف قواعد و متغیرهای زائد

متغیرهای زائد (غیرمفید) دو نوع دارند.

الف: متغیرهایی که با شروع از آنها هیچ رشته ای تولید نمی‌شود (خروجی ندارند)

ب: متغیرهایی که با شروع از S در دسترس نیستند. (ورودی ندارند)

حذف آنها باید به ترتیب (الف_ب) باشد. زیرا حذف نوع (الف) ممکن است باعث تولید نوع (ب) شود.

مثال: حذف متغیرهای زائد از گرامر G:

با شروع از D هیچ رشته ای به دست نمی‌آید. پس D متغیر زائد نوع (الف) است.

در سمت راست هیچ قاعده‌ای متغیر نیامده است پس B متغیر زائد نوع (ب) است.

ابتدا D و همه قواعد شامل آن را حذف کنیم.

سپس B و همه قواعد شامل آن را حذف کنیم.

G :	حذف D ⇒	G' :	حذف B ⇒	G'' :
$S \rightarrow Aab \mid D$		$S \rightarrow Aab$		$S \rightarrow Aab$
$A \rightarrow abS \mid DaA \mid a$		$A \rightarrow abS \mid a$		$A \rightarrow abS \mid a$
$B \rightarrow Bba \mid b$		$B \rightarrow Bba \mid b$		
$D \rightarrow aaD \mid Db$				

$$L(G) = L(G') = L(G'')$$

پیچیدگی گرامر مستقل از متن

هر قاعده تولید $V \rightarrow \alpha$ در گرامر مستقل از متن دارای عدد پیچیدگی $1 + |\alpha|$ است. مجموع این مقادیر را پیچیدگی گرامر می‌نامیم.

$$\text{complexity}(G) = \sum_{(V \rightarrow \alpha) \in P} (1 + |\alpha|)$$

مثال: پیچیدگی گرامر مقابل را حساب کنیم:

G :

$S \rightarrow Aab \mid bS$

$A \rightarrow aA \mid a$

$B \rightarrow Db$

$$\text{complexity}(G) = (1 + 3) + (1 + 2) + (1 + 2) + (1 + 1) + (1 + 2) = 15$$

نکته: دقت کنید ۵ قاعده تولید داریم و در مجموع سمت راست آنها رشته به طول ۱۰ است. مجموع این دو عدد ۱۵ است. یعنی پیچیدگی از این طریق هم به دست می‌آید.

تاثیر اقدامات ساده سازی بر پیچیدگی گرامر:

هنگام ساده سازی گرامر این اقدامات را انجام دادیم:

۱- حذف قواعد تولید λ

۲- حذف قواعد تولید واحد

۳- حذف قواعد تولید زائد

دو اقدام اول مقدار پیچیدگی را افزایش می‌دهند یا ثابت نگه می‌دارند. هرچند این دو اقدام باعث می‌شوند گرامر غیرانقباضی شود. اقدام سوم پیچیدگی را کاهش می‌دهد یا ثابت نگه می‌دارد.

۱- حذف قواعد تولید λ $\Uparrow$

۲- حذف قواعد تولید واحد $\Uparrow$ تاثیر اقدامات سه گانه بر پیچیدگی گرامر

۳- حذف قواعد تولید زائد $\Downarrow$

توجه:

انجام سه اقدام فوق، لزوماً نمی‌تواند گرامر را به گرامر ساده (S – grammar) تبدیل کنند.

فصل ششم: ساده سازی گرامر های مستقل از متن – فرم های نرمال
جلسه سیزدهم بخش دوم:
فرم های نرمال گرامر مستقل از متن: چامسکی و گریباخ

فرمهای نرمال چامسکی و گریباخ

توضیح مقدماتی:

هر گرامر مستقل از متن را لزوماً نمی‌توان به گرامر ساده معادل تبدیل کرد. اما هر گرامر مستقل از متن (فاقد تولید λ) را می‌توان به گرامر های معادل در فرم نرمال چامسکی و گریباخ تبدیل کرد. در این جلسه با این فرم های نرمال و نحوه رسیدن به آنها آشنا می‌شویم.

در مسیر تبدیل گرامر G به فرم های نرمال ابتدا سه اقدام پیشین را انجام می‌دهیم:

۱- حذف قواعد تولید λ

۲- حذف قواعد تولید واحد

۳- حذف قواعد تولید زائد

در مورد فرم گریباخ، یک اقدام دیگر هم لازم است که حذف بازگشتی چپ است.

قواعد بازگشتی چپ

دو نوع بازگشتی چپ داریم. بازگشتی چپ مستقیم مانند:

$$A \rightarrow Aab \mid Ab \mid ABb$$

بازگشتی چپ غیر مستقیم مانند:

$$A \rightarrow Bab$$

$$B \rightarrow Aaa$$

بازگشت چپ غیر مستقیم با جایگزینی باعث بازگشت چپ مستقیم می‌شود.

حذف قاعده بازگشتی چپ

این قواعد بازگشتی چپ را در نظر بگیرید:

$$A \rightarrow Ax_1 \mid Ax_2 \dots \mid Ax_n$$

$$A \rightarrow y_1 \mid y_2 \dots \mid y_k$$

با شروع از متغیر A قواعد فوق چنین عباراتی را تولید می‌کنند:

$$A \xRightarrow{*} y_m x_k \dots x_j x_i$$

می‌توانیم همین عبارات را بدون استفاده از بازگشتی چپ و با استفاده از بازگشتی راست تولید کنیم:

$$A \rightarrow y_1 \mid y_2 \dots \mid y_k$$

$$A \rightarrow y_1 A_1 \mid y_2 A_1 \dots \mid y_k A_1$$

$$A_1 \rightarrow x_1 A_1 \mid x_2 A_1 \dots \mid x_n A_1$$

$$A_1 \rightarrow x_1 \mid x_2 \dots \mid x_n$$

خلاصه‌ی حذف بازگشتی چپ: (peter linz) تمرین ۲۳ فصل ۶ قسمت ۱

$$\begin{array}{c}
 A \rightarrow Ax_1 \mid Ax_2 \dots \mid Ax_n \mid y_1 \mid y_2 \dots \mid y_k \\
 \Rightarrow \\
 \begin{array}{l}
 A \rightarrow y_1 A_1 \mid y_2 A_1 \dots \mid y_k A_1 \mid y_1 \mid y_2 \dots \mid y_k \\
 A_1 \rightarrow x_1 A_1 \mid x_2 A_1 \dots \mid x_n A_1 \mid x_1 \mid x_2 \dots \mid x_n
 \end{array}
 \end{array}$$

مثال: حذف قاعده بازگشتی چپ:

$$\begin{array}{c}
 A \rightarrow Aab \mid Ab \mid ab \mid c \\
 \Rightarrow \\
 \begin{array}{l}
 A \rightarrow abA_1 \mid cA_1 \mid ab \mid c \\
 A_1 \rightarrow abA_1 \mid bA_1 \dots \mid ab \mid b
 \end{array}
 \end{array}$$

البته در حالت کلی x_i و y_i می‌توانند شامل متغیر هم باشند.

فرم نرمال چامسکی: (Chomsky normal form)

گرامر مستقل از متن G به فرم نرمال چامسکی است اگر تمام قواعد آن به فرم های $A \rightarrow a$ و $A \rightarrow BC$ باشند.

با این تعریف، گرامر چامسکی نمی تواند λ را تولید کند. اما هر گرامر مستقل از متن G که $\lambda \notin L(G)$ دارای گرامر معادل به فرم نرمال چامسکی است.

تعریف فرم نرمال چامسکی در منبع (sipser) قدری متفاوت است :

گرامر مستقل از متن G به فرم نرمال چامسکی است اگر تمام قواعد آن به فرم های $A \rightarrow BC$ و $A \rightarrow a$ باشند. به استثنای متغیر شروع (S) که می تواند تولید λ داشته باشد: $S \rightarrow \lambda$. در ضمن از آنجا که اجازه دادیم قاعده $S \rightarrow \lambda$ در گرامر باشد پس متغیرهای سمت راست یعنی B و C نباید متغیر شروع باشند: $B \neq S$ و $C \neq S$. (زیرا نمی خواهیم قواعد تولید واحد بتوانند ظاهر شوند) با این تعریف، هر گرامر مستقل از متن G که $\lambda \in L(G)$ دارای گرامر معادل به فرم نرمال چامسکی است. حتی اگر $\lambda \in L(G)$ باشد.

توجه:

رفرنس Linz گرامر نرمال چامسکی را برای $L(G) - \{\lambda\}$ مد نظر قرار می دهد پس گرامر چامسکی نباید λ تولید کند.

رفرنس Sipser گرامر نرمال چامسکی را برای $L(G)$ مد نظر قرار می دهد پس گرامر چامسکی اجازه دارد λ تولید کند. اما این تولید فقط باید به وسیله S باشد.

فرم نرمال چامسکی در sipser :

DEFINITION 2.8

A context-free grammar is in *Chomsky normal form* if every rule is of the form

$$A \rightarrow BC$$

$$A \rightarrow a$$

where a is any terminal and A , B , and C are any variables—except that B and C may not be the start variable. In addition, we permit the rule $S \rightarrow \epsilon$, where S is the start variable.

THEOREM 2.9

Any context-free language is generated by a context-free grammar in Chomsky normal form.

مثال Linz:

طبق تعریف لینز گرامر G_1 فرم نرمال چامسکی است. گرامر G_2 فرم نرمال چامسکی نیست.
طبق تعریف سیپسر گرامر G_1 فرم نرمال چامسکی محسوب نمی‌شود زیرا S در سمت راست قواعد تولید ظاهر شده است.

The grammar

$$\begin{aligned} G_1 \quad S &\rightarrow AS|a, \\ A &\rightarrow SA|b \end{aligned}$$

is in Chomsky normal form. The grammar

$$\begin{aligned} G_2 \quad S &\rightarrow AS|AAS, \\ A &\rightarrow SA|aa \end{aligned}$$

is not; both productions $S \rightarrow AAS$ and $A \rightarrow aa$ violate the conditions

تبدیل گرامر مستقل از متن به گرامر معادل در فرم نرمال چامسکی:

مرحله ۱:

متغیر شروع جدیدی با قاعده $S_0 \rightarrow S$ به گرامر اضافه کنیم. اگر $\lambda \in L(G)$ قاعده $S_0 \rightarrow \lambda$ نیز اضافه خواهد شد.

مرحله ۲:

اقدامات سه گانه برای تبدیل گرامر به گرامر غیرانقباضی و ساده سازی را انجام می دهیم:

۱-۲- حذف قواعد تولید λ

۲-۲- حذف قواعد تولید واحد

۳-۲- حذف قواعد تولید زائد

مرحله ۳:

فرض کنید قاعده $A \rightarrow AabB$ را در گرامر داشته باشیم. به جای الفباهای a و b از متغیرهای X_a و X_b استفاده می کنیم. قاعده $A \rightarrow AX_aX_bB$ را با معرفی متغیرهای کمکی به چند قاعده در فرم چامسکی تبدیل می کنیم:

$A \rightarrow AabB$	$\Rightarrow$	$A \rightarrow AY_1$ $Y_1 \rightarrow X_a Y_2$ $Y_2 \rightarrow X_b B$ $X_a \rightarrow a$ $X_b \rightarrow b$
----------------------	---------------	--

مثال: گرامر مستقل از متن G_6 را به گرامر معادل در فرم نرمال چامسکی تبدیل کنید:

مرحله ۱:

متغیر شروع S_0 جدیدی با قاعده $S_0 \rightarrow S$ به گرامر اضافه شد.

1. The original CFG G_6 is shown on the left. The result of applying the first step to make a new start variable appears on the right.

$$\begin{array}{lcl}
 G_6: & S \rightarrow ASA \mid aB \\
 & A \rightarrow B \mid S \\
 & B \rightarrow b \mid \varepsilon
 \end{array}
 \Rightarrow
 \begin{array}{l}
 S_0 \rightarrow S \\
 S \rightarrow ASA \mid aB \\
 A \rightarrow B \mid S \\
 B \rightarrow b \mid \varepsilon
 \end{array}$$

مرحله ۲: حذف قواعد تولید ε

$$\begin{array}{lcl}
 S_0 \rightarrow S & 2. \text{ Remove } \varepsilon\text{-rules} & S_0 \rightarrow S \\
 S \rightarrow ASA \mid aB & \Rightarrow & S \rightarrow ASA \mid aB \mid a \mid SA \mid AS \mid S \\
 A \rightarrow B \mid S & & A \rightarrow B \mid S \mid \varepsilon \\
 B \rightarrow b \mid \varepsilon & & B \rightarrow b
 \end{array}$$

مرحله ۳: حذف قواعد تولید واحد در دو مرحله زیر

3a. Remove unit rules $S \rightarrow S$, shown on the left, and $S_0 \rightarrow S$, shown on the right.

$$\begin{array}{lcl}
 S_0 \rightarrow S & & S_0 \rightarrow S \mid ASA \mid aB \mid a \mid SA \mid AS \\
 S \rightarrow ASA \mid aB \mid a \mid SA \mid AS \mid S & & S \rightarrow ASA \mid aB \mid a \mid SA \mid AS \\
 A \rightarrow B \mid S & & A \rightarrow B \mid S \\
 B \rightarrow b & & B \rightarrow b
 \end{array}$$

3b. Remove unit rules $A \rightarrow B$ and $A \rightarrow S$.

$$\begin{array}{lcl}
 S_0 \rightarrow ASA \mid aB \mid a \mid SA \mid AS & & S_0 \rightarrow ASA \mid aB \mid a \mid SA \mid AS \\
 S \rightarrow ASA \mid aB \mid a \mid SA \mid AS & & S \rightarrow ASA \mid aB \mid a \mid SA \mid AS \\
 A \rightarrow B \mid S \mid b & & A \rightarrow S \mid b \mid ASA \mid aB \mid a \mid SA \mid AS \\
 B \rightarrow b & & B \rightarrow b
 \end{array}$$

مرحله پایانی: معرفی متغیرهای کمکی و تبدیل به فرم چامسکی:

$$\begin{array}{l}
 S_0 \rightarrow ASA \mid aB \mid a \mid SA \mid AS \\
 S \rightarrow ASA \mid aB \mid a \mid SA \mid AS \\
 A \rightarrow b \mid ASA \mid aB \mid a \mid SA \mid AS \\
 B \rightarrow b
 \end{array}
 \Rightarrow
 \begin{array}{l}
 S_0 \rightarrow AA_1 \mid UB \mid a \mid SA \mid AS \\
 S \rightarrow AA_1 \mid UB \mid a \mid SA \mid AS \\
 A \rightarrow b \mid AA_1 \mid UB \mid a \mid SA \mid AS \\
 A_1 \rightarrow SA \\
 U \rightarrow a \\
 B \rightarrow b
 \end{array}$$

مثال دیگر: گرامر مستقل از متن G را به گرامر معادل در فرم نرمال چامسکی تبدیل کنید:

$$\begin{array}{l}
 G: \quad S \rightarrow ASB \\
 \quad A \rightarrow aAS \mid a \mid \varepsilon \\
 \quad B \rightarrow SbS \mid A \mid bb
 \end{array}$$

مرحله ۱: متغیر شروع S_0 جدیدی با قاعده $S_0 \rightarrow S$ به گرامر اضافه شد.

$$\begin{array}{l}
 S_0 \rightarrow S \\
 S \rightarrow ASB \\
 A \rightarrow aAS \mid a \mid \varepsilon \\
 B \rightarrow SbS \mid A \mid bb
 \end{array}$$

مرحله ۲: حذف قواعد تولید ϵ

Eliminating $A \rightarrow \epsilon$

$$\begin{aligned} S_0 &\rightarrow S \\ S &\rightarrow ASB|SB \\ A &\rightarrow aAS|a|aS \\ B &\rightarrow SbS|A|bb|\epsilon \end{aligned}$$

remove $B \rightarrow \epsilon$

$$\begin{aligned} S_0 &\rightarrow S \\ S &\rightarrow ASB|SB|S|AS \\ A &\rightarrow aAS|a|aS \\ B &\rightarrow SbS|A|bb \end{aligned}$$

مرحله ۳: حذف قواعد تولید واحد

removing $B \rightarrow A$:

$$\begin{aligned} S_0 &\rightarrow S \\ S &\rightarrow ASB|SB|S|AS \\ A &\rightarrow aAS|a|aS \\ B &\rightarrow SbS|bb|aAS|a|aS \end{aligned}$$

remove $S \rightarrow S$:

$$\begin{aligned} S_0 &\rightarrow S \\ S &\rightarrow ASB|SB|AS \\ A &\rightarrow aAS|a|aS \\ B &\rightarrow SbS|bb|aAs|a|aS \end{aligned}$$

eliminate $S_0 \rightarrow S$:

$$\begin{aligned} S_0 &\rightarrow ASB|SB|AS \\ S &\rightarrow ASB|SB|AS \\ A &\rightarrow aAS|a|aS \\ B &\rightarrow SbS|bb|aAs|a|aS \end{aligned}$$

مرحله پایانی: معرفی متغیرهای کمکی و تبدیل به فرم چامسکی:

$$\begin{array}{ll}
 S_0 \rightarrow ASB|SB|AS & \Rightarrow S_0 \rightarrow AU_1|SB|AS \\
 S \rightarrow ASB|SB|AS & S \rightarrow AU_2|SB|AS \\
 A \rightarrow aAS|a|aS & A \rightarrow V_1U_3|a|V_1S \\
 B \rightarrow SbS|bb|aAs|a|aS & B \rightarrow SU_4|V_2V_2|V_1U_5|a|V_1S \\
 & U_1 \rightarrow SB \\
 & U_2 \rightarrow SB \\
 & U_3 \rightarrow AS \\
 & U_4 \rightarrow V_2S \\
 & U_5 \rightarrow AS \\
 & V_1 \rightarrow a \\
 & V_2 \rightarrow b
 \end{array}$$

فرم نرمال گریباخ : (Greibach normal form)

گرامر مستقل از متن G به فرم نرمال گریباخ است اگر تمام قواعد آن به فرم های زیر باشد:

$$A \rightarrow a \text{ و } A \rightarrow aX$$

$X \in V^*$ رشته ای دلخواه از متغیرها است.

توجه: در گرامر ساده نیز قواعد تولید به همین فرم بودند اما با یک شرط اضافی که هر جفت (A, a) حداکثر در یک قاعده تولید ظاهر شده باشند. پس هر گرامر ساده، به فرم گریباخ است اما هر گرامر گریباخ، یک گرامر ساده نیست.

گرامر ساده $\longleftarrow$ گرامر گریباخ

گرامر ساده $\not\rightarrow$ گرامر گریباخ

مثلا این گرامر به فرم گریباخ است اما گرامر ساده نیست چون متغیر A با الفبای b دو قاعده تولید دارد.

$G :$

$$S \rightarrow aASB \mid b$$

$$A \rightarrow bA \mid bB \mid a$$

$$B \rightarrow b$$

قضیه: برای هر زبان مستقل از متن L ، زبان $L - \{\lambda\}$ دارای گرامر فرم نرمال گریباخ است. گرامر فرم نرمال گریباخ نمی تواند رشته تهی را تولید کند.

تبدیل گرامر مستقل از متن به گرامر معادل در فرم نرمال گریباخ:

مرحله ۱:

اقدامات سه گانه برای ساده سازی و تبدیل گرامر به گرامر غیرانقباضی را انجام دهیم:

۱-۱- حذف قواعد تولید λ

۱-۲- حذف قواعد تولید واحد

۱-۳- حذف قواعد تولید زائد

مرحله ۲:

حذف قواعد بازگشتی چپ.

مرحله ۳:

با استفاده از متغیرهای کمکی همه قواعد را به فرم گریباخ تبدیل کنیم. برای مثال در قاعده

$$A \rightarrow cbSB$$

به جای الفبای b متغیر X_b قرار دهیم:

$$A \rightarrow cX_bSB$$

این قاعده به فرم گریباخ است. در قواعدی مانند:

$$S \rightarrow AB$$

با جایگزین کردن اشتقاق های A سعی کنیم به قواعدی برسیم که با یک الفبا آغاز شوند.

مثال: تبدیل گرامر مستقل از متن G به گرامر معادل در فرم نرمال گریباخ را انجام دهید.

G :

$$S \rightarrow AB \mid aBAcb$$

$$A \rightarrow Aa \mid cd$$

$$B \rightarrow bA \mid ca$$

پاسخ: گرامر G قواعد تولید λ و قواعد تولید واحد و متغیرهای زائد ندارد.

قواعد بازگشتی چپ را به بازگشتی راست تبدیل کنیم.

G' :

$$S \rightarrow AB \mid aBAcb$$

$$A \rightarrow cd \mid cdA_1$$

$$A_1 \rightarrow a \mid aA_1$$

$$B \rightarrow bA \mid ca$$

جایگزینی قواعد تولید A در قاعده $S \rightarrow AB$

G'' :

$$S \rightarrow cdB \mid cdA_1B \mid aBAcb$$

$$A \rightarrow cd \mid cdA_1$$

$$A_1 \rightarrow a \mid aA_1$$

$$B \rightarrow bA \mid ca$$

نوشتن گرامر معادل در فرم نرمال گریباخ:

در G'' همه رشته های سمت راست با یک الفبا آغاز می شوند. حالا با تعریف متغیرهای کمکی می توانیم گرامر گریباخ را بنویسیم.

G'' :

$$S \rightarrow cX_dB \mid cdA_1B \mid aBAX_cX_bA \rightarrow cd \mid cdA_1$$

$$A_1 \rightarrow a \mid aA_1$$

$$B \rightarrow bA \mid cX_a$$

$$X_a \rightarrow a$$

$$X_b \rightarrow b$$

$$X_c \rightarrow c$$

$$X_d \rightarrow d$$

فصل ششم: ساده سازی گرامر های مستقل از متن - فرم های نرمال
جلسه چهاردهم بخش یک: ویژگی های نرمال چامسکی و الگوریتم *CYK*

ویژگی های گرامر نرمال چامسکی

۱- هر گرامر مستقل از متن G که $\lambda \notin L(G)$ دارای گرامر معادل به فرم نرمال چامسکی است.
(در برخی منابع عضویت λ مشکلی ندارد) همچنین گرامر معادل G در فرم نرمال چامسکی یکتا نیست. مثلا گرامر G دارای گرامرهای معادل G' و G'' در فرم نرمال چامسکی است.

$G:$	$G':$	$G'':$
$S \rightarrow ABa$	$S \rightarrow AY$	$S \rightarrow AY$
$A \rightarrow BBa \mid a$	$A \rightarrow BY \mid a$	$A \rightarrow BZ \mid a$
$B \rightarrow b$	$B \rightarrow b$	$B \rightarrow b$
	$Y \rightarrow BX_a$	$Y \rightarrow BX_a$
	$X_a \rightarrow a$	$Z \rightarrow BX_a$
		$X_a \rightarrow a$

۲- پس از حذف قواعد تولید و قواعد تولید واحد و رسیدن به گرامر غیرانتقباوسی G ، هر قاعده تولید که طول رشته سمت راست آن حداکثر m باشد را میتوان با حداکثر $m - 1$ قاعده به فرم نرمال چامسکی برد. با لحاظ کردن متغیرهایی مانند X_a که نماینده الفباها هستند، یک گرامر نرمال چامسکی معادل با وجود دارد که حداکثر $|P| + (m - 1)|P| + |\Sigma|$ قاعده دارد. m بیشترین طول سمت راست قواعد تولید است. مثلاً یک قاعده که طول سمت راست آن ۴ است به ۳ قاعده به فرم چامسکی تبدیل می‌شود. دو قاعده هم برای الفباها داریم.

$$\begin{aligned} A &\rightarrow EY \\ A \rightarrow EBab &\quad \Rightarrow \quad Y \rightarrow BW \\ W &\rightarrow X_a X_b \\ X_a &\rightarrow a \\ X_b &\rightarrow b \end{aligned}$$

مثال - در این گرامر ۴ قاعده داریم و طول رشته سمت راست حداکثر ۳ است.

$G :$

$$S \rightarrow ABa$$

$$A \rightarrow BBa \mid a$$

$$B \rightarrow b \qquad (m - 1)|P| + |\Sigma| = (2)(4) + 2 = 10.$$

پس دستکم یک گرامر معادل چامسکی با حداکثر ۱۰ قاعده تولید وجود دارد.

۳- در گرامر نرمال چامسکی هر رشته $w \in L(G)$ به طول n دقیقاً با $2n - 1$ مرحله اشتقاق به دست می‌آید. حتی اگر چندین روش تولید داشته باشد همه آنها دارای $2n - 1$ مرحله اشتقاق هستند.

مثلاً گرامر چامسکی مقابل، مبهم است و برای $w = aab$ چند روش تولید وجود دارد اما همه آنها دارای ۵ مرحله اشتقاق هستند:

$G :$

$S \rightarrow AB$ $S \rightarrow AB \rightarrow AAB \rightarrow aAB \rightarrow aaB \rightarrow aab$

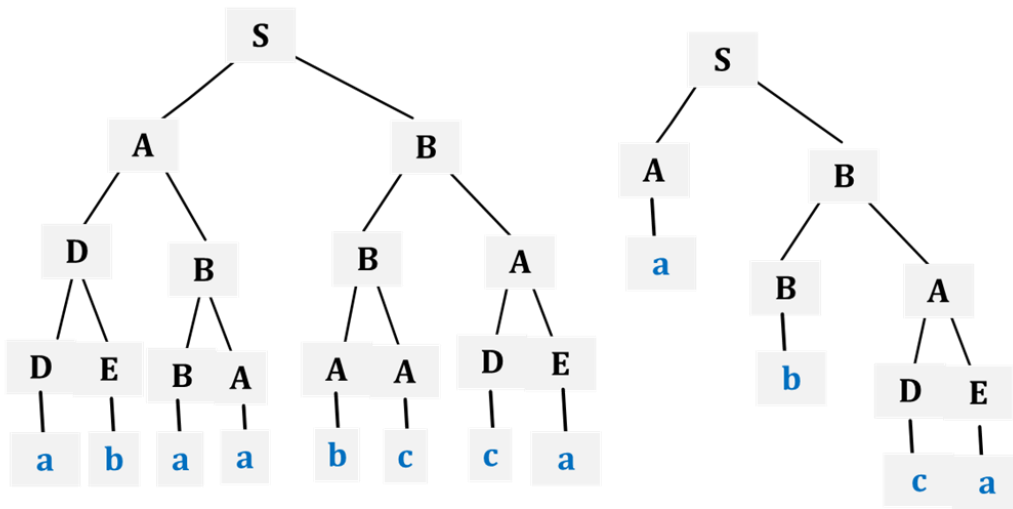
$A \rightarrow AA \mid a$ $S \rightarrow AB \rightarrow AAB \rightarrow AAb \rightarrow Aab \rightarrow aab$

$B \rightarrow AB \mid b$

۴- در گرامر نرمال چامسکی: درخت اشتقاق دودویی است. اگر رشته w تولید شود و درخت اشتقاق آن به عمق d باشد، (ارتفاع درخت d باشد) داریم:

$$d \leq |w| \leq 2^{d-1}$$

در شکل میبینید که اگر ارتفاع درخت اشتقاق ۴ باشد طول w حداقل ۴ و حداکثر ۸ است.



۵- در گرامر نرمال چامسکی، رابطه طول رشته w با عمق درخت اشتقاق چنین است:
(نتیجه مستقیم نکته ۴)

$$1 + \lceil \log_2 |w| \rceil \leq d \leq |w|$$

$$d \leq |w| \leq 2^{d-1}$$

الگوریتم تعیین عضویت CYK :

فرض کنیم $G = (V, T, S, P)$ گرامر نرمال چامسکی باشد. الگوریتم CYK رشته دلخواه $w = a_1 a_2 \dots a_n$ را می‌گیرد و به روش پویا تعیین می‌کند که آیا w عضو $L(G)$ هست یا خیر؟
مرتبه زمانی این الگوریتم $O(n^3)$ است.

مراحل اجرا:

فرض کنیم: $w_{ij} = a_i a_{i+1} \dots a_j \quad 1 \leq i \leq j \leq n$

مجموعه $V[i, j]$ شامل متغیرهایی است که w_{ij} با شروع از آنها قابل تولید است.

$$V[i, j] = \{X \in V \mid X \xrightarrow{*} w_{ij}\}$$

هر $V[i, j]$ یک زیرمساله است. با محاسبه $V[i, i]$ آغاز می‌کنیم. $w_{ii} = a_i$ یک الفبا است. اگر قاعده $A \rightarrow a_i$ موجود باشد، A عضو $V[i, i]$ است. حالا با روش پویای پایین به بالا زیرمساله‌های بزرگتر را محاسبه می‌کنیم:

$$V[i, j] = \bigcup_{k=i}^{j-1} \{A \mid A \rightarrow BC, B \in V[i, k], C \in V[k+1, j]\}$$

هدف به دست آوردن مجموعه $V[1, n]$ است. اگر این مجموعه شامل S بود به این معناست که با شروع از S می‌توان $w = a_1 a_2 \dots a_n$ را تولید کرد پس $w \in L(G)$ در غیر اینصورت $w \notin L(G)$.

$$w \in L(G) \Leftrightarrow S \in V[1, n]$$

تمرین: با اجرای CYK عضویت یا عدم عضویت $abab$ در $L(G)$ را تعیین کنید.

G:

$S \rightarrow AA|EE$

$A \rightarrow BA|a$

$B \rightarrow EB|b$

$E \rightarrow AB|EA|c$

پاسخ:

	1	2	3	4
$w = abab$	a	b	a	b

در اولین تکرار زیرمساله های $V_{i,i}$ را حل کنیم. با توجه به قواعد $A \rightarrow a$ و $B \rightarrow b$ داریم:

$$V_{11} = \{A\}, \quad V_{22} = \{B\}, \quad V_{33} = \{A\}, \quad V_{44} = \{B\}$$

در تکرار بعد زیرمساله های $V_{i,i+1}$ را حل کنیم. برای V_{12} به تولید رشته $w_{12} = ab$ توجه کنیم، فقط یک راه برای تفکیک آن به دو رشته ناتهی وجود دارد.

به همین شکل V_{23} و V_{34} را حساب کرده ایم. $A \in V_{11}$ و $B \in V_{22}$ و فقط متغیر E اشتقاق $E \rightarrow AB$ دارد. پس: $V_{12} = \{E\}$

E	
A	B
1	2
a	b

$$V_{12} = \{E\},$$

A	
B	A
2	3
b	a

$$V_{23} = \{A\},$$

E	
A	B
3	4
a	b

$$V_{34} = \{E\}$$

محاسبه V_{24} و V_{13} :

W_{13} به دو طریق به دو رشته ناتهی تفکیک می شود:

$A \in V_{23}$ و $A \in V_{11}$ همچنین $S \rightarrow AA$ پس $S \in V_{13}$.

$E \in V_{12}$ و $A \in V_{33}$ همچنین $E \rightarrow EA$ پس $E \in V_{13}$.

S			E		
A		A	E		A
1	2	3	1	2	3
a	b	a	a	b	a

$$V_{13} = \{S, E\}$$

W_{24} به دو صورت به دو رشته ناتهی تفکیک می شود:

$E \in V_{34}$ و $B \in V_{22}$ ولی هیچ متغیری به BE مشتق نمی شود.

$B \in V_{44}$ و $A \in V_{23}$ همچنین $E \rightarrow AB$ پس $E \in V_{24}$.

?			E		
B		E	A		B
2	3	4	2	3	4
b	a	b	b	a	b

$$V_{24} = \{E\}$$

محاسبه V_{14} :

W_{14} به سه صورت به دو رشته ناتهی تفکیک می‌شود:

$A \in V_{11}$ و $E \in V_{24}$ ولی هیچ متغیری به AE مشتق نمی‌شود.

$E \in V_{12}$ و $E \in V_{34}$ همچنین $S \rightarrow EE$ پس $S \in V_{14}$.

$E \in V_{13}$ و $B \in V_{44}$ همچنین $B \rightarrow EB$ پس $B \in V_{14}$.

انتخاب دیگر $S \in V_{13}$ و $B \in V_{44}$ است ولی هیچ متغیری به BS مشتق نمی‌شود.

در نهایت: $V_{14} = \{B, S\}$.

?				S				B				?			
A		E		E		E		E		B		S		B	
1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
a	b	a	b	a	b	a	b	a	b	a	b	a	b	a	b

$$V_{14} = \{B, S\}$$

در پایان: می‌بینیم که $S \in V_{14}$ پس $w \in L(G)$.

مثال قبل به طور خلاصه::

G: $w = abab$

$S \rightarrow AA|EE$

$A \rightarrow BA|a$

$B \rightarrow EB|b$

$E \rightarrow AB|EA|c$

1	2	3	4
a	b	a	b

$$V_{11} = \{A\}, \quad V_{22} = \{B\}, \quad V_{33} = \{A\}, \quad V_{44} = \{B\}$$

$$V_{12} = \{E\}, \quad V_{23} = \{A\}, \quad V_{34} = \{E\}$$

$$V_{13} = \{S, E\} \quad V_{24} = \{E\}$$

$$V_{14} = \{B, S\}$$

مرتبه زمانی الگوریتم CYK :

به ازای هر جفت (i, j) که $1 \leq i \leq j \leq n$ یک زیرمساله داریم پس:

$$\text{تعداد زیرمساله ها} = \binom{n}{2} + n = \frac{n(n+1)}{2} = O(n^2)$$

برای حل هر کدام به اجتماع حداکثر n زیرمساله کوچکتر نیاز داریم. پس مرتبه زمانی الگوریتم $O(n^3)$ است.

فصل هفتم: ماشین پشته ای
جلسه چهاردهم بخش دو: تعریف ماشین پشته ای و چند نمونه معروف آن

تعریف ماشین پشته‌ای (PDA) (Pushdown automata)

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

Q مجموعه‌ای ناتهی و متناهی از حالت های داخلی واحد کنترل است.

Σ مجموعه الفبای رشته های ورودی است . متناهی و ناتهی است.

Γ مجموعه الفبا و نمادهای پشته است. متناهی و ناتهی است.

q_0 یکی از اعضای Q که حالت اولیه است.

F زیرمجموعه ای از Q که حالات پایانی (پذیرش) نام دارند.

z علامتی که ابتدا در پشته قرار دارد. خروج آن نشانه خالی شدن پشته است.

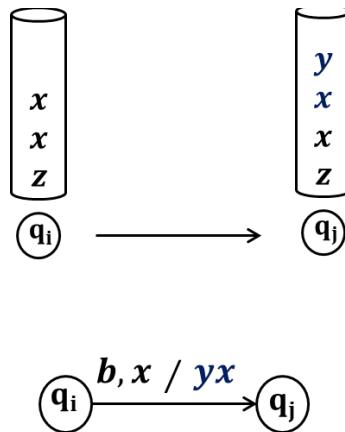
δ : تابع انتقال ماشین پشته‌ای:

$$\delta: Q \times (\Sigma \cup \lambda) \times (\Gamma \cup \lambda) \rightarrow P(Q \times \Gamma^*)$$

دستور انتقال $\delta(q_i, b, x) = (q_j, yx)$ به این معنی است:

اگر ماشین در حالت q_i باشد و از ورودی الفبای b را خوانده شود و از پشته x استخراج

شود، آنگاه ماشین به حالت q_j می‌رود و yx در پشته درج می‌شود (ابتدا x سپس y)

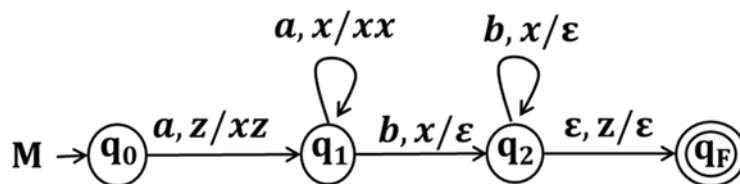


توجه: زبان L مستقل از متن است اگر و تنها اگر ماشین پشته ای پذیرنده داشته باشد.

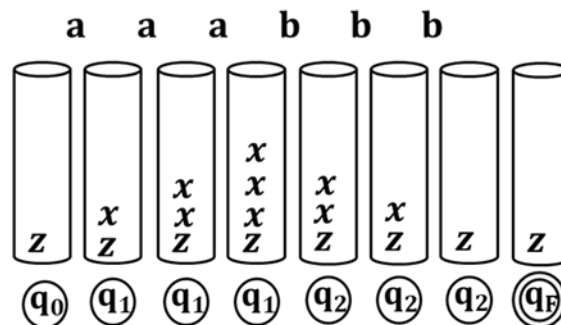
مثال ۱-الف: ماشین پشته ای پذیرنده $L = \{a^n b^n \mid n \geq 1\}$ را طراحی کنیم.

پاسخ:

ایده اصلی آن است که با مشاهده هر a یک علامت مانند x به پشته اضافه شود. سپس با مشاهده هر b یکی از این علامت x حذف شود. اگر هنگام به پایان رسیدن رشته، بالای پشته Z بود نشانه تساوی تعداد a و b است. در ضمن چون ترتیب الفباها مهم است به تغییر حالت با مشاهده اولین b نیاز داریم.

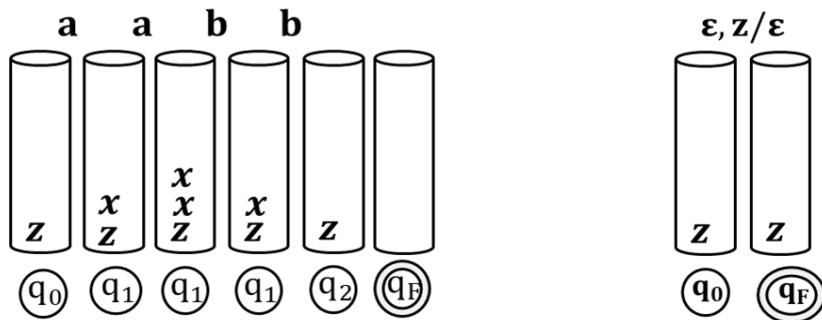
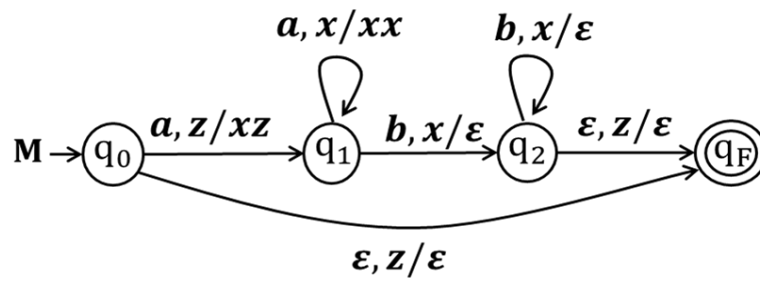


$$L(M) = \{a^n b^n \mid n \geq 1\}$$

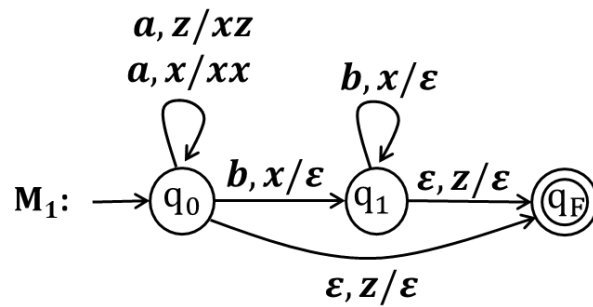


مثال ۱-ب: ماشین پشته ای پذیرنده $L = \{a^n b^n \mid n \geq 0\}$ را طراحی کنیم.

پاسخ: از همان ماشین M در مثال (۱-الف) استفاده کنیم اما با افزودن انتقالی تھی از حالت اولیه به حالت پذیرش.



البته ماشین پشته ای M_1 نیز پذیرنده L است: $L(M_1) = L(M) = L$

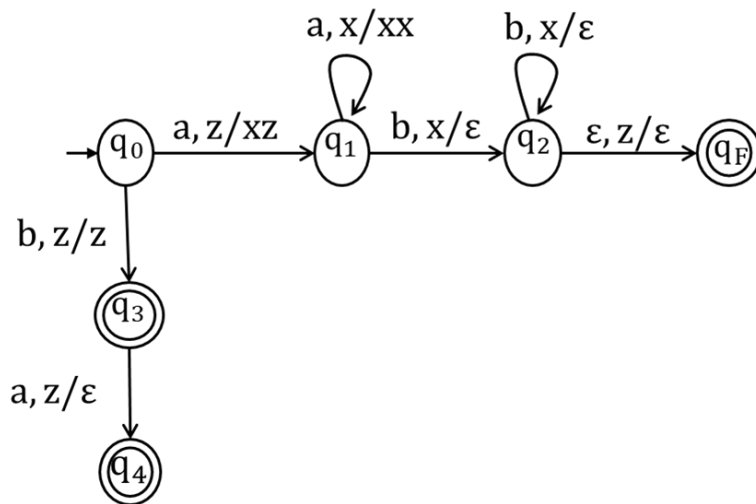


مثال ۱-ج: ماشین پشته ای پذیرنده $L = \{a^n b^n \mid n \geq 1\} \cup \{b, ba\}$ را طراحی کنیم.

پاسخ:

باز هم از ماشین (۱- الف) استفاده کنیم با این تفاوت که با افزودن انتقال ها و حالات پذیرش جدید ، رشته های b و ba نیز پذیرفته شوند.

این مثال نشان می دهد: با افزودن تعداد متناهی رشته به زبان مستقل از متن، باز هم زبانی مستقل از متن خواهیم داشت.



مثال ۲- الف:

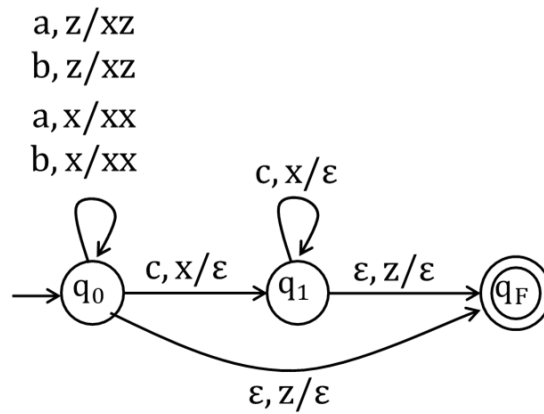
ماشین پشته ای پذیرنده $L = \{wc^k | w \in (a+b)^*, k = |w|\}$ را طراحی کنیم.

پاسخ:

اعضای L به فرم wc^k هستند که $k = |w|$. از آنجا که $w \in (a+b)^*$ پس اندازه w با $n_a(w) + n_b(w)$ برابر است.

$$L = \{wc^k | w \in (a+b)^*, k = n_a(w) + n_b(w)\}$$

پس در q_0 با مشاهده هر a و هر b یک علامت x به پشته اضافه کنیم. سپس با مشاهده c به q_1 تغییر حالت دهیم و به ازای هر c یک x از پشته برداریم. اگر هنگام به پایان رسیدن رشته، z بالای پشته بود به حالت پذیرش برویم. رشته تهی هم به شکل جداگانه پذیرفته شود.

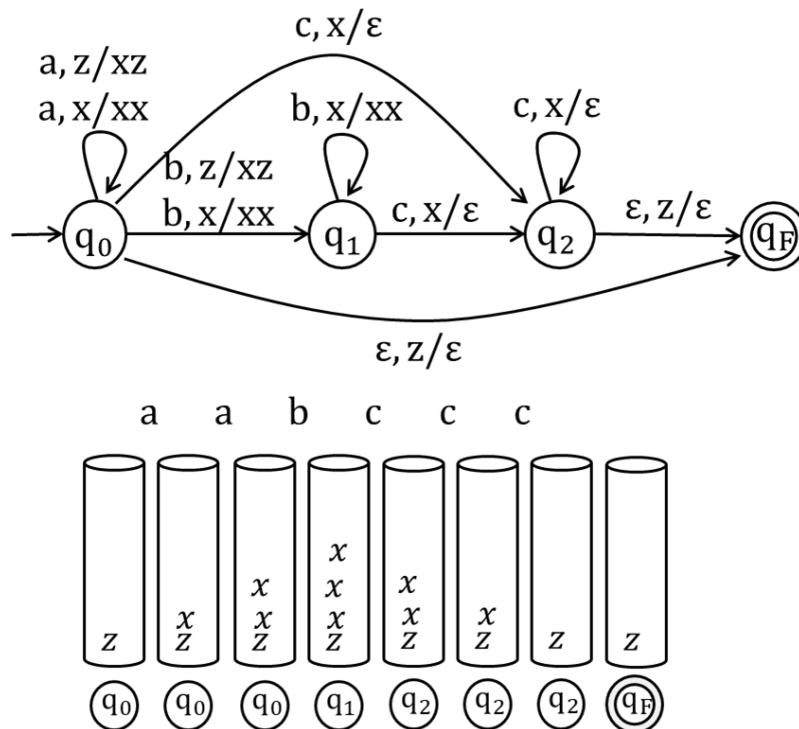


مثال ۲-ب:

ماشین پشته ای پذیرنده $L = \{a^n b^m c^k | k - n = m\}$ را طراحی کنیم.

پاسخ:

در این زبان نیز می‌خواهیم $k = n + m$ باشد پس تا حدودی شبیه مثال قبل است. با این تفاوت که ترتیب a و b نیز مهم است. به سه حالت q_0 ، q_1 و q_2 نیاز داریم که در دوتای اول مصرف a^n و b^m را انجام دهیم و در سومی با مصرف c^k تساوی $k = n + m$ را بررسی کنیم. اگر تساوی برقرار باشد به حالت پذیرش می‌رویم. انتقال‌هایی هم برای حالتی که $n = 0$ یا $m = 0$ باشد در نظر می‌گیریم. رشته تهی هم پذیرفته می‌شود.



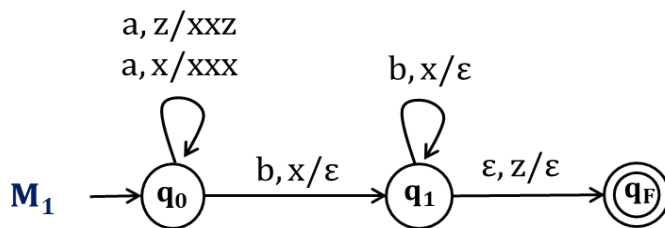
مثال ۳: ماشین پشته ای پذیرنده برای $L = \{a^n b^{2n} | n \geq 1\}$ طراحی کنیم.

پاسخ:

برای این زبان به چند روش می توان ماشین پشته ای طراحی کرد.

در M_1 : به ازای هر a دو علامت x به پشته اضافه می شود سپس با هر b یک x برداشته

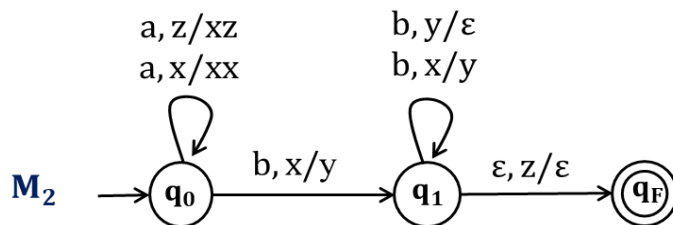
می شود.



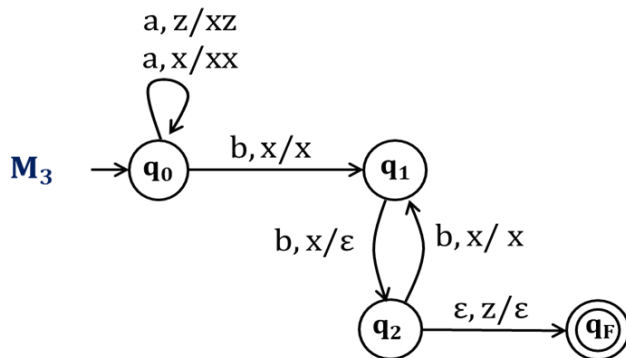
در M_2 : به ازای هر a یک علامت x به پشته اضافه می شود. سپس با هر جفت b یک x

برداشته می شود. به این صورت که با مشاهده اولین b علامت x به تبدیل می شود و با

مشاهده دومین b علامت حذف می شود.



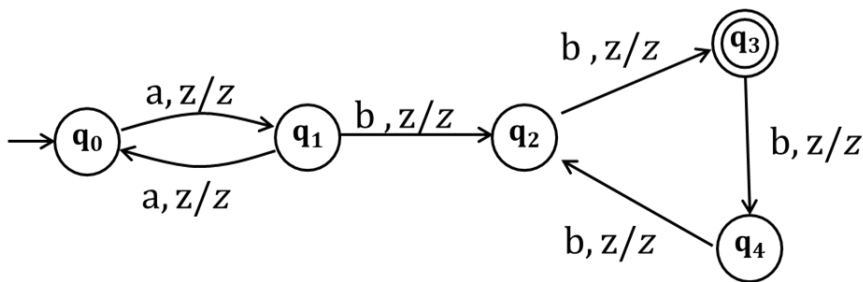
در M_3 : به ازای هر a یک علامت x به پشته اضافه می‌شود. سپس با هر جفت b یک x برداشته می‌شود. به این صورت که اولین b خنثی است و با دومین b علامت x حذف می‌شود.



مثال ۴: هر زبان منظم، مستقل از متن است.

هر زبان منظم دارای ماشین متناهی است. می‌توان فرض کرد ماشین متناهی یک ماشین پشته ای هست که از پشته استفاده نمی‌کند در هر انتقال z را که از پشته خارج شده است به پشته برمی‌گرداند. برای مثال به زبان منظم L توجه کنید. ماشین متناهی پذیرنده آن را به ماشین پشته‌ای معادل تبدیل می‌کنیم:

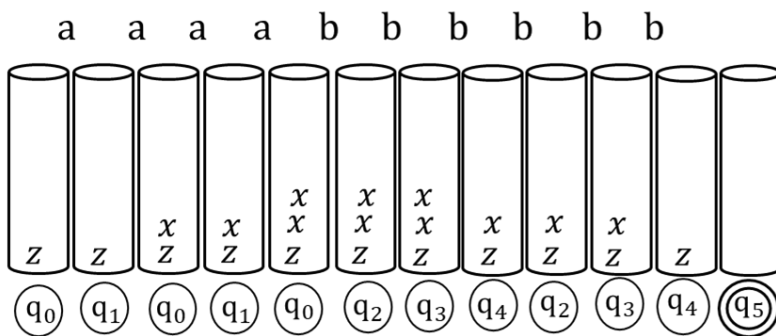
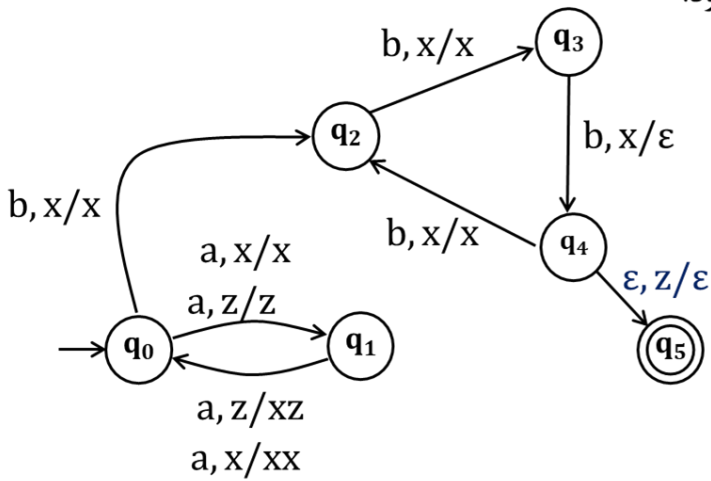
$$L = \{a^{2n+1}b^{3m+1} \mid n, m \geq 0\}$$



مثال ۵: ماشین پشته ای پذیرنده برای $L = \{a^{2n}b^{3n} | n \geq 1\}$ طراحی کنیم.

پاسخ:

زوج بودن تعداد a را با یک دور به طول ۲ کنترل می کنیم. برای تشخیص رابطه بین تعداد a و b ، ابتدا با خواندن هر جفت a یک علامت x به پشته اضافه می شود. سپس به ازای هر سه تا b ، یک x برداشته می شود.



مثال ۶: ماشین پشته ای پذیرنده زبان L را طراحی کنیم.
 $L = \{w \in (a + b)^* | n_a(w) = n_b(w)\}$

پاسخ:

در این رشته‌ها ترتیب a و b مهم نیست پس لازم نیست برای هر کدام یک حالت مجزا داشته باشیم. از همان علامت‌های a و b به عنوان الفبای پشته استفاده کنیم.
با خواندن هر a : یک a به پشته اضافه می‌شود مگر آنکه b بالای پشته باشد، که در این صورت b را از پشته برمی‌داریم.
با خواندن هر b : یک b به پشته اضافه می‌شود مگر آنکه a بالای پشته باشد، در این صورت a را از پشته برمی‌داریم.

$a, z / az$

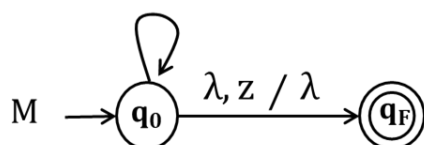
$b, z / bz$

$a, a / aa$

$b, b / bb$

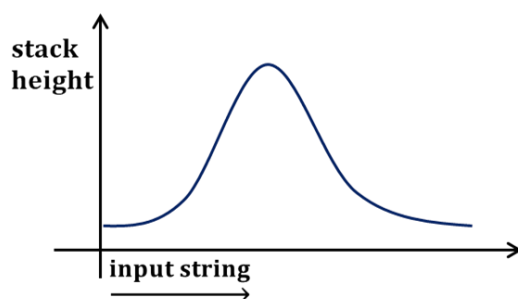
$a, b / \lambda$

$b, a / \lambda$

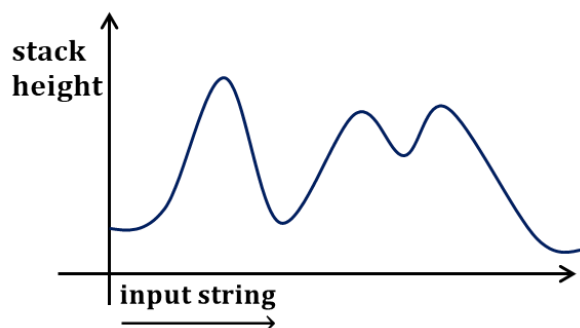


ماشین پشته‌ای یک نوبته

ماشین پشته‌ای پذیرنده یک زبان را یک نوبته گوییم هرگاه دارای این ویژگی باشد: پس از اولین انتقالی که باعث کاهش عناصر پشته شود، تعداد عناصر پشته افزایش نیابد و فقط بتواند کاهش پیدا کند. یعنی تعداد عناصر درج شده در پشته تا یک جا افزایشی و بعد از آن کاهش‌ی باشد.



ماشین پشته‌ای یک نوبته



ماشین پشته‌ای چندنوبته

توجه: زبان L خطی است اگر و تنها اگر ماشین پشته ای یک نوبته داشته باشد.

ماشین های پشته ای مربوط به این زبان ها در مثال های فوق ، یک نوبته هستند:

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

$$L = \{a^n b^m c^k \mid k = n + m\}$$

$$L = \{wc^k \mid w \in (a + b)^*, k = |w|\}$$

اما ماشین پشته ای این زبان یک نوبته نیست:

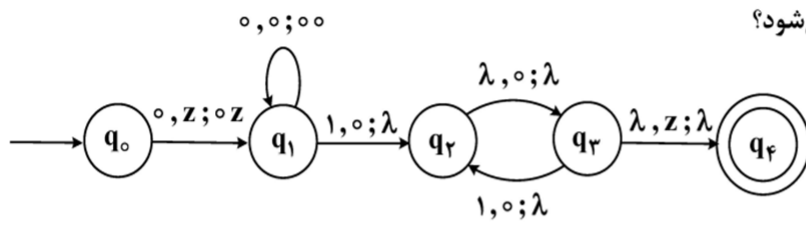
$$L = \{w \in (a + b)^* \mid n_a(w) = n_b(w)\}$$

-15-

فصل هفتم: ماشین پشته ای
جلسه پانزدهم بخش یک: ماشین پشته ای قطعی و غیرقطعی

۸۹- اتوماتای پشته‌ای زیر مفروض است. اگر $\alpha = 000011$ و $\beta = 000101010$ ، کدام یک از این دو رشته، توسط

این اتوماتا پذیرش می‌شود؟



(۱) فقط α

(۲) فقط β

(۳) هم α هم β

(۴) نه α و نه β

علوم کامپیوتر ۱۴۰۳

پاسخ: گزینه ۱

روش اول: با بررسی انتقالها متوجه می‌شویم ابتدا الفبای 0 مصرف می‌شود و با هر 0 ورودی،

یک 0 به پشته اضافه می‌شود. سپس از یک جا به بعد فقط الفبای 1 مصرف می‌شود و به

ازای هر 1 ورودی، دوتا 0 از پشته برداشته می‌شود. بنابراین فقط رشته هایی پذیرفته

می‌شوند که به فرم $0^{2n}1^n$ باشند.

روش دوم: با امتحان کردن رشته ها متوجه می‌شویم α پذیرفته می‌شود اما β در q_2

متوقف می‌شود.

مقدمه:

در مورد ماشین های متناهی دیدیم که توان محاسباتی ماشین های متناهی قطعی و غیرقطعی برابر است:

(مجموعه زبان های دارای dfa) = (مجموعه زبان های دارای nfa) = (مجموعه زبان های منظم)

اما توان محاسباتی ماشین های پشته ای قطعی و غیر قطعی با هم برابر نیست.

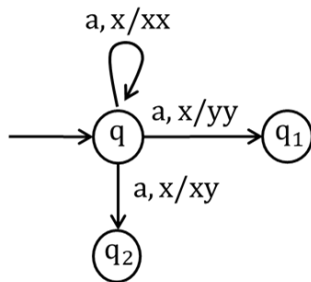
ابتدا ببینیم ماشین پشته ای قطعی چه شرایطی دارد؟

ماشین پشته ای قطعی

در ماشین پشته ای غیر قطعی که همان حالت کلی ماشین پشته ای است تابع انتقال چنین است:

$$\delta: Q \times (\Sigma \cup \lambda) \times (\Gamma \cup \lambda) \rightarrow P(Q \times \Gamma^*)$$

حاصل $\delta(q, a, x)$ می تواند زیرمجموعه دلخواهی از $Q \times \Gamma^*$ باشد مثلاً:



اما ماشین پشته ای قطعی دو محدودیت دارد:

الف: $\delta(q, a, x)$ فقط یک عضو از $Q \times \Gamma^*$ می تواند باشد. از حالت q با الفبای a و نماد x بالای پشته فقط یک انتقال داریم.

$$\delta: Q \times (\Sigma \cup \lambda) \times (\Gamma \cup \lambda) \rightarrow Q \times \Gamma^*$$

ب: اگر انتقال λ برای $\delta(q, \lambda, x)$ داشتیم دیگر نباید از همین حالت q و همین نماد x بالای پشته، انتقال با الفبا داشته باشیم.

ماشین پشته ای قطعی و غیرقطعی را به اختصار **dpda** و **npda** می نامیم.

همه زبان های مستقل از متن دارای ماشین پشته ای (غیرقطعی) هستند.
 هر زبان مستقل از متن که ماشین پشته ای قطعی نیز داشته باشد یک زبان مستقل از متن قطعی نام دارد. زبان های مستقل از متن قطعی زیرمجموعه ای محض از زبان های مستقل از متن هستند.

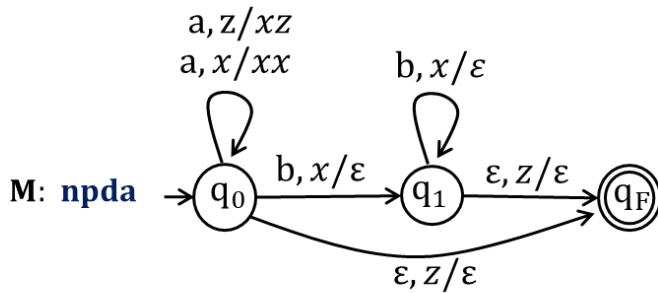
زبان های مستقل از متن **npda**



توان محاسباتی ماشین های پشته ای (غیرقطعی) مجموعه همه مستقل از متن ها است.
 توان محاسباتی ماشین های پشته ای قطعی مجموعه مستقل از متن های قطعی است.
 هر ماشین پشته ای قطعی، دارای ماشین معادل غیرقطعی است.
 هر ماشین پشته ای غیرقطعی لزوماً ماشین معادل قطعی ندارد.

مثال ۱: در جلسه قبل ماشین پشته ای پذیرنده $L = \{a^n b^n \mid n \geq 0\}$ را دیدیم. آیا این ماشین قطعی است؟ آیا معادل قطعی دارد؟

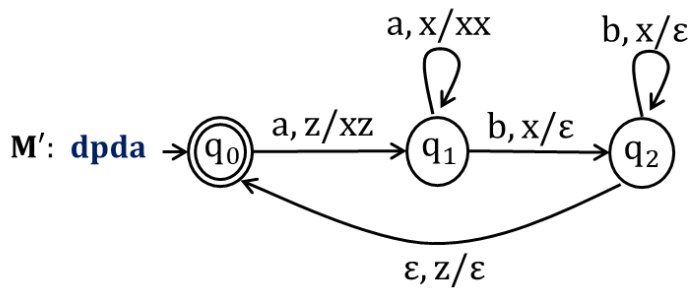
پاسخ: ماشینی که برای این زبان یافتیم غیرقطعی است چون از حالت q_0 با علامت Z بالای پشته هم انتقال تهی دارد و هم انتقال با الفبای a دارد.



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

در ماشین قطعی برای پذیرفته شدن رشته تهی ناچاریم q_0 را حالت پذیرش فرض کنیم. سایر

انتقال ها به این صورت انجام می شوند:



در مورد مثال ۱:

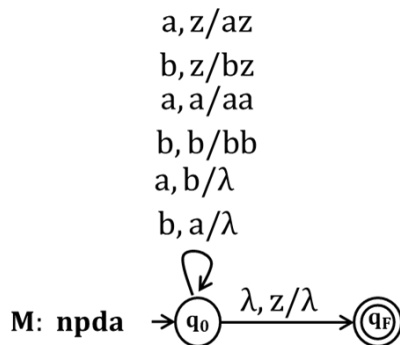
نکته اول: متوجه شدیم که زبان $L = \{a^n b^n \mid n \geq 0\}$ ماشین پشته ای قطعی نیز دارد پس یک زبان مستقل از متن قطعی است. در واقع وقتی به نحوه استفاده از پشته برای رشته های $a^n b^n$ دقت کنیم میبینیم که در هر گام بدون ابهام و با قطعیت میتوانیم نوع انتقال را تعیین کنیم. می دانیم که وقتی الفبای a به پایان رسید و اولین الفبای a دیدیم باید از آن پس فقط الفبای b ببینیم و شروع به خالی کردن پشته کنیم. این ویژگی زبان های قطعی است که ما در نحوه تغییر حالات ماشین دچار ابهام (و دوراهی) نمی شویم.

نکته دوم: در ماشین قطعی برای انتقال از q_2 به حالت پذیرش q_0 از $\epsilon, Z/\epsilon$ استفاده کرده ایم یعنی Z را به پشته برگردانده ایم تا پشته خالی شود. این انتخاب دقیق و مهم است زیرا اگر از $\epsilon, Z/Z$ استفاده می کردیم وضعیت پشته به حالت اولیه برمی گشت و حالا یک بار دیگر انتقال ها می توانستند تکرار شوند به عبارتی رشته های به فرم $a^n b^n$ و $a^m b^m$ و $a^n b^n a^m b^m a^k b^k$ و ... نیز پذیرفته می شدند و به جای زبان L ، زبان L^* پذیرفته می شد.

مثال ۲: (بررسی مثال ۶ جلسه قبل)

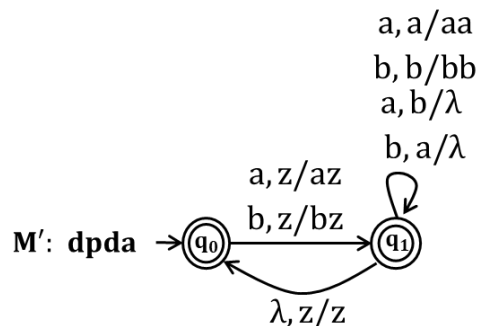
در جلسه قبل ماشین پشته ای پذیرنده $L = \{w \in (a+b)^* \mid n_a(w) = n_b(w)\}$ را یافتیم. آیا این ماشین قطعی است؟ آیا معادل قطعی دارد؟

پاسخ: ماشینی که برای این زبان یافتیم غیرقطعی است چون از حالت q_0 با علامت Z بالای پشته، هم انتقال تهی وجود دارد و هم انتقال با الفباهای a و b وجود دارد.



در ماشین قطعی برای پذیرفته شدن رشته تهی ناچاریم q_0 را حالت پذیرش فرض کنیم. سایر انتقال ها به این صورت انجام می شوند:

پس این زبان نیز قطعی است. در ماشین قطعی پذیرنده این زبان برای انتقال از q_1 به حالت پذیرش q_0 از $\epsilon, Z/Z$ استفاده کرده ایم یعنی Z را به پشته برگردانده ایم تا انتقال ها بتوانند تکرار شوند. مثلاً این زبان هم رشته a^2b^2 دارد هم رشته a^2b^2ab دارد یعنی برخی رشته هایش پیشوند رشته دیگر هستند. در این مورد در بخش دوم کلاس بحث کرده ایم.



مثال های بیشتر:

توضیحات	زبان
مستقل از متن نیست چون ماشین پشته ای ندارد. شرط تساوی دو بخش از رشته در حالت کلی مستقل از متن نمی شود مگر آنکه تعداد الگوهایش متناهی شود. مثلا اگر شرط $ w = 2$ را داشت مستقل از متن بود.	$L = \{wdw \mid w \in (a+b)^*\}$
مستقل از متن است. پشته می تواند w را در خودش ذخیره کند سپس الفبای w^R را با الفبای بالای پشته مقایسه کند. در ضمن چون الفبای d در w ظاهر نمی شود با قطعیت می دانیم که تا کدام الفبا را در پشته ذخیره کنیم و چه زمانی تغییر حالت بدهیم. یعنی مستقل از متن قطعی است.	$L = \{w d w^R \mid w \in (a+b)^*\}$
مستقل از متن است اما قطعی نیست. (دیگر نشانه ای مانند الفبای d بین دو بخش وجود ندارد).	$L = \{w w^R \mid w \in (a+b)^*\}$
هر زبان منظم، مستقل از متن و قطعی است.	$L = (\text{یک زبان منظم})$

توضیحات	زبان
مستقل از متن و قطعی است. (در همین بخش بررسی شد)	$L = \{a^n b^n \mid n \geq 0\}$
مستقل از متن است. اما غیرقطعی است. (بعد از ذخیره تعداد الفبای a در پشته یک دوراهی داریم: با هر b یک علامت از پشته کم کنیم یا با هر جفت b یک علامت از پشته کم کنیم.)	$L = \{a^n b^n \mid n \geq 0\} \cup \{a^n b^{2n} \mid n \geq 0\}$
مستقل از متن و قطعی است. (در همین بخش بررسی شد)	$L = \{w \in (a + b)^* \mid n_a(w) = n_b(w)\}$

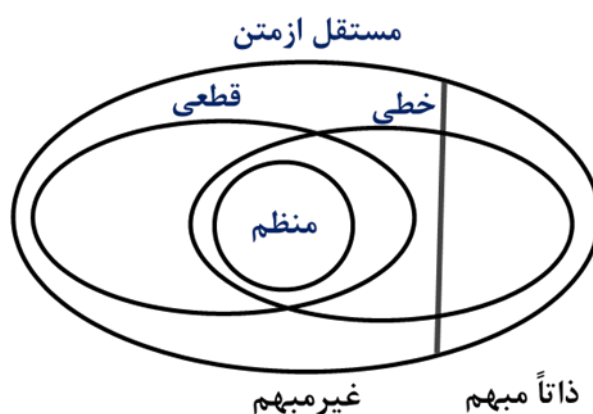
ارتباط زبان های مستقل از متن غیرمبهم با زبان های مستقل از متن قطعی:

- ۱- هر زبان منظم، مستقل از متن قطعی است.
- ۲- هر زبان مستقل از متن قطعی، لزوماً منظم نیست. مثلاً $\{a^n b^n \mid n \geq 0\}$.
- ۳- هر زبان مستقل از متن قطعی، حتماً غیرمبهم است.
- ۴- هر زبان غیرمبهم، لزوماً قطعی نیست. مثلاً $L = \{w w^R \mid w \in (a + b)^*\}$.
این زبان گرامر غیرمبهم $S \rightarrow aSb \mid bSa \mid \lambda$ دارد. اما ماشین پشته ای قطعی ندارد.



مستقل از متن $\longrightarrow$ غیرمبهم $\longrightarrow$ قطعی $\longrightarrow$ منظم

مستقل از متن $\longrightarrow$ خطی $\longrightarrow$ منظم



فصل هفتم: ماشین پشته ای

جلسه پانزدهم بخش دو:
ماشین پشته ای قطعی و غیرقطعی (پذیرش توسط پشته خالی)

پذیرش در ماشین پشته‌ای: ماشین پشته‌ای غیرقطعی دو صورت می‌تواند رشته بپذیرد:

پذیرش توسط حالات نهایی: رشته پذیرفته می‌شود اگر و تنها اگر با مصرف آن
 به حالت پذیرش (نهایی) برسیم. } **npda**
 پذیرش توسط پشته خالی: رشته پذیرفته می‌شود اگر و تنها اگر با مصرف آن
 به پشته خالی برسیم.

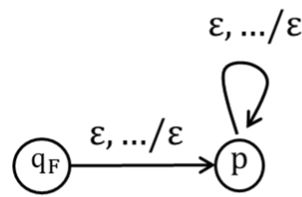
البته هر npda با پذیرش توسط حالات نهایی قابل تبدیل به npda با پذیرش توسط پشته خالی است و بالعکس. پس هر زبان مستقل از متن دارای ماشین پشته‌ای غیرقطعی از هر دو نوع هست.

در ماشین پشته‌ای **قطعی** نیز پذیرش رشته به دو صورت ممکن است تعریف شود:

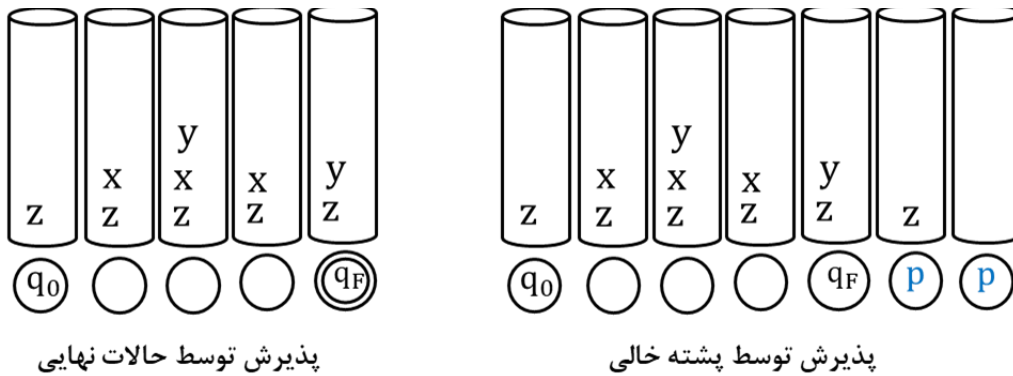
پذیرش توسط حالات نهایی: رشته پذیرفته می‌شود اگر و تنها اگر با مصرف آن
 به حالت پذیرش (نهایی) برسیم. } **dpda**
 پذیرش توسط پشته خالی: رشته پذیرفته می‌شود اگر و تنها اگر با مصرف آن
 به پشته خالی برسیم.

هر زبان مستقل از متن قطعی دارای dpda با پذیرش توسط حالات نهایی است. اما فقط برخیز آنها دارای dpda با پذیرش توسط پشته خالی هستند.

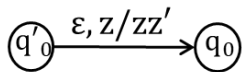
هر ماشین npda با پذیرش توسط حالات نهایی قابل تبدیل به npda با پذیرش توسط پشته خالی است: فرض کنید q_F حالت پذیرش باشد. اگر با مصرف به برسیم با تعریف انتقالهایی به صورت:



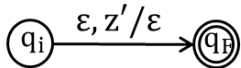
یعنی با مصرف ϵ ، بالای پشته هرچه باشد آن را برداشته و به جایش چیزی درج نمی‌کنیم. به این ترتیب در حالت p در نهایت پشته خالی می‌شود.



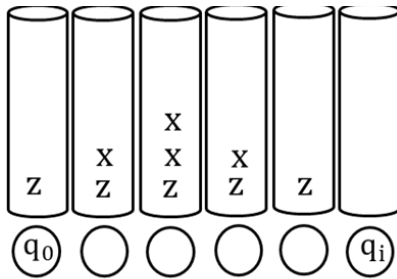
هر ماشین npda با پذیرش توسط پشته خالی قابل تبدیل به npda با پذیرش توسط حالات نهایی است:
فرض کنید در ماشین پشته‌ای، هرگاه پشته خالی شود، در هر حالتی که باشیم، رشته پذیرفته می‌شود.
حالا به همین ماشین حالت اولیه q'_0 با انتقال زیر را اضافه کنیم:



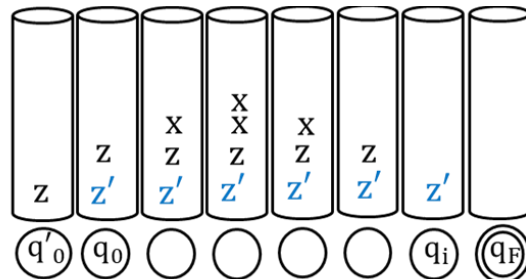
حالا در حالتی که قبلا پشته خالی به دست می‌آمد، Z' بالای پشته باقی می‌ماند. پس می‌توانیم انتقال
هایی به صورت



را به ماشین اضافه کنیم تا به جای پشته خالی، رسیدن به حالت پذیرش q_F شرط پذیرش باشد.

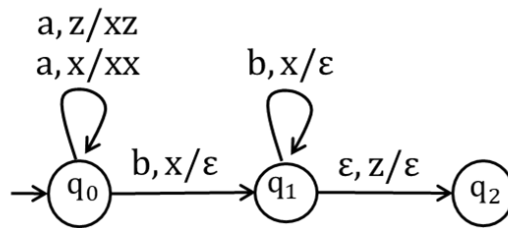


پذیرش توسط پشته خالی



پذیرش توسط حالات نهایی

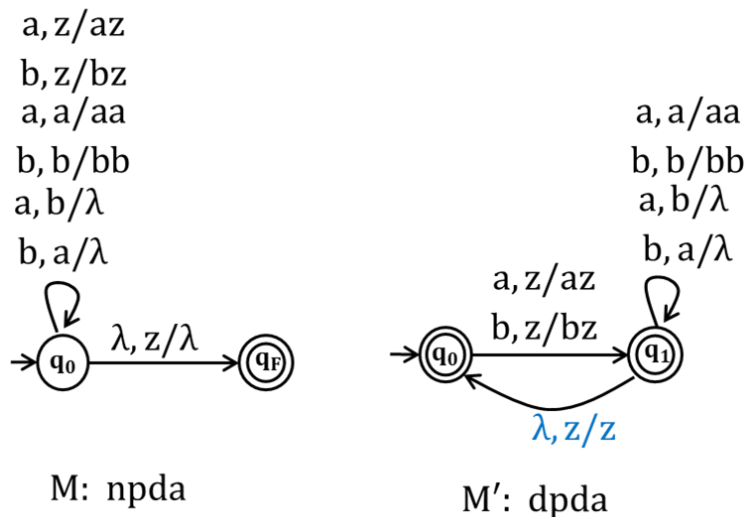
مثال: قبلا دیده ایم که زبان $L = \{a^n b^n \mid n \geq 1\}$ قطعی است یعنی دارای پذیرنده های npda و dpda با پذیرش حالت نهایی است. می توانیم مطابق شکل برای این زبان یک ماشین پشته ای قطعی (و غیرقطعی) پیدا کنیم که به جای حالت پذیرش، خالی شدن پشته به معنای پذیرفته شدن رشته باشد.



M: dpda

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

مثال: قبلا دیده ایم که زبان $L = \{w \in (a + b)^* | n_a(w) = n_b(w)\}$ قطعی است یعنی دارای پذیرنده های npda و dpda با پذیرش حالت نهایی است. در ماشین غیرقطعی همزمان با ورود به پذیرش، پشته هم خالی می شود. اما در ماشین قطعی، ناچاریم از خالی شدن پشته جلوگیری کنیم زیرا برخی رشته ها مانند a^2b^2a زیر رشته a^2b^2 نیز پذیرفته می شود پس اگر با مصرف آن پشته خالی شود نمی توانیم به مصرف سایر الفبا ادامه دهیم. پس این زبان ماشین قطعی دارد اما ماشین قطعی با شرط پذیرش پشته خالی ندارد.

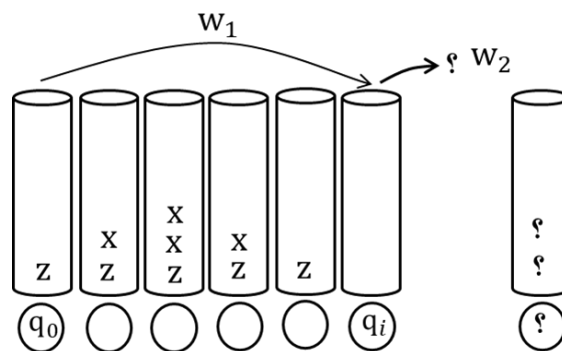


dpda با پذیرش توسط پشته خالی:

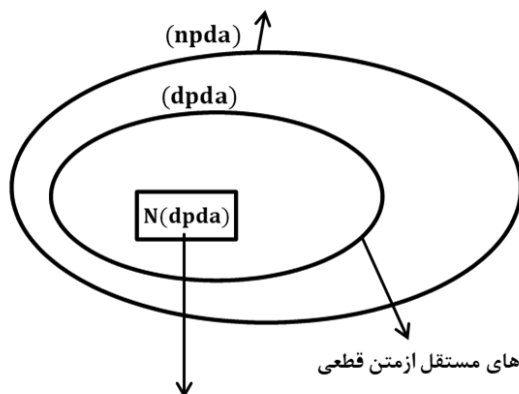
فقط زبان های مستقل از متن قطعی که در آنها هیچ رشته ای پیشوند رشته دیگر نباشد این نوع ماشین پذیرنده را دارند.

اصطلاحاً می‌گوییم این زبان های قطعی، شرط پیشوندی دارند. این مجموعه را با علامت $N(dpda)$ نشان می‌دهیم. مثلاً در $\{a^n b^n \mid n \geq 1\}$ هیچ رشته ای پیشوند رشته دیگر نیست. پس شرط پیشوندی دارد و ماشین قطعی با پذیرش پشته خالی هم دارد. اما زبان $\{w \in (a+b)^* \mid n_a(w) = n_b(w)\}$ شرط پیشوندی ندارد. در این زبان برخی رشته ها پیشوند رشته دیگر هستند. مثلاً $w_1 = a^2 b^2$ و $w_1 w_2 = a^2 b^2 a b$ هر دو عضو زبان هستند.

با فرض خلف، اگر ماشین پشته‌ای قطعی با پذیرش پشته خالی داشته باشد باید $a^2 b^2 a b$ را بپذیرد اما پس از مصرف $a^2 b^2$ پشته خالی می‌شود و دیگر هیچ انتقالی قابل انجام نیست. این تناقض است. پس چنین زبانی ماشین قطعی با پذیرش پشته خالی ندارد.



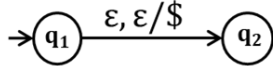
توان محاسباتی **npda** با پذیرش توسط حالات نهایی = توان محاسباتی **npda** با پذیرش توسط پشته خالی = زبان‌های مستقل از متن



توان محاسباتی **dpda** با پذیرش توسط حالات نهایی = زبان‌های مستقل از متن قطعی

توان محاسباتی **dpda** با پذیرش توسط پشته خالی = زبان‌های مستقل از متن قطعی پیشوندی

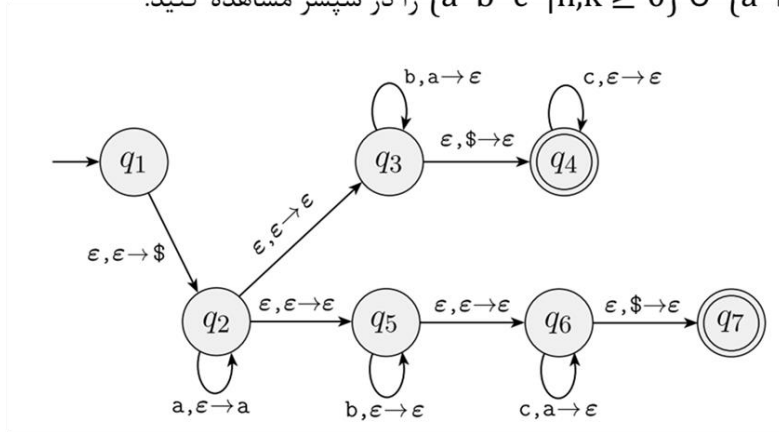
نمایش دیگری از ماشین پشته ای در منبع sipser:



در این رفرنس فرض می‌کند ابتدا پشته خالی است و با انتقال

نماد $\$$ را در پشته درج می‌کند. (از این پس علامت $\$$ نقش همان Z را دارد.) همچنین در تابع انتقال می‌توانیم ϵ را از پشته استخراج کنیم. برای مثال در رفرنس لینز اگر الفبای ورودی a باشد، بالای پشته a باشد و بخواهیم یک a دیگر هم اضافه شود انتقال را به شکل $a, a/aa$ می‌نویسیم اما در سیپسر به شکل $a, \epsilon/a$ نوشته می‌شود. ماشین پشته ای پذیرنده زبان

$\{a^n b^n c^k | n, k \geq 0\} \cup \{a^n b^k c^n | n, k \geq 0\}$ را در سیپسر مشاهده کنید:



در q_4 رشته های $a^n b^n c^k$ پذیرفته می‌شوند.

در q_7 رشته های $a^n b^k c^n$ پذیرفته می‌شوند.

-16-

فصل هشتم: ویژگیهای بستاری زبان‌های مستقل از متن
جلسه شانزدهم بخش یک:
بررسی بسته بودن زبان‌های مستقل از متن نسبت به عملگرهای زبانی

یادآوری: اگر L_1 و L_2 منظم باشند آنگاه این زبان‌ها منظم هستند:

$$L_1 \cup L_2, L_1 \cap L_2, L_1 - L_2, \overline{L_1} = L_1^c, L_1 L_2, L_1^n, L_1^*, L_1^R, h(L_1), h^{-1}(L_1)$$

قضیه: اگر L_1 و L_2 مستقل از متن باشند آنگاه این زبان‌ها مستقل از متن هستند:

$$L_1 \cup L_2, L_1 L_2, L_1^n, L_1^*, L_1^R, h(L_1), h^{-1}(L_1)$$

اما زبان‌های مستقل از متن نسبت به این عملگرها بسته نیستند:

$$L_1 \cap L_2, L_1 - L_2, \overline{L_1} = L_1^c$$

ابتدا با ذکر چند مثال بسته نبودن زبان‌های مستقل از متن نسبت به اشتراک، مکمل، و تفاضل را نشان می‌دهیم:

مثال: L_1 و L_2 مستقل از متن هستند اما اشتراک آنها مستقل از متن نیست:

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

$$L_2 = \{a^k b^n c^n \mid n, k \geq 0\}$$

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

توجه: در زبان های مستقل از متن با استفاده از بسته نبودن نسبت به اشتراک می توان بسته نبودن نسبت به مکمل را ثابت کرد.

بسته نبودن نسبت به اشتراک $\leftarrow$ بسته نبودن نسبت به مکمل

اثبات به این صورت است:

برهان خلف. فرض کنیم زبان های مستقل از متن نسبت به مکمل بسته باشند. در این صورت

برای زبان های L_1 و L_2 مثال فوق مکمل آنها L_1^c و L_2^c نیز مستقل از متن هستند پس

$L_1^c \cup L_2^c$ مستقل از متن است در نتیجه مکمل آن هم مستقل از متن است. اما طبق

قانون دمورگان، مکمل این زبان $L_1 \cap L_2$ است:

$$(L_1^c \cup L_2^c)^c = L_1 \cap L_2$$

پس $L_1 \cap L_2$ مستقل از متن است. (که این تناقض است).

پس مکمل زبان مستقل از متن، لزوماً مستقل از متن نیست.

مثال معروف برای بسته نبودن نسبت به مکمل:

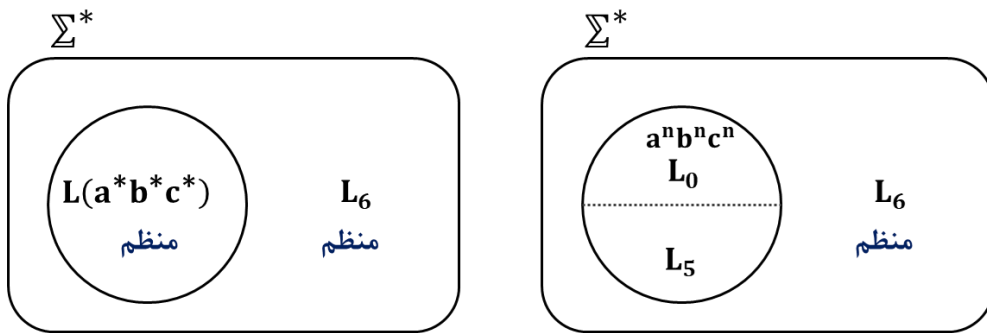
می‌خواهیم یک زبان مستقل از متن پیدا کنیم که مکمل آن، مستقل از متن نیست. با این توضیحات همراه شوید:

$L(a^*b^*c^*)$ منظم است. پس مکمل آن نیز منظم است (L_6).

$$L(a^*b^*c^*) = \{a^i b^j c^k \mid i, j, k \geq 0\}$$

$$L_6 = \overline{L(a^*b^*c^*)}$$

هر زبان منظم، مستقل از متن هست. پس L_6 مستقل از متن است.



$L = L(a^*b^*c^*)$ نیز به دو بخش قابل تفکیک است:

$$L_0 = \{a^i b^j c^k \mid i = j = k\} = \{a^n b^n c^n \mid n \geq 0\}$$

$$L_5 = \{a^i b^j c^k \mid j \neq i \text{ یا } j \neq k\}$$

فرض کنیم:

$$L_1 = \{a^i b^j c^k \mid i > j\}$$

L_1 مستقل از متن است زیرا گرامر مستقل از متن دارد:

G_1 :

$$S \rightarrow S_1 c \mid X$$

$$X \rightarrow aXb \mid A$$

$$A \rightarrow aA \mid a$$

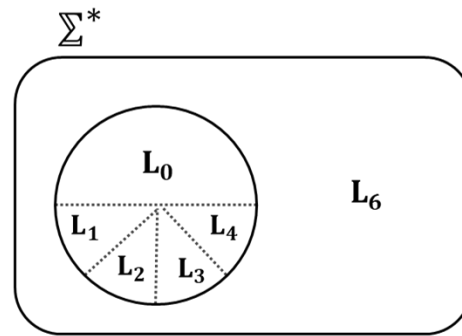
به دلیل مشابه زبان های زیر همگی مستقل از متن هستند:

$$L_1 = \{a^i b^j c^k \mid i > j\}$$

$$L_2 = \{a^i b^j c^k \mid i < j\}$$

$$L_3 = \{a^i b^j c^k \mid j > k\}$$

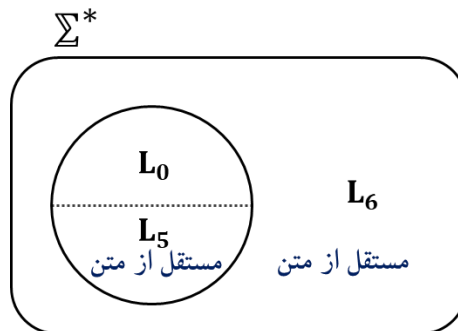
$$L_4 = \{a^i b^j c^k \mid j < k\}$$



اجتماع این ۴ زبان برابر با L_5 است:

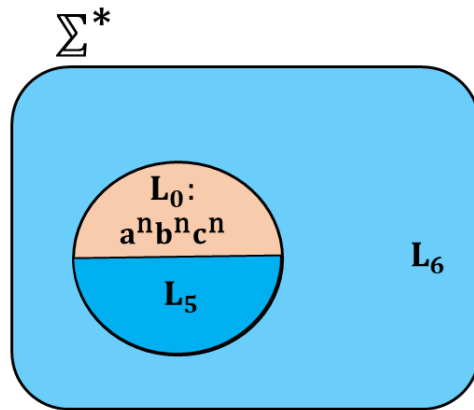
$$L_5 = \{a^i b^j c^k \mid j \neq i \text{ یا } j \neq k\}$$

پس L_5 مستقل از متن است.



نتیجه: اگر فرض کنیم $L = L_5 \cup L_6$ آنگاه L مستقل از متن است

اما مکمل آن $\overline{L} = L_0$ مستقل از متن نیست.



$$L_0 = \{a^n b^n c^n \mid n \geq 0\}$$

$$L_5 = \{a^i b^j c^k \mid j \neq i \text{ یا } j \neq k\}$$

$$L_6 = \overline{L(a^* b^* c^*)}$$

خلاصه:

زبان $\overline{\{a^n b^n c^n \mid n \geq 0\}}$ مستقل از متن است،
اما $\{a^n b^n c^n \mid n \geq 0\}$ مستقل از متن نیست.

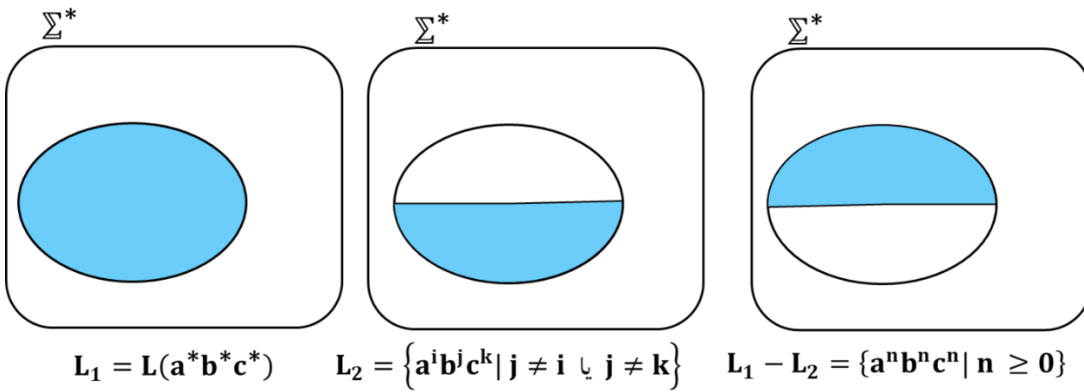
مثال برای بسته نبودن نسبت به تفاضل:

در این شکل L_1 و L_2 مستقل از متن هستند اما تفاضل آنها مستقل از متن نیست.

$$L_1 = L(a^*b^*c^*) = \{a^ib^jc^k \mid i, j, k \geq 0\}$$

$$L_2 = \{a^ib^jc^k \mid j \neq i \text{ یا } j \neq k\}$$

$$L = L_1 - L_2 = \{a^n b^n c^n \mid n \geq 0\}$$



بسته بودن زبان های مستقل از متن نسبت به اجتماع، الحاق، بستار، وارون، همریختی:

گرامر های مستقل از متن G_1 و G_2 را داریم.

G_1 :

$S \rightarrow aSb \mid bAA \mid \lambda$

$A \rightarrow bA \mid ba$

G_2 :

$S \rightarrow bA$

$A \rightarrow aS \mid a$

گرامر مستقل از متن برای این زبان ها ارائه دهید:

الف: $L(G_1) \cup L(G_2)$

ب: $L(G_1)L(G_2)$

ج: $L(G_1)^*$

د: $L(G_1)^R$

ه: $h(L(G_1))$ (h همریختی است)

پاسخ:

ابتدا متغیرهای آغازین را S_1 و S_2 بنامیم و متغیرهای مشترک مانند A را در گرامر G_2 عوض کنیم.

G_1 :

$$S \rightarrow aSb \mid bAA \mid \lambda$$

$$A \rightarrow bA \mid ba$$



G_1 :

$$S_1 \rightarrow aS_1b \mid bAA \mid \lambda$$

$$A \rightarrow bA \mid ba$$

G_2 :

$$S \rightarrow bA$$

$$A \rightarrow aS \mid a$$



G_2 :

$$S_2 \rightarrow bB$$

$$B \rightarrow aS_2 \mid a$$

حالا گرامرهای تولید کننده اجتماع، الحاق، و بستار را تعریف می‌کنیم:

الف: $L(G_1) \cup L(G_2)$

فرض کنیم گرامر G همه قواعد تولید گرامرهای G_1 و G_2 را با متغیر آغاز S و

قواعد $S \rightarrow S_1 | S_2$ دارد. مجموعه قواعد تولید G چنین است:

$$P = P_1 \cup P_2 \cup \{S \rightarrow S_1 | S_2\}$$

این گرامر مستقل از متن زبان $L(G_1) \cup L(G_2)$ را تولید می‌کند.

G:

$$L(G) = L(G_1) \cup L(G_2)$$

$$S \rightarrow S_1 | S_2$$

$$P_1 \left\{ \begin{array}{l} S_1 \rightarrow aS_1b \mid bAA \mid \lambda \\ A \rightarrow bA \mid ba \end{array} \right.$$

$$P_2 \left\{ \begin{array}{l} S_2 \rightarrow bB \\ B \rightarrow aS_2 \mid a \end{array} \right.$$

ب: $L(G_1)L(G_2)$

فرض کنیم گرامر G همه قواعد تولید گرامرهای G_1 و G_2 را با متغیر آغاز S و قواعد $S \rightarrow S_1S_2$ دارد. مجموعه قواعد تولید G چنین است:

$$P = P_1 \cup P_2 \cup \{S \rightarrow S_1S_2\}$$

این گرامر مستقل از متن، زبان $L(G_1)L(G_2)$ را تولید می‌کند.

G :

$$L(G) = L(G_1)L(G_2)$$

$$S \rightarrow S_1S_2$$

$$P_1 \left\{ \begin{array}{l} S_1 \rightarrow aS_1b \mid bAA \mid \lambda \\ A \rightarrow bA \mid ba \end{array} \right.$$

$$P_2 \left\{ \begin{array}{l} S_2 \rightarrow bB \\ B \rightarrow aS_2 \mid a \end{array} \right.$$

ج: $L(G_1)^*$

فرض کنیم گرامر G همه قواعد تولید گرامر G_1 را به همراه متغیر آغاز S و قواعد تولید $S \rightarrow SS \mid S_1 \mid \lambda$ دارد. این گرامر $L(G_1)^*$ را تولید می‌کند. البته قواعد $S \rightarrow SS_1 \mid \lambda$ نیز قابل استفاده اند. پس به دو شکل گرامر مستقل از متنی برای $L(G_1)^*$ به دست می‌آید.

$G :$ $S \rightarrow SS \mid S_1 \mid \lambda$ $P_1 \left\{ \begin{array}{l} S_1 \rightarrow aS_1b \mid bAA \mid \lambda \\ A \rightarrow bA \mid \lambda \end{array} \right.$	$L(G) = L(G_1)^*$
$G :$ $S \rightarrow S_1 S \mid \lambda$ $P_1 \left\{ \begin{array}{l} S_1 \rightarrow aS_1b \mid bAA \mid \lambda \\ A \rightarrow bA \mid \lambda \end{array} \right.$	$L(G) = L(G_1)^*$

د: $L(G_1)^R$

اگر G_1 گرامر مستقل از متن تولید کننده L_1 باشد، با معکوس کردن تمام عبارات سمت راست قواعد تولید، گرامر مستقل از متن G_R به دست می‌آید که تولید کننده L_1^R است. پس زبان های مستقل از متن نسبت به وارون ، بسته است. مثلاً:

$$G_1: \quad L_1 = L(G_1) = \{a^n c d b^n \mid n \geq 0\}$$

$$S_1 \rightarrow a S_1 b \mid cd$$

$$G_R: \quad L(G_R) = \{b^n d c a^n \mid n \geq 0\} = L_1^R$$

$$S_1 \rightarrow b S_1 a \mid dc$$

ه: زبان های مستقل از متن تحت همریختی بسته هستند.

اگر $L_1 = L(G_1)$ مستقل از متن باشد، و h یک همریختی باشد، $h(L_1)$ مستقل از متن است. اگر به جای هر الفبای a در گرامر G_1 مقدار $h(a)$ را قرار دهیم گرامری به دست می آید که مولد $h(L_1)$ است.

مثال:

G_1 :

$S_1 \rightarrow aS_1b \mid cd$

$$L(G_1) = L_1 = \{a^n c d b^n \mid n \geq 0\}$$

$$h(a) = c,$$

$$h(b) = c,$$

$$h(c) = b,$$

$$h(d) = b.$$

G_h :

$S_1 \rightarrow c S_1 c \mid bb$

$$L(G_h) = \{c^n b b c^n \mid n \geq 0\} = h(L_1)$$

فصل هشتم: ویژگیهای زبانهای مستقل از متن

جلسه شانزدهم بخش دو: لم تزریق زبان های مستقل از متن

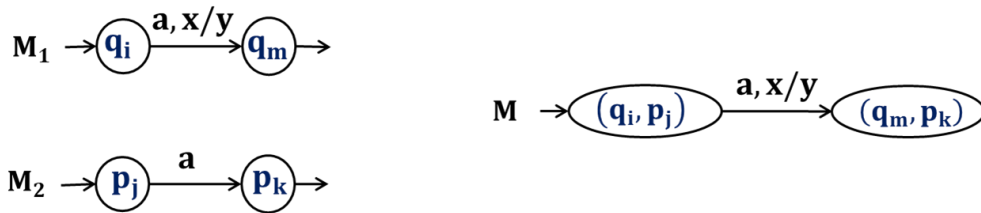
بسته بودن نسبت به اشتراک منظم:

اگر L_1 مستقل از متن و L_2 منظم باشند آنگاه $L_1 \cap L_2$ مستقل از متن است.

خلاصه برهان:

اگر M_1 ماشین پشته‌ای پذیرنده L_1 و M_2 ماشین dfa پذیرنده L_2 باشد می‌توانیم ماشین پشته‌ای M پذیرنده $L_1 \cap L_2$ را با مجموعه حالات $Q_1 \times Q_2$ داشته باشیم. حالات نهایی عبارتند از $F_1 \times F_2$ و تابع انتقال به این صورت تعریف می‌شود:

$$\begin{aligned} \delta_1(q_i, a, x) &= (q_m, y) \\ \delta_2(p_j, a) &= p_k \end{aligned} \Rightarrow \delta((q_i, p_j), a, x) = ((q_m, p_k), y)$$



نکته مهم این است که ماشین M_2 از پشته استفاده نمی‌کند به همین دلیل توانستیم انتقال را همان انتقال M_1 تعریف کنیم. اگر هر دو بخواهند از پشته استفاده کنند، به طور همزمان نمی‌شد هر دو انتقال را انجام داد.

تمرین linz:

ثابت کنید زبان L مستقل از متن است:

$$L = \{a^n b^n \mid n \leq 100 \text{ or } n \geq 200\}$$

پاسخ:

می‌دانیم زبان $L_1 = \{a^n b^n \mid n \geq 0\}$ مستقل از متن است. همچنین زبان

$$L_2 = \{a^n b^n \mid n = 101, 102, \dots, 199\}$$

نیز منظم است. حالا:

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

زبان L اشتراک یک زبان مستقل از متن و یک زبان منظم است. در نتیجه مستقل از متن است.

نتیجه:

از برهان به کار رفته در مثال فوق نتیجه می‌گیریم که :

اگر L_1 مستقل از متن و L_2 منظم باشد آنگاه $L_1 - L_2$ مستقل از متن است.

یادآوری: لم تزریق زبان های منظم چنین بود:

اگر زبان نامتناهی L منظم باشد آنگاه:

$$\exists n \quad \forall w \in L \quad |w| \geq n \quad \exists x, y, z \quad s.t$$

$$w = xyz$$

$$1. |y| \geq 1$$

$$2. |xy| \leq n$$

$$3. xy^kz \in L \quad k \geq 0$$

اکنون لم تزریق زبان های مستقل از متن را مطالعه کنیم. با کمک این لم می توان شرط کافی برای مستقل از متن نبودن یک زبان پیدا کرد. باز هم دقت کنید که زبان های متناهی منظم هستند پس مستقل از متن هستند. پس تمام تمرکز ما بر زبان های نامتناهی است.

$\exists n \quad \forall w \in L \quad |w| \geq n \quad \exists x, u, y, v, z \quad \text{s.t.}$

$$w = xuyvz$$

1. $|uv| \geq 1$

2. $|uyv| \leq n$

3. $xu^kyv^kz \in L \quad k = 0, 1, 2, \dots$

اندازه بحرانی و ثابت لم تزریق در زبانهای مستقل از متن:

ابتدا گرامر مستقل از متنی برای تولید آن زبان در نظر بگیریم. لازم است با ساده‌سازی آن را به گرامری با قواعد غیرانقباضی (به استثناء قاعده $S \rightarrow \lambda$) تبدیل کنیم. ثابت لم تزریق، تعداد متغیرهای گرامر G است که با $|V|$ نشان داده می‌شود. همچنین اگر m بیشترین طول عبارات، در سمت راست قواعد تولید باشد، عدد $n = m^{|V|+1}$ اندازه بحرانی نام دارد به این معنی که برای هر رشته w که $|w| \geq n$ تجزیه $w = xuyvz$ با شرایط لم تزریق قابل انجام است.

مثال: اندازه بحرانی را برای این گرامر محاسبه کنید:

G :

$S \rightarrow aSb \mid bAA \mid \lambda$

$A \rightarrow bA \mid ba$

پاسخ: قواعد گرامر به جز قاعده‌ی $S \rightarrow \lambda$ غیرانقباضی‌اند. پس نیازی به ساده‌سازی نیست. این گرامر دارای ۲ متغیر است. بیشترین طول سمت راست نیز ۳ است. پس اندازه بحرانی برابر است با:

$$n = m^{|V|+1} = 3^{2+1} = 27$$

لم تزریق زبان های خطی:

اگر زبان نامتناهی L خطی باشد آنگاه:

$$\exists n \quad \forall w \in L \quad |w| \geq n \quad \exists x, u, y, v, z \quad \text{s.t.}$$

$$w = xuyvz$$

$$1. |uv| \geq 1$$

$$2. |xuvz| \leq n$$

$$3. x u^k y v^k z \in L \quad k = 0, 1, 2, \dots$$

توجه:

برای اثبات مستقل از متن بودن زبان L :

کافیست گرامر مستقل از متن G پیدا کنیم که: $L(G) = L$.

کافیست یک ماشین پشته ای M پیدا کنیم که: $L(M) = L$.

توجه:

برای اثبات مستقل از متن نبودن زبان L :

کافیست نشان دهیم در شرایط لم تزریق زبان های مستقل از متن، صدق نمی کند.

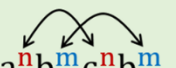
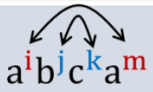
البته: می توانیم با کمی تمرین، مشاهده نمونه های معروف، و استفاده از تجربیات و

مشاهدات زبان های مستقل از متن، خطی بودن آنها، مبهم یا غیرمبهم بودن آنها را در

با یک نگاه تشخیص دهیم.

به نمونه های معروف زیر توجه کنید:

نمونه های معروف:

زبان	توضیحات
$L = \{a^n b^m c^k \mid n \neq m\}$	اجتماع دو مستقل از متن ($n > m, n < m$)، مستقل از متن است.
$L = \{a^i b^j \mid i = j^2\}$ $L = \{a^{n^2} \mid n > 1\}$ $L = \{a^{n!} \mid n \geq 0\}$	روابط غیرخطی (درجه ۲ یا بالاتر، فاکتوریل، ...) زبان مستقل از متن نمی سازند مگر آنکه تعداد الگوها متناهی شود.
نکته مهم: اگر زبان L مستقل از متن و تک الفبایی باشد ($L \subseteq a^*$) آنگاه باید منظم باشد.	
$L = \{a^n b^m c^m b^n \mid n, m \geq 0\}$	 مستقل از متن است. ماشین پشته ای دارد.
$L = \{a^n b^m c^n b^m \mid n, m \geq 0\}$	 مستقل از متن نیست. ماشین پشته ای ندارد.
$L = \{a^i b^j c^k a^m \mid i \geq j + m\}$	مستقل از متن است. فقط یک رابطه خطی دارد. شرط $i \geq j + m$ را پشته می تواند تحلیل کند. تعداد a^i را در پشته ذخیره کند سپس تعداد b^j را از پشته بردارد، c^k خنثی باشد و در پایان a^m را از پشته بردارد. اگر پشته تا پایان کار کاملاً خالی نشود رشته پذیرفته می شود.
$L = \{a^i b^j c^k a^m \mid i \geq m, j \geq k\}$	 مستقل از متن است. ماشین پشته ای دارد.
$L = \{a^i b^j c^k a^m \mid i \geq k, j \geq m\}$	 مستقل از متن نیست. ماشین پشته ای ندارد.

توجه:

اگر زبانی با فرمت شبیه به نمونه های فوق دارای چندین **رابطه خطی** (تساوی یا نامساوی) باشد و این روابط به شکل پرانتزهای **تودرتو** یا **کاملا مجزا** باشند، زبان مستقل از متن است.

$$L = \{a^i b^j c^k a^m c^p d^n \mid i \geq m, j < 2k, p = 2n\}$$



$$\{uvwv^R \mid u, v, w \in (a + b)^+, |u| = |w| = 2\}$$

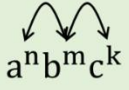
چون $|u| = |w| = 2$ پس تعداد الگوهای u و w متناهی است (با دو الفبا، ۴ رشته به طول ۲ وجود دارد) تعداد کل حالات برای u و w برابر ۱۶ است. این زبان اجتماع ۱۶ زبان مستقل از متن است. مثلاً یکی از زبانها این است: $\{abv aa v^R \mid v \in (a + b)^+\}$ ماشین پشته ای می تواند الفبای رشته v را در پشته ذخیره کند و سپس آن را با v^R مطابقت دهد. اجتماع تعدادی متناهی زبان مستقل از متن، مستقل از متن است.

$$\{a^n w w^R b^n \mid n \geq 1, w \in (a + b)^*\}$$



مستقل از متن است. ماشین پشته ای دارد.

در حالت کلی ماشین پشته ای نمی تواند تکرار $w w$ را تحلیل دهد اما می تواند $w w^R$ را تحلیل کند. (به دلیل ماهیت پشته که LIFO است).

$\{a^n b^m c^k \mid n = m, k \geq m\}$	 مستقل از متن نیست. ماشین پشته‌ای ندارد.
$\{a^n b^m c^k \mid n = m \text{ or } k \geq m\}$	اجتماع دو زبان مستقل از متن است: $(k \geq m \text{ یا } n = m)$ پس مستقل از متن است.
$\{a^n b^m c^k \mid k = n - m \}$	اجتماع دو زبان مستقل از متن است. پس مستقل از متن است.
$\{a^n b^m c^k \mid k = nm\}$	مستقل از متن نیست. $k = nm$ رابطه غیرخطی است.

-17-

فصل هشتم: ویژگی های زبان های مستقل از متن

جلسه هفدهم بخش یک:

مسائل تصمیم پذیر در زبان های مستقل از متن

مساله های تصمیم پذیر برای زبان مستقل از متن:

عضویت، تهی بودن، و متناهی بودن سه مساله معروف تصمیم پذیر برای زبان های مستقل از متن هستند.

مساله تصمیم پذیر	الگوریتم تصمیم
؟ $w \in L$	$L - \{\lambda\}$ یک زبان مستقل از متن است پس دارای گرامر مستقل از متن G است. یک گرامر نرمال چامسکی مانند G_1 دارد. برای هر رشته ناتهی w، با الگوریتم CYK می توان عضویت یا عدم عضویت w در $L(G_1)$ را تشخیص داد. این الگوریتم در هر دو صورت متوقف خواهد شد (در زمان $O(n^3)$ به پایان می رسد). البته الگوریتم جستجوی کامل نیز برای گرامر غیرانقباضی، در هر دو صورت متوقف می شود اما زمان اجرای آن نمایی است. با این توضیحات، مساله تعیین عضویت w در L تصمیم پذیر است. مساله تشخیص عضویت رشته تهی: الگوریتمی داریم که متغیرهای پوچ شونده را در گرامر G تشخیص می دهد. کفایت ببینیم آیا S پوچ شونده هست یا خیر. λ عضو L است اگر و تنها اگر S پوچ شونده باشد.
؟ $L = \phi$	L مستقل از متن است پس یک گرامر مستقل از متن G وجود دارد که $L = L(G)$. الگوریتمی وجود دارد که متغیرهای زائد و غیرمفید را تشخیص می دهد. کفایت ببینیم متغیر شروع S مفید است یا غیرمفید؟ اگر مفید باشد یعنی با شروع از S دستکم یک رشته تولید می شود.
L متناهی است ؟	L مستقل از متن است پس دارای گرامر مستقل از متن G است. ابتدا با حذف قواعد λ، حذف قواعد واحد و حذف متغیرهای زائد آن را به گرامر غیرانقباضی G_1 تبدیل می کنیم ($L = L(G) = L(G_1)$). حالا گراف وابستگی G_1 را تشکیل می دهیم. متغیرهای G_1 گره های این گراف هستند و اگر با شروع از A بتوان به عبارتی شامل B رسید، یال جهت داری از A به B وجود دارد. زبان L نامتناهی است اگر و تنها اگر این گراف شامل دور باشد. الگوریتم هایی وجود دارند که وجود یا عدم وجود دور در گراف جهت دار را تشخیص می دهند.

مثال - برای این گرامرها، متناهی یا نامتناهی بودن $L(G)$ را بررسی کنید.

G_1 :

$S \rightarrow ABa$

$A \rightarrow BBa \mid a$

$B \rightarrow Db \mid b$

$D \rightarrow ab$

G_2 :

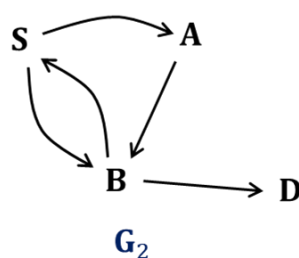
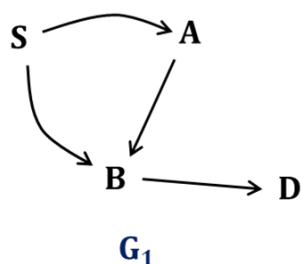
$S \rightarrow ABa$

$A \rightarrow BBa \mid a$

$B \rightarrow Db \mid bS$

$D \rightarrow ab$

الف: گراف وابستگی گرامر G_1 فاقد دور است. $L(G_1)$ متناهی است
ب: گراف وابستگی گرامر G_2 دارای دور است. $L(G_2)$ نامتناهی است.



مساله های تصمیم پذیر دیگر:

با اطمینان از تصمیم پذیری سه مساله قبل، و استفاده از ویژگی های بستاری زبان های مستقل از متن می توانیم از تصمیم پذیر بودن برخی مساله ها مطمئن شویم. برای نمونه این دو مساله تصمیم پذیرند:

مساله تصميم پذير	الگوريتم تصميم
آيا رشته اي با طول حداكثر m در L وجود دارد؟	L مستقل از متن است. فرض كنيم L_m مجموعه همه رشته هاي به طول حداكثر m باشد. $L_m = L(\{\lambda\} + \Sigma + \Sigma^2 + \dots + \Sigma^m)$ L_m متناهي است پس منظم است. پس $L \cap L_m$ مستقل از متن است. مسأله تهی یا ناتهی بودن $L \cap L_m$ بودن تصميم پذير است. زبان L رشته اي به طول حداكثر m دارد اگر و تنها اگر $L \cap L_m$ ناتهی باشد.
آيا رشته اي با طول حداقل m در L وجود دارد؟	فرض كنيم L_3 مجموعه رشته هاي با طول کمتر از m باشد. $\overline{L_3}$ شامل رشته هاي با طول حداقل m است. L_3 منظم است پس L_3 هم منظم است. پس $L \cap \overline{L_3}$ مستقل از متن است. رشته اي با طول حداقل m در L وجود دارد اگر و تنها اگر $L \cap \overline{L_3}$ ناتهی باشد. کافيست تهی بودن یا نبودن اشتراك را تعيين كنيم. اين مساله تصميم پذير است.
آيا زبان مستقل از متن L زیر مجموعه زبان منظم L' است؟	فرض كنيم L مستقل از متن و L' منظم باشد. $L \subseteq L'$ اگر و تنها اگر $L - L' = \emptyset$. در ضمن $L - L'$ مستقل از متن است زیرا اشتراك يك زبان مستقل از متن و يك زبان منظم است: $(L - L' = L \cap \overline{L'})$ پس اين مساله تصميم پذير است.

تمرین: تشخیص زبان های مستقل از متن، قطعی، و غیرقطعی.

زبان	منظم	مستقل از متن	
		قطعی	غیر قطعی
$L = \{a^n b^m c^k \mid n = m \text{ or } k \geq m\}$	x	x	✓
$L = \{a^n b^m c^k \mid n = m \text{ or } k \neq m\}$	x	x	✓
$L = \{a^n b^m \mid 2n \leq m \leq 3n\}$	x	x	✓
$L = \{a^{2^n} \mid n \bmod 3 = 0\}$	x	x	x
$L = \{w \mid n_a(w) = n_b(w) = n_c(w)\}$	x	x	x
$L = \{a^n b^n c^n \mid n \geq 0\}$	x	x	x
$L = \{uvwu^R v^R \mid u, v \in (a + b)^+\}$	x	x	x

زبان	منظم	مستقل از متن	
		قطعی	غیر قطعی
$L = \{a^p \mid p \text{ is prime} \}$	x	x	x
$L = \{a^n b^m c^n d^m \mid n, m \geq 0\}$	x	x	x
$L = \{a^n b^j c^k \mid k > n, k < j\}$	x	x	x
$L = \{a^n b^j c^k \mid n < j, n \leq k \leq j\}$	x	x	x
$L = \{a^n b^m c^k d^j \mid n = k, m = j\}$	x	x	x
$L = \{a^n b^m c^m d^n c^{2m} \mid n, m \geq 0\}$	x	x	x
$L = \{a^n b^m c^n d^m \mid n, m \geq 0\}$	x	x	x
$L = \{a^{nm} \mid n, m \text{ are primes} \}$	x	x	x

مثال (SIPSER): تشخیص زبان های مستقل از متن، قطعی، و غیرقطعی.

زبان های C و D مستقل از متن نیستند.

$$C = \{a^i b^j c^k \mid 0 \leq i \leq j \leq k\}.$$

$$D = \{ww \mid w \in \{0,1\}^*\} \quad \text{are not CFLs.}$$

زبان های A و B مستقل از متن هستند اما قطعی نیستند. (ماشین پشته ای قطعی ندارند).

$$A = \{a^i b^j c^k \mid i, j, k \geq 0 \text{ and } i = j \text{ or } i = k\}$$

$$B = \{ww^R \mid w \in \{0,1\}^*\}, \quad \text{are not DCFLs.}$$

مروری بر ویژگیهای بستاری:

خصوصیات بستاری زبان های منظم

اجتماع نامتناهی	همومورفیسم و همومورفیسم معکوس	بستار ستاره	متمم	معکوس	خارج قسمت از چپ	خارج قسمت از راست	تفاضل	الحاق	اشتراک	اجتماع
×	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓



توجه: در مورد اجتماع نامتناهی توجه کنید که هیچکدام از رده های زبان های منظم و زبان های مستقل از متن نسبت به اجتماع نامتناهی بسته نیستند. برای مثال زبان $L = \{a^n b^n c^n : n \geq 0\}$ نه منظم است نه مستقل از متن است. اما می توان آن را به صورت اجتماع بی نهایت تک عضوی نوشت. هر تک عضوی، منظم و مستقل از متن است:

$$L = \{a^n b^n c^n : n \geq 0\} = \{\lambda\} \cup \{abc\} \cup \{a^2 b^2 c^2\} \cup \dots$$

خصوصیات بستاری زبان های مستقل از متن

نوع زبان	اجتماع	اشتراک	الحاق	تفاضل	خارج قسمت	خارج قسمت منظم	معکوس	متمم	بستار ستاره	همریختی	همریختی معکوس
مستقل از متن	✓	✗	✓	✗	✗	✓	✓	✗	✓	✓	✓
مستقل از متن قطعی	✗	✗	✗	✗	✗	✓	✗	✓	✗	✗	✓

$$\frac{\text{مستقل از متن}}{\text{منظم}} = \text{مستقل از متن} \quad \frac{\text{مستقل از متن}}{\text{مستقل از متن}} = \text{نامعلوم}$$

THEOREM 2.42

The class of DCFLs is closed under complementation.

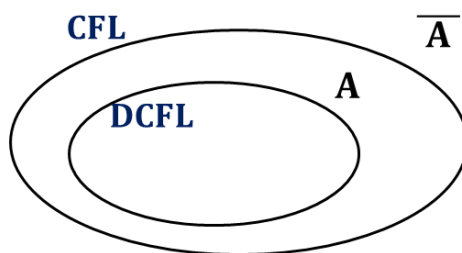
قضیه:

زبان های مستقل از متن (غیرقطعی) نسبت به مکمل بسته نیستند اما زبان های مستقل از متن قطعی نسبت به مکمل بسته اند.

بنابراین:

اگر A مستقل از متن باشد اما $\overline{A}$ مستقل از متن نباشد می توان نتیجه گرفت که A مستقل از متن قطعی نیست.

$$A \in \text{CFL} \text{ and } \overline{A} \notin \text{CFL} \Rightarrow A \notin \text{DCFL}$$



مثال (SIPSER): این مثال نمونه ای از کاربرد نکته قبل است.
ثابت کنید زبان A مستقل از متن است اما قطعی نیست.

$$A = \{a^i b^j c^k \mid i \neq j \text{ or } j \neq k\}$$

پاسخ:

با توجه به مفهوم (or) زبان A اجتماع دو زبان مستقل از متن است. پس مستقل از متن است. اما ثابت می کنیم مکمل A مستقل از متن نیست.
برهان خلف: با فرض خلف، اگر $\overline{A}$ مستقل از متن باشد آنگاه اشتراک آن با یک زبان منظم، منظم است پس $\overline{A} \cap L(a^* b^* c^*)$ باید منظم باشد.
اما با محاسبه اشتراک می بینیم که :

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

این زبان حساس به متن است و مستقل از متن نیست. پس تناقض رخ داد.
پس فرض خلف نادرست است یعنی $\overline{A}$ مستقل از متن نیست.
نتیجه: با آنکه A مستقل از متن است اما $\overline{A}$ مستقل از متن نیست. نتیجه می گیریم A مستقل از متن از نوع قطعی نیست.

LEMMA 2.41

Every DPDA has an equivalent DPDA that always reads the entire input string.

نکته (SIPSE):

ماشین های پشته ای در حالت کلی ممکن است پیش از مصرف کامل رشته ورودی دچار حالت توقف (Hanging) یا دچار حلقه (Looping) شوند. اما اگر ماشین پشته ای از نوع قطعی باشد می توان آن را طوری بازطراحی کرد که هیچ گاه این دو حالت رخ ندهند و هر رشته ورودی، تا انتها مصرف شود.

خلاصه و جمع بندی ویژگی های بستاری:

در جدول زیر رنگ سبز نشان دهنده ویژگی های بستاری تایید شده است. رنگ مشکی، بسته نبودن نسبت به آن عملگر را نشان می دهد.

$h(L_1)$	L_1^R	L_1^*	$L_1 L_2$	$\overline{L_1}$	$L_1 - L_2$	$L_1 \cap L_2$	$L_1 \cup L_2$	
								منظم
								خطی
								مستقل از متن
								مستقل از متن قطعی

- زبان های خطی تحت اشتراک منظم و الحاق منظم بسته اند.

- زبان های منظم و مستقل از متن تحت عمل بر زدن بسته اند.

مساله های تصمیم ناپذیر برای زبان مستقل از متن:

مساله شمول (زیرمجموعه بودن) برای زبان ها و گرامرهای منظم تصمیم پذیر بود اما برای گرامرهای مستقل از متن و رده های بزرگتر از آن، تصمیم پذیر نیست. مساله های تصمیم ناپذیر را در فصل دهم به شکل کاملتری بررسی خواهیم کرد. در اینجا به دو مورد اشاره شده است:

مسائل تصمیم ناپذیر	توضیحات
؟ $L_1 \subseteq L_2$	$L_1 \subseteq L_2$ اگر و تنها اگر $L_1 - L_2 = \emptyset$. اما تفاضل دو زبان مستقل از متن لزوماً مستقل از متن نیست. (در رده زبان های منظم این مساله تصمیم پذیر بود زیرا تفاضل دو زبان منظم، منظم است). توجه: اگر L_1 مستقل از متن و L_2 منظم باشد این مساله تصمیم پذیر می شود. چون $L_1 - L_2$ مستقل از متن می شود.
؟ $L_1 = L_2$	$L_1 = L_2$ اگر و تنها اگر $L_1 \subseteq L_2$ و $L_2 \subseteq L_1$. پس این مساله به مساله قبلی کاهش می یابد. در حالت کلی و برای دو زبان مستقل از متن، تصمیم پذیر نیست. (در رده زبان های منظم این مساله تصمیم پذیر بود).

-18-

فصل نهم: ماشین تورینگ
جلسه هجدهم بخش یک: ماشین تورینگ استاندارد.

ماشین تورینگ استاندارد (turing machine)

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

Q مجموعه‌ای ناتهی و متناهی از حالت های داخلی واحد کنترل است.

Σ مجموعه الفبای رشته های ورودی است ، متناهی و ناتهی است.

Γ مجموعه الفبای و علامات روی نوار حافظه، متناهی و ناتهی است.

q_0 یکی از اعضای Q که حالت اولیه است.

F زیرمجموعه ای از Q که حالات پایانی (پذیرش) نام دارند.

$\square$ علامتی برای نشان دادن سلول $\square$ ، نباید عضو Σ باشد.

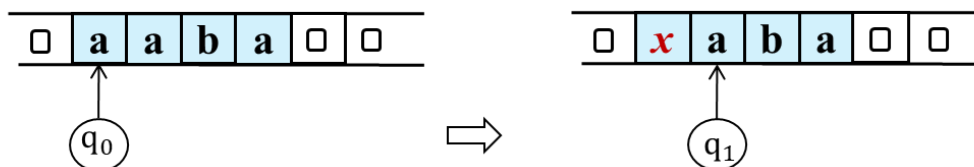
δ : تابع انتقال ماشین تورینگ است:

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

(برای تورینگ غیرقطعی به جای یک انتقال مجموعه ای از انتقال ها داریم.)

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

تغییر حالت از q_0 به q_1 ، نوشتن x به جای a ، و حرکت هد به سمت راست:

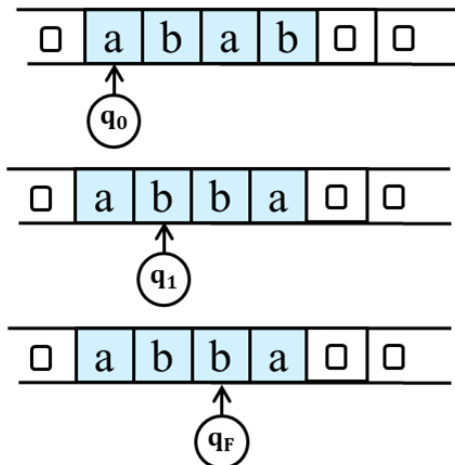


مثال ۱: زبان های منظم نیز ماشین تورینگ پذیرنده دارند. مثلا ماشین تورینگ پذیرنده برای $L(ab(a + b)^*)$ طراحی کنیم:

پاسخ:



برای رشته ورودی $w = abab$ ماشین پس از دو گام در حالت پذیرش متوقف می شود، در حالی که هنوز رشته ورودی را به طور کامل مشاهده نکرده است.

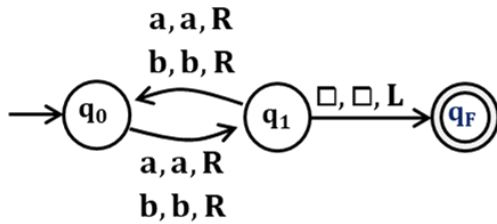


نتیجه: برخلاف ماشین DFA که تعداد گامها دقیقا به اندازه طول رشته بود، در ماشین تورینگ تعداد گامها ممکن است کمتر (یا بیشتر) از طول رشته باشد.

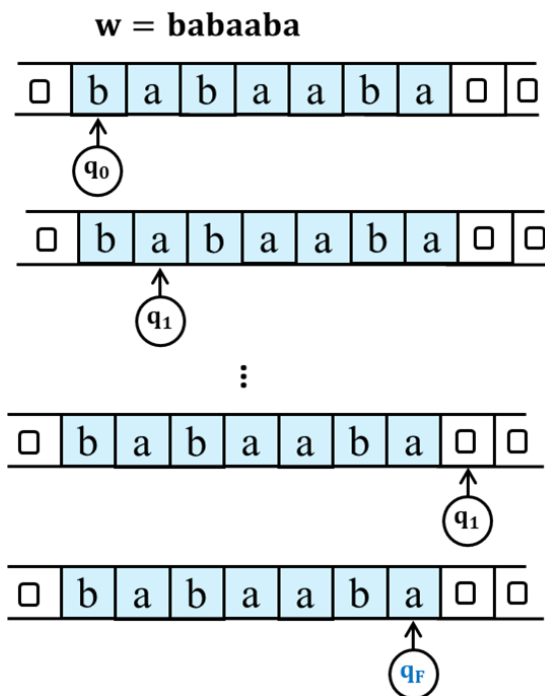
مثال ۲: ماشین تورینگ پذیرنده برای رشته‌های به طول فرد طراحی کنیم.

$$L = \{w: |w| = 2k + 1, k \geq 0\}, \quad \Sigma = \{a, b\}$$

پاسخ:



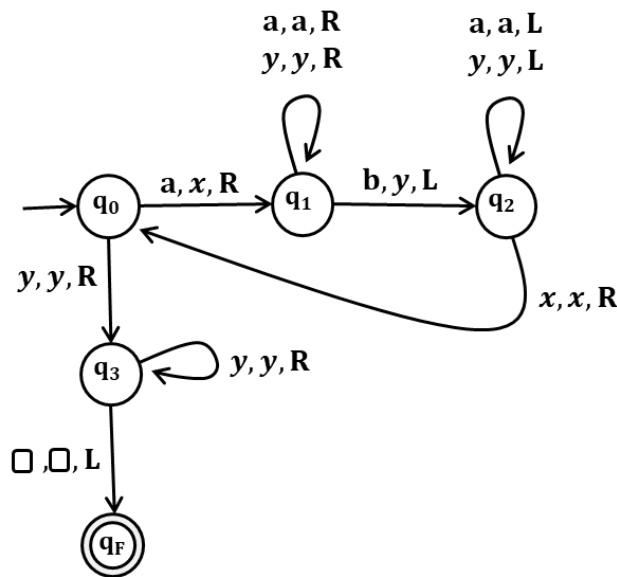
فقط برای رشته‌های به طول فرد در در q_1 با $\square$ روبرو می‌شویم. این ماشین اگر در q_1 با $\square$ روبرو شود وارد حالت نهایی شده همانجا متوقف می‌شود. اگر در q_0 با $\square$ روبرو شود در همان حالت غیر نهایی متوقف می‌شود، چون انتقالی تعریف نشده است.



در دو مثال فوق، برای پذیرش دو زبان منظم، از ماشین تورینگ استفاده کردیم. تابع انتقال در این ماشین‌ها محتوای هیچ سلولی را تغییر نمی‌دهد مانند آن است که از نوشتن استفاده نکرده باشد. همچنین (به جز آخرین انتقال) هد فقط به راست حرکت می‌کند پس هیچ‌گاه به یادآوری سلول‌های قبل نیاز ندارد. هنگامی که ماشین تورینگ از حافظه نوشتنی استفاده نکند (یا مقدار حافظه نوشتنی‌اش $O(c)$ باشد) توان محاسباتی‌اش با ماشین متناهی برابر است و $L(M)$ منظم است.

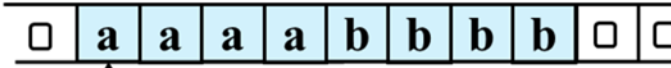
مثال ۳: ماشین تورینگ توانایی پذیرش زبان مستقل از متن را هم دارد. برای مثال:

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

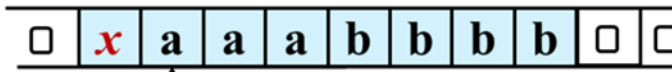


مثال: در همین ماشین نحوه پذیرش رشته $w = a^4 b^4$ را مشاهده کنیم:

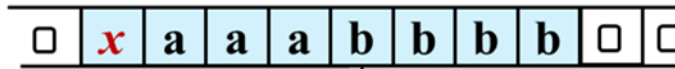
$$w = a^4b^4$$



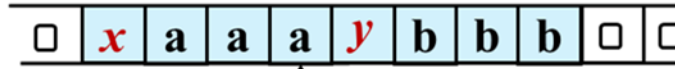
q_0 اولین **a** رو با **x** علامت گذاری می‌کنم



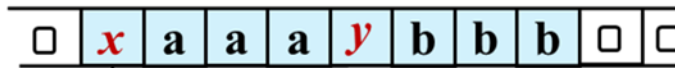
q_1 دنبال **b** می‌گردم با سایر حروف کاری ندارم $\Rightarrow$



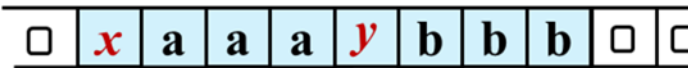
q_1 **b** رو یافتیم. با **y** علامت می‌زنم برمی‌گردم



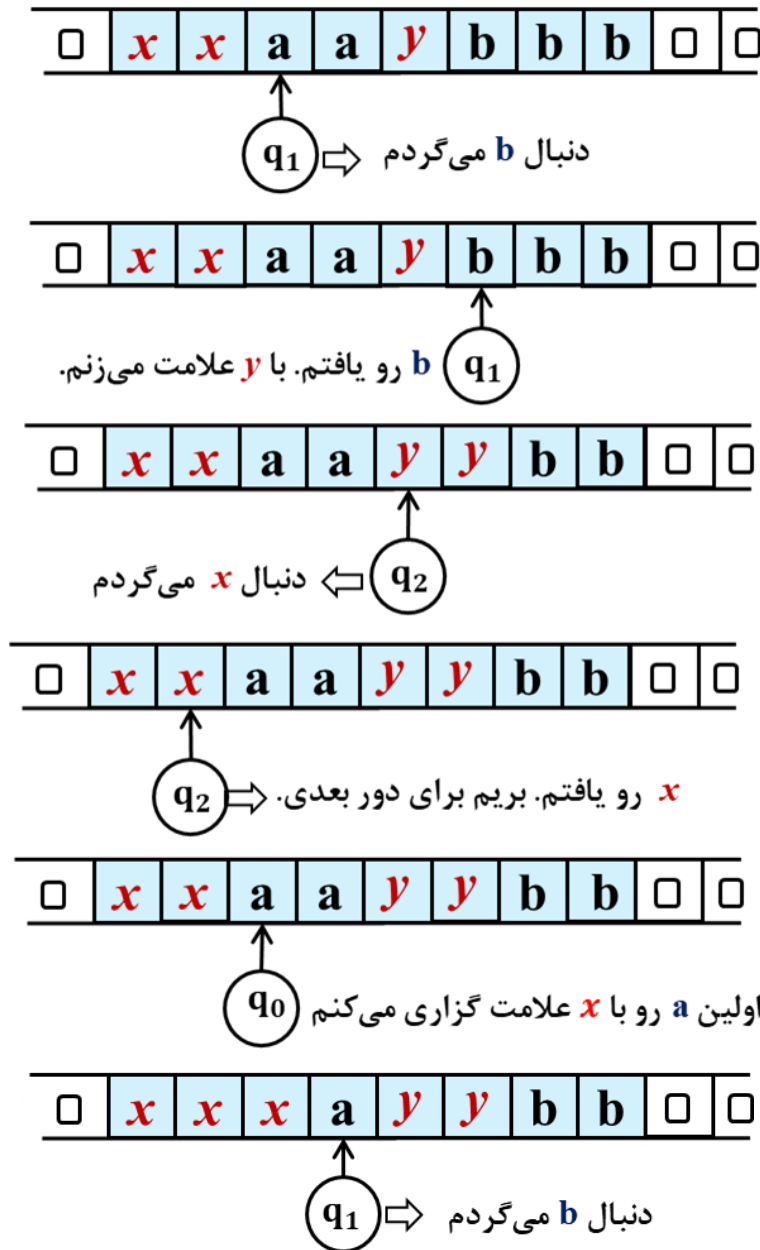
q_2 دنبال **x** می‌گردم تا برگردم به ابتدای رشته $\Leftarrow$



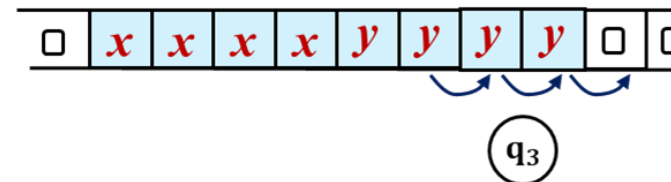
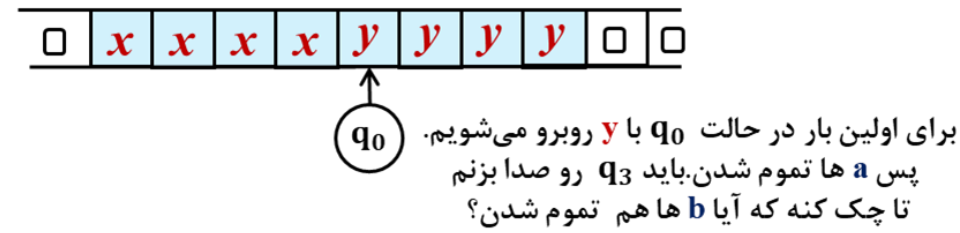
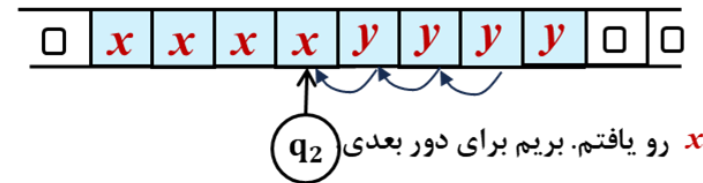
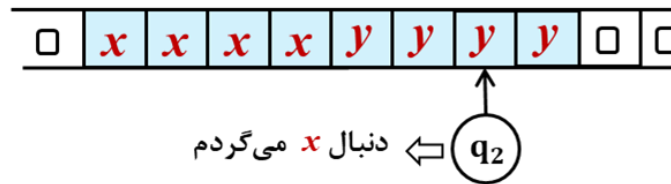
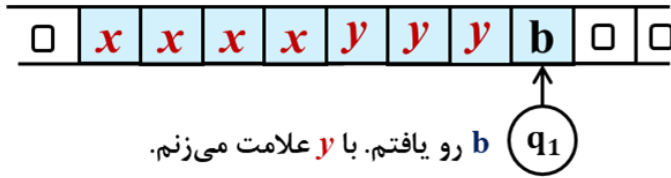
q_2 **x** رو یافتیم. بریم برای دور بعدی.

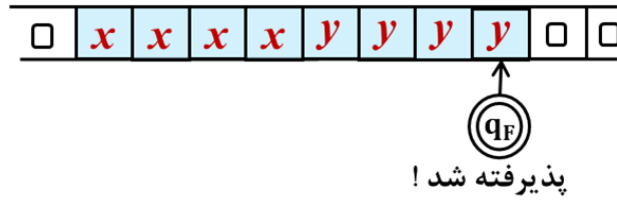


q_0 اولین **a** رو با **x** علامت گذاری می‌کنم



به همین ترتیب هر جفت a و b را علامت گذاری می‌کند در آخرین دور پس از علامت گذاری آخرین b با y اینطور ادامه می‌یابد:





فصل نهم: ماشین تورینگ

جلسه هجدهم بخش دو:

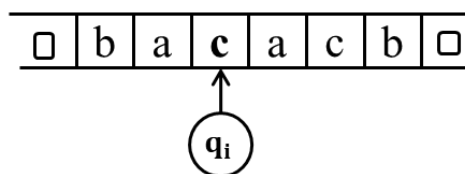
توصیف لحظه‌ای ماشین تورینگ - ماشین تورینگ کران‌دار خطی

توصیف لحظه ای ماشین تورینگ

بهترین روش برای نمایش توالی حرکات در ماشین تورینگ، توصیف لحظه ای آن است. این توصیف، پیکربندی ماشین (حالت جاری، محتوای نوار، موقعیت هد، حالت بعدی) را در همان لحظه نشان می‌دهد.

برای مثال $baq_i cacb$ یک توصیف لحظه‌ای است، طبق این توصیف، هم اکنون رشته $bacacb$ روی نوار درج شده، ماشین در حالت q_i قرار دارد و هد روی سومین سلول قرار دارد و حرف c را می‌خواند. قرارداد آن است که سلولی که هد روی آن قرار دارد دقیقاً در سمت راست نام حالت فعلی بیاید.

$baq_i cacb$



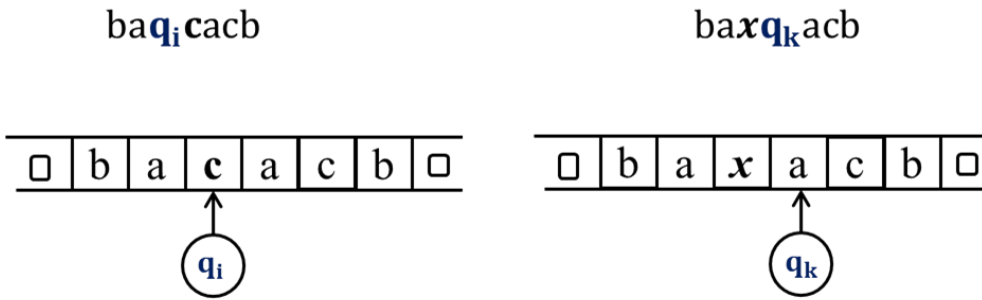
انتقال از یک پیکربندی به پیکربندی دیگر را با علامت $\vdash$ نشان می‌دهیم. این عبارت:

$$baq_i c a c b \vdash baxq_k a c b$$

نشان می‌دهد در حالت ماشین در حالت q_i با خواندن c روی نوار، به حالت q_k

می‌رود روی نوار x را می‌نویسد و هد را به سمت راست می‌برد. به عبارتی

$$\delta(q_i, c) = (q_k, x, R)$$

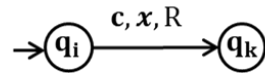


طبق معمول برای نمایش تعداد دلخواهی انتقال، از علامت $\vdash^*$ استفاده می‌کنیم:

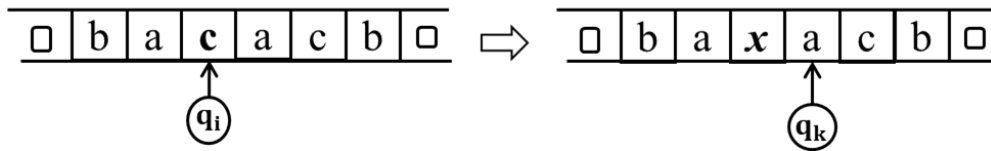
$$baq_i c a c b \vdash^* baxyxq_m b$$

مثال:

انتقال $\delta(q_i, c) = (q_k, x, R)$ به این فرم ها قابل نمایش است:



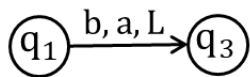
$$baq_i c a c b \vdash b a x q_k a c b$$

**مثال:**

به نمایش یک انتقال با سه روش مختلف توجه کنید:

$$abaq_1 bba \vdash abq_3 aaba$$

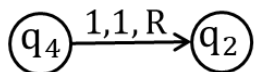
$$\delta(q_1, b) = (q_3, a, L)$$



مثال: به نمایش یک انتقال با سه روش مختلف توجه کنید:

$$0010q_4 1 \vdash 00101q_2$$

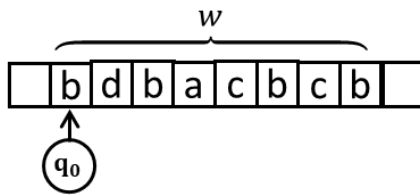
$$\delta(q_4, 1) = (q_2, 1, R)$$



توجه:

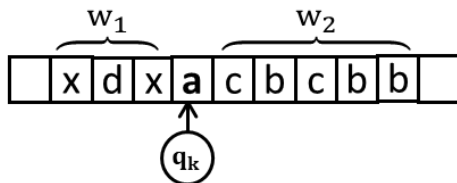
در شروع کار، رشته w ورودی روی نوار درج شده، ماشین در حالت q_0 قرار دارد و هد روی اولین حرف از w است. توصیف لحظه اولیه تورینگ استاندارد چنین

است: $q_0 w$



فرض کنید پس از چند حرکت، به توصیف $w_1 q_k a w_2$ برسیم:

$$q_0 w \vdash_M^* w_1 q_k a w_2$$



اکنون در حالت q_k هستیم و هد حرف a را می‌خواند. اگر انتقال $\delta(q_k, a)$ تعریف نشده باشد، ماشین همین جا متوقف می‌شود.

تعریف:

به دنباله انتقال هایی که از $q_0 w$ به یک حالت توقف می‌رسد یک محاسبه می‌گوییم.

سه حالت ممکن برای هر رشته ورودی w :

فرض کنیم ماشین تورینگ M پذیرنده زبان L باشد. برای هر رشته w روی نوار، یکی از این ۳ حالت رخ خواهد داد:

الف: اگر $w \in L$ ماشین در یک حالت نهایی (پذیرش) متوقف می شود:

$$q_0 w \vdash_M^* w_1 q_F w_2$$

ب: اگر $w \notin L$ یکی از این دو حالت رخ می دهد:

حالت اول: ماشین در یک حالت غیرنهایی (غیرپذیرش) متوقف می شود.

$$q_0 w \vdash_M^* w_1 q_F w_2$$

حالت دوم: ماشین در یک حلقه بی نهایت قرار بگیرد و متوقف نمی شود.

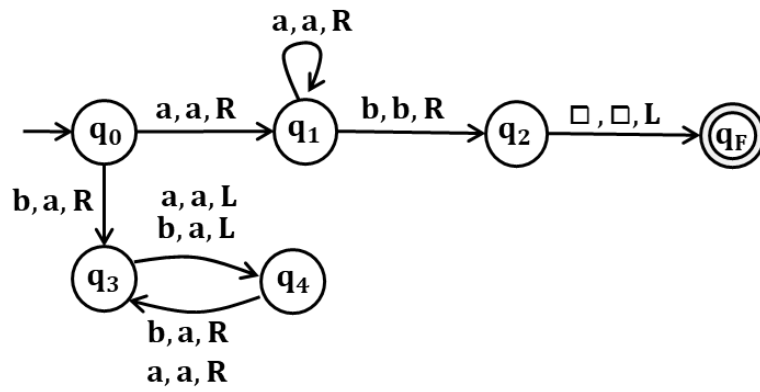
$$q_0 w \vdash_M^* \infty$$

خلاصه:

$$\forall w \in \Sigma^* \left\{ \begin{array}{ll} w \in L \longrightarrow & \text{توقف در حالت نهایی} \\ w \notin L \left\{ \begin{array}{l} \longrightarrow \text{توقف در حالت غیرنهایی} \\ \longrightarrow \text{حلقه بینهایت - عدم توقف} \end{array} \right. & \end{array} \right.$$

مثال:

این ماشین تورینگ زبان $L(M) = \{aa^n b | n \geq 0\}$ را می‌پذیرد.



به ازای $w = aaab \in L$ ماشین در حالت نهایی q_F متوقف می‌شود.

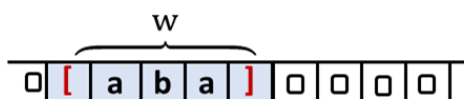
به ازای $w = acb \notin L$ ماشین در حالت غیرنهایی q_2 متوقف می‌شود.

به ازای $w = baab \notin L$ ماشین در حلقه بی‌نهایت افتاده و متوقف نمی‌شود:

$$baab \vdash_M^* \infty$$

تورینگ کراندار خط :

ماشین تورینگ را کراندار خطی (LBA) می‌نامیم هرگاه در پردازش هر رشته ورودی w فقط از همان سلول هایی که رشته ورودی در آنها درج شده است برای نوشتن استفاده کند. در این صورت حافظه نوشتنی $\theta(n)$ است. برای ایجاد تمایز بین لبه های رشته ورودی با سایر سلول های خالی، رشته ورودی را درون علامت [] قرار می‌دهیم.



البته اگر حداکثر حافظه مصرفی یک ماشین تورینگ برای هر رشته به طول n تابع خطی به فرم $An + B$ باشد (به عبارتی $O(n)$ باشد) باز هم می‌توان با استفاده از نمادها و حالات بیشتر، آن را به ماشین تورینگی معادلی تبدیل کرد که فقط از همان سلول های w برای نوشتن استفاده می‌کند. به همین دلیل در یک تعریف جامع تر داریم:

تعریف:

ماشین تورینگ M ، را کراندار خطی گوییم هرگاه حافظه مصرفی آن برای هر رشته به طول n از مرتبه $O(n)$ باشد.

توجه:

توان محاسباتی ماشین تورینگ LBA، خانواده زبان‌های حساس به متن است.

زبان های حساس به متن:

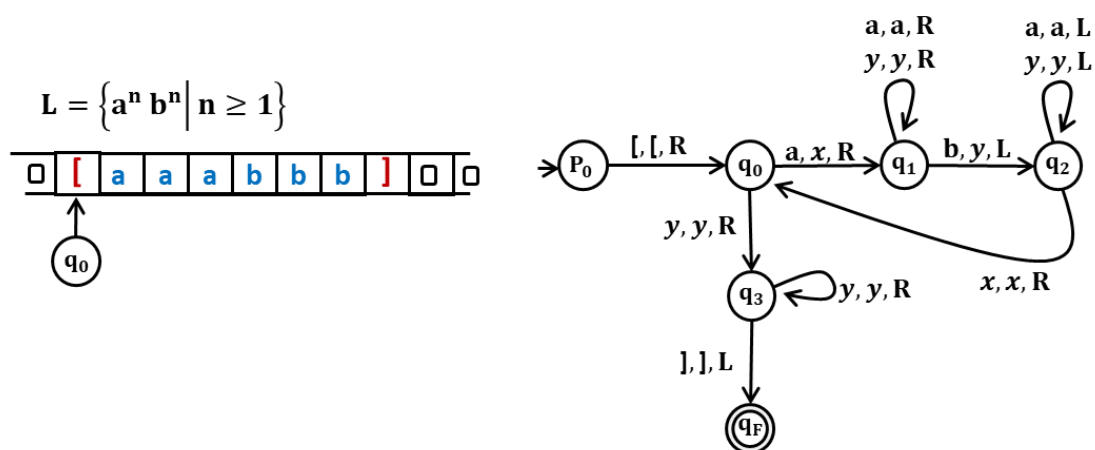
L حساس به متن است $\Leftrightarrow L - \{\lambda\}$ گرامر حساس به متن داشته باشد.

L حساس به متن است $\Leftrightarrow L - \{\lambda\}$ ماشین پذیرنده LBA داشته باشد.

یادآوری انواع گرامرها:

انواع گرامر	فرم قواعد	سمت چپ	سمت راست
بدون محدودیت	$\alpha \rightarrow \beta$	$\alpha \in (TUV)^+ - (T)^*$	$\beta \in (TUV)^*$
حساس به متن	$\alpha \rightarrow \beta$	$\alpha \in (TUV)^+ - (T)^*$	$\beta \in (TUV)^*, \alpha \leq \beta $
مستقل از متن	$A \rightarrow \beta$	$A \in V$	$\beta \in (TUV)^*$
خطی	$A \rightarrow xBy$ $A \rightarrow x$	$A \in V$	$B \in V$ $x \in T^*$
خطی راست (منظم)	$A \rightarrow xB$ $A \rightarrow x$	$A \in V$	$B \in V$ $x \in T^*$
خطی چپ (منظم)	$A \rightarrow Bx$ $A \rightarrow x$	$A \in V$	$B \in V$ $x \in T^*$

مثال ۱: پیش از این برای $L = \{a^n b^n | n \geq 1\}$ یک تورینگ پذیرنده طراحی کردیم. با کمی دقت متوجه می‌بینیم که این ماشین کراندار خطی است. می‌توانیم با قرار دادن هر رشته بین دو علامت ابتدا و انتها این ماشین را بازطراحی کنیم. این زبان حساس به متن است.



(البته پیش از این می‌دانستیم این زبان مستقل از متن است و هر زبان مستقل از متن، حساس به متن هم هست).

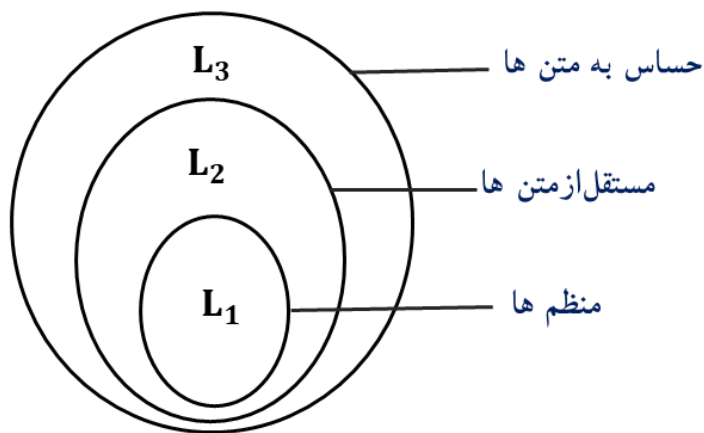
تمرین ۱: به این زبانها دقت کنید:

$$L_3 = \{a^n b^n c^n \mid n \geq 1\}$$

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

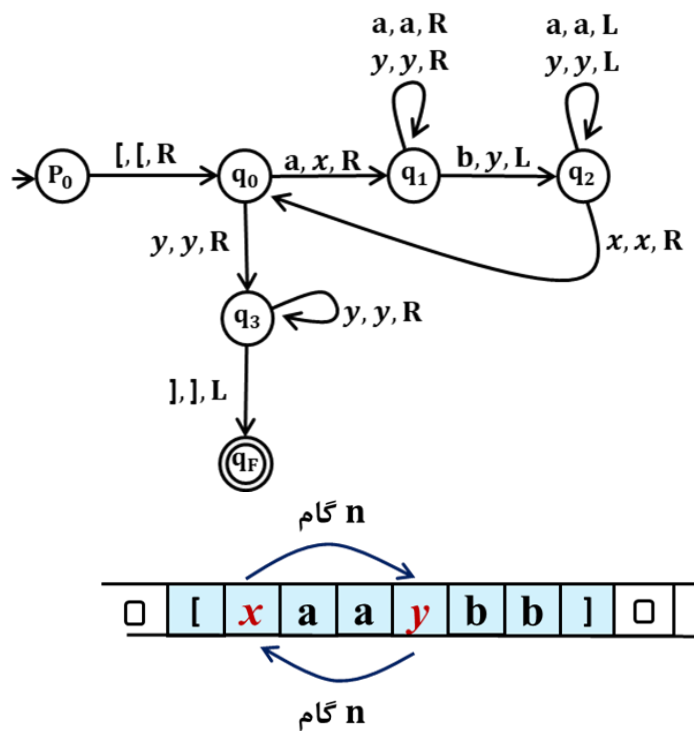
$$L_1 = \{a^n \mid n \geq 1\}$$

L_1 منظم است پس مستقل از متن و حساس به متن هم هست.
 L_2 منظم نیست، اما مستقل از متن و در نتیجه حساس به متن است.
 L_3 منظم یا مستقل از متن نیست، اما حساس به متن است.
 مکان قرار گرفتن هر کدام را در رده بندی زبانها مشاهده می کنید:



تمرین ۲: مرتبه زمانی پذیرش w در LBA مربوط به $L = \{a^n b^n | n \geq 1\}$

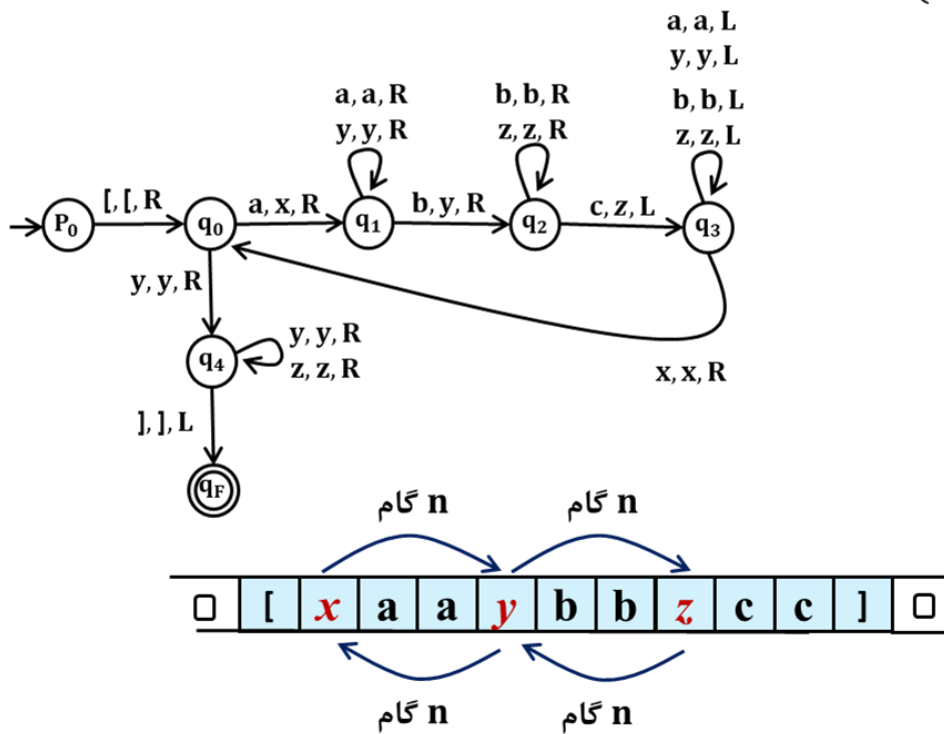
دیدیم که در ماشین LBA حافظه مصرفی برای پذیرش هر رشته به طول n از مرتبه $O(n)$ است. اما مرتبه زمانی بستگی به نحوه عملکرد ماشین دارد. مرتبه زمانی، یعنی تعداد گامها برای پذیرش رشته های عضو L ، را در این LBA بررسی کرده نشان می دهیم $O(n^2)$ است.



رشته $w = a^n b^n$ به طول $|w| = 2n$ است. این رشته از n جفت a, b تشکیل شده و ماشین باید هر جفت از آنها را با هم تطبیق دهد. برای تطبیق هر جفت a و b باید n گام به راست و n گام به چپ برداشته شود. پس n جفت و برای هر جفت $2n$ گام داریم: تعداد کل گامها $n \times 2n = 2n^2$ است. پس پذیرش رشته ها در زمان $O(n^2)$ انجام می شود.

تمرین ۳: مرتبه زمانی پذیرش w در LBA مربوط به $L = \{a^n b^n c^n | n \geq 1\}$

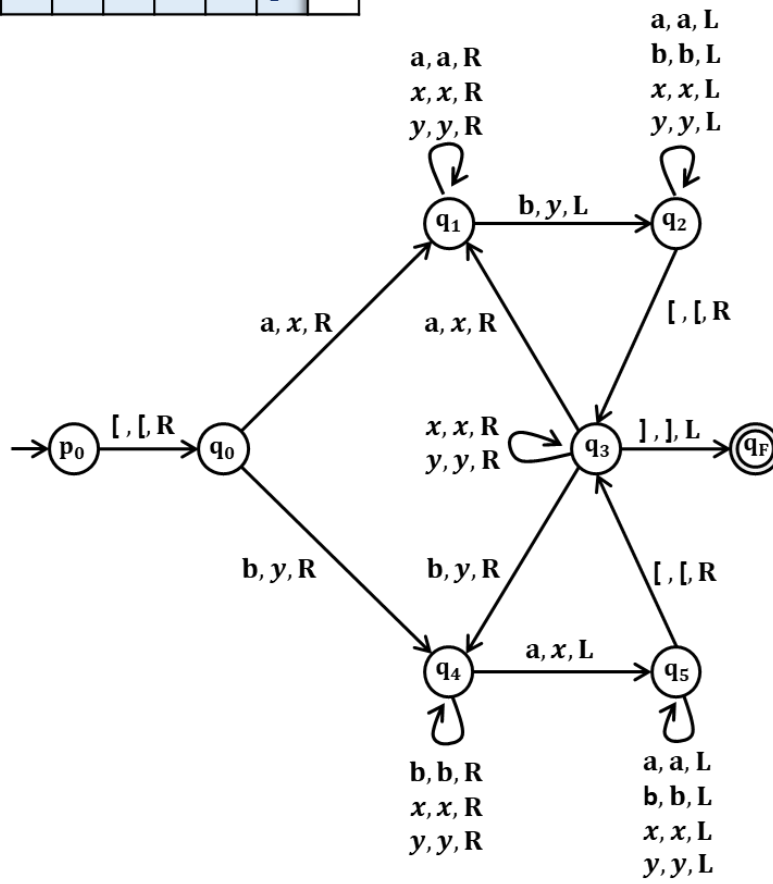
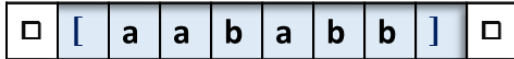
این زبان هم دارای ماشین LBA است. در این ماشین نیز مرتبه حافظه مصرفی $O(n)$ است و مرتبه زمانی، یعنی تعداد گامها برای پذیرش هر رشته عضو L ، در $O(n^2)$ است.



$w = a^n b^n c^n$ رشته به طول $3n$ است. این رشته دارای n سه تایی a, b, c است و ماشین برای تطبیق هر کدام، $4n$ گام برمی دارد. پس تعداد گامها $4n^2$ است. مرتبه زمانی پذیرش w ، $O(n^2)$ است.

تمرین ۴: ماشین تورینگ کران دار خطی برای زبان L

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



-19-

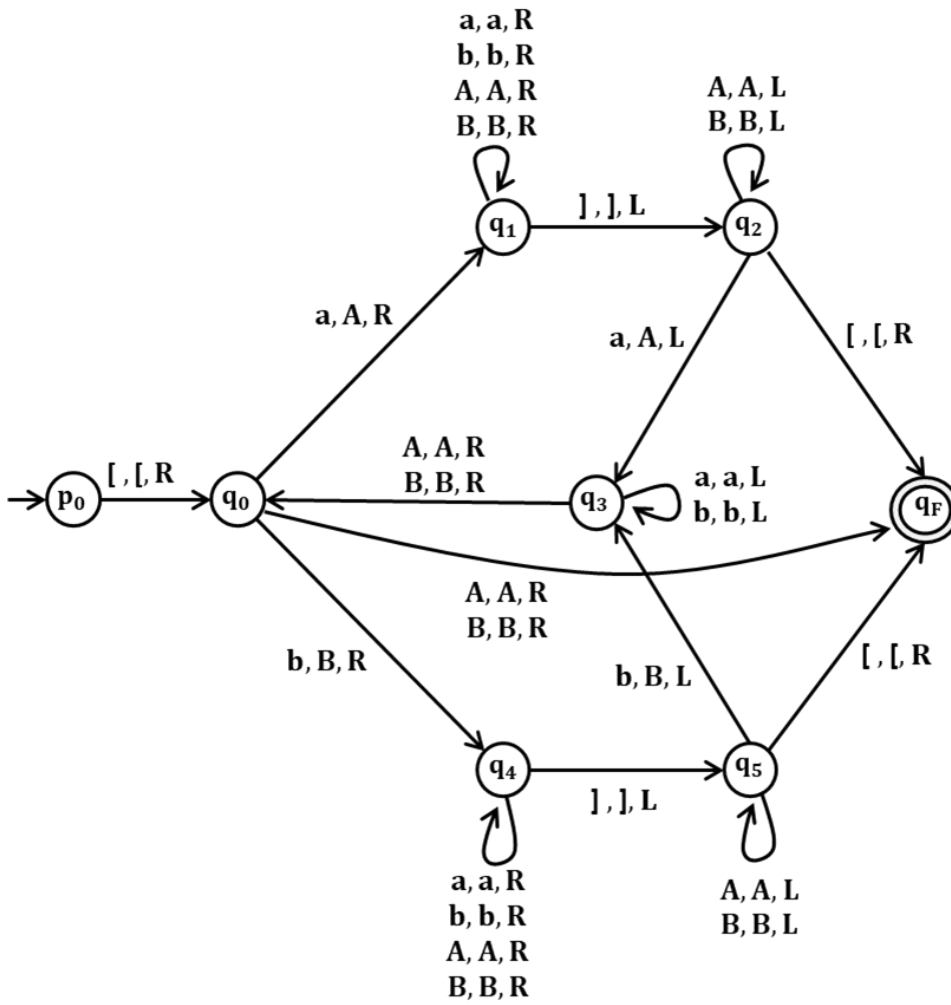
فصل نهم: ماشین تورینگ

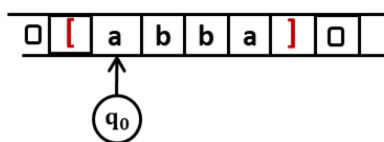
جلسه نوزدهم بخش یک:

ماشین تورینگ کران دار خطی - انواع ماشین های تورینگ

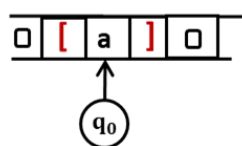
مثال ۴: ماشین تورینگ LBA برای زبان رشته‌های آینه‌ای (پالیندروم)

$$L = \{w \in (a + b)^+ \mid w^R = w\}$$





این رشته در پایان محاسبات، از حالت q_0 وارد حالت پذیرش می‌شود.



این رشته در پایان محاسبات، از حالت q_2 وارد حالت پذیرش می‌شود.

ماشین تورینگ غیر قطعی ، ماشین تورینگ قطعی :

در ماشین تورینگ قطعی، نتیجه انتقال $\delta(q, a)$ عنصری یکتا از $Q \times T \times \{R, L\}$ است.

مثلاً:

$$\delta: Q \times T \rightarrow Q \times T \times \{R, L\}$$

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

اما در ماشین تورینگ غیر قطعی نتیجه انتقال $\delta(q, a)$ زیرمجموعه‌ای از این مجموعه

است. برای مثال:

$$\delta: Q \times T \rightarrow 2^{Q \times T \times \{R, L\}}$$

$$\delta(q_0, a) = \{ (q_1, b, R), (q_2, c, L) \}$$

در هر دو ماشین انتقال λ می‌تواند وجود داشته باشد.

توجه: توان محاسباتی تورینگ غیر قطعی = توان محاسباتی تورینگ قطعی

هر ماشین تورینگ غیر قطعی دارای ماشین تورینگ معادل قطعی است.

توان محاسباتی ماشین تورینگ، مجموعه همه زبان های دارای گرامر است.

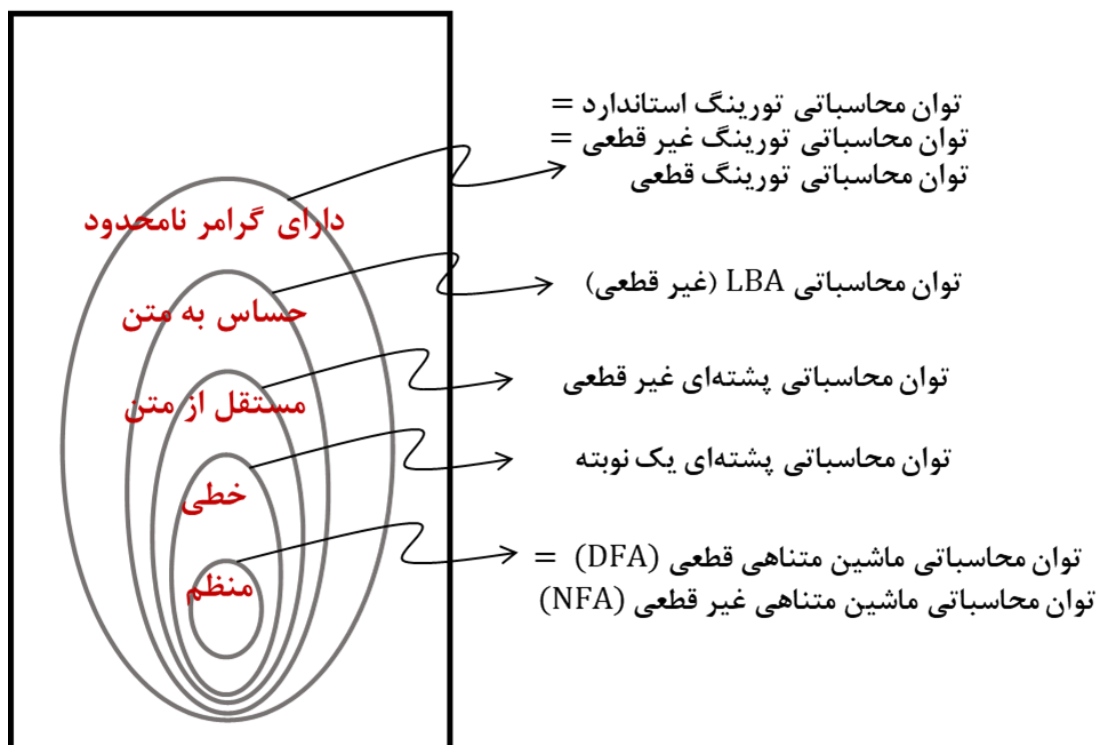
آیا توان LBA های غیر قطعی و LBA قطعی یکسان است؟

آیا ماشین های تورینگ کران دار خطی غیرقطعی نیز دارای ماشین تورینگ کران دار خطی قطعی معادل است؟ این یک مساله حل نشده است. وقتی از ماشین تورینگ LBA صحبت می کنیم به صورت پیش فرض منظورمان غیرقطعی است. فرض کنیم M_1 یک ماشین تورینگ غیرقطعی از نوع LBA باشد. M_1 حتما دارای یک ماشین تورینگ قطعی معادل به نام M_2 است، اما معلوم نیست که M_2 از نوع LBA باشد.

توان محاسباتی ماشین‌ها در یک نگاه:

مجموعه همه زبان‌ها ناشماراست، اما مجموعه همه ماشین‌های تورینگ شماراست.
پس همه زبان‌ها دارای ماشین تورینگ پذیرنده نیستند. [البته اگر اجازه داشته باشیم تعداد حالات را نامتناهی کنیم، هر زبان دارای ماشین پذیرنده است]

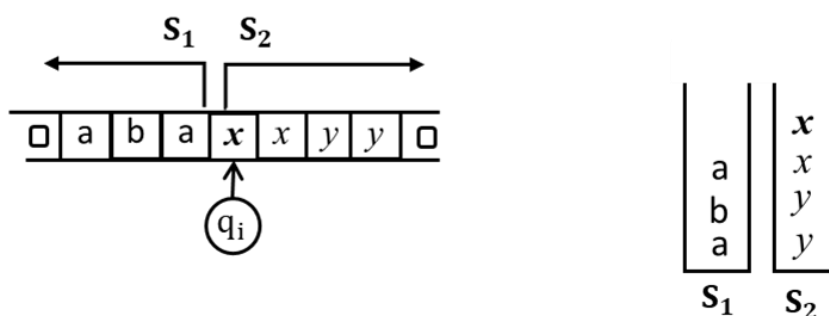
مجموعه همه زبان‌ها



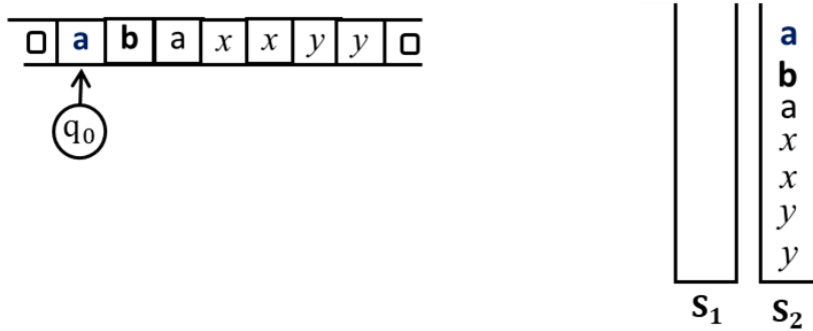
ماشین دو پشته‌ای (یا بیشتر) ، با تورینگ استاندارد معادل است.

توان محاسباتی ماشین های پشته‌ای مجموعه زبان های مستقل از متن است، پس توان آنها اکیدا کمتر از تورینگ استاندارد است.

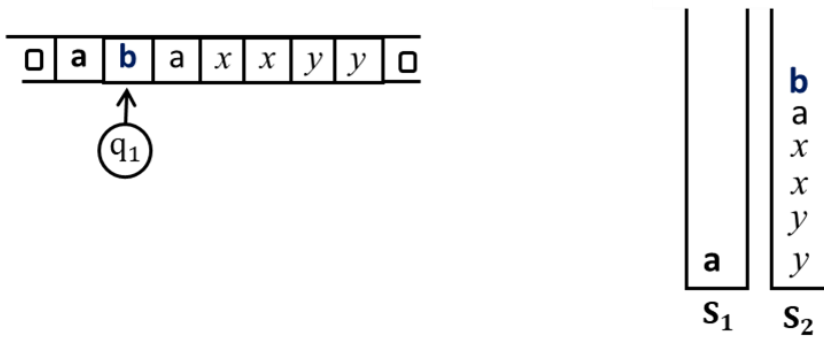
اما با مجهز شدن به دو پشته (یا بیشتر) ماشین پشته ای می‌تواند تمام حرکات ماشین تورینگ استاندارد را شبیه سازی کند و توان محاسباتی آنها با هم برابر می‌شود. مطابق شکل ، محتوای نوار با توجه به مکان هد، در دو پشته قابل نمایش است.



در شروع کار همه نمادهای روی نوار در پشته S_2 قرار دارند. سلولی که هد روی آن قرار دارد بالای پشته S_2 است.



با هر انتقال، نمادهای واقع در سمت چپ هد، در پشته S_1 و نمادهای واقع در همان سلول و سمت راست آن در S_2 نمایش داده می‌شوند. با تغییر حالت ماشین تورینگ، ماشین پشته‌ای هم تغییر حالت می‌دهد.



فصل نهم: ماشین تورینگ
جلسه نوزدهم بخش دو: انواع دیگر ماشین تورینگ

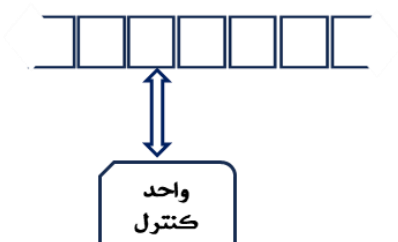
انواع دیگر ماشین تورینگ:

با افزودن قابلیت ها و امکانات یا محدود کردن و حذف قابلیت ها، می توان انواع مختلفی از ماشین تورینگ ایجاد کرد که توان محاسباتی اغلب آنها با تورینگ استاندارد برابر است و در برخی موارد، محدودیت ها باعث کاهش توان محاسبه می شود. انواع معروف تر را بررسی می کنیم.

یادآوری ماشین تورینگ استاندارد

در تورینگ استاندارد: نوار از هر دو سمت نامحدود است، قابلیت خواندن و نوشتن در همه سلول های نوار را دارد. پس فضای نوشتنی نامتناهی است.

حرکت هد در دو جهت چپ یا راست امکان پذیر است: (Left, Right).
توان محاسباتی آن مجموعه همه زبان های دارای گرامر بدون محدودیت است.



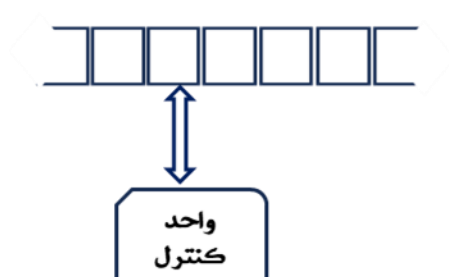
$$\delta: Q \times T \rightarrow Q \times T \times \{R, L\}$$

ماشین تورینگ با امکان توقف هد ماشین:

نوار از هر دو سمت نامحدود است.

قابلیت خواندن و نوشتن در همه سلول های نوار را دارد.

در هر انتقال، هد می تواند به چپ یا راست حرکت کند یا بی حرکت بماند.



$$\delta: Q \times T \rightarrow Q \times T \times \{R, L, S\}$$

(Left, Right, Stop)

امکان توقف هد، چیزی به توانایی ماشین تورینگ اضافه نمی کند چون تورینگ استاندارد

هم می تواند با انتقال به چپ و راست، حرکت توقف را شبیه سازی کند:

انتقال همراه با توقف هد مانند:

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

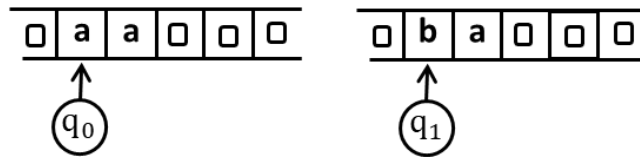
را می توان با دو انتقال زیر شبیه سازی کرد:

$$\delta(q_0, a) = (q'_1, b, R)$$

$$\delta(q'_1, a) = (q_1, a, L)$$

انتقال با توقف هد

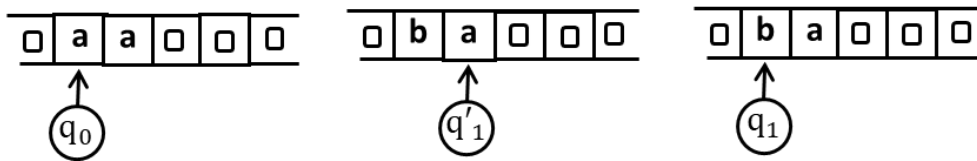
$$\delta(q_0, a) = (q_1, b, S)$$



همان انتقال بدون توقف هد

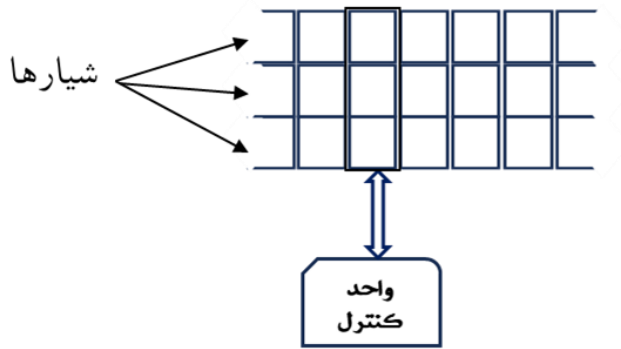
$$\delta(q_0, a) = (q'_1, b, R)$$

$$\delta(q'_1, a) = (q_1, a, L)$$



ماشین تورینگ چند شیاره:

نوار حافظه در این ماشین از m شیار تشکیل شده است .
در هر گام، هد به جای یک علامت ، m علامت را از نوار می خواند.
توان محاسباتی این ماشین با تورینگ استاندارد برابر است.

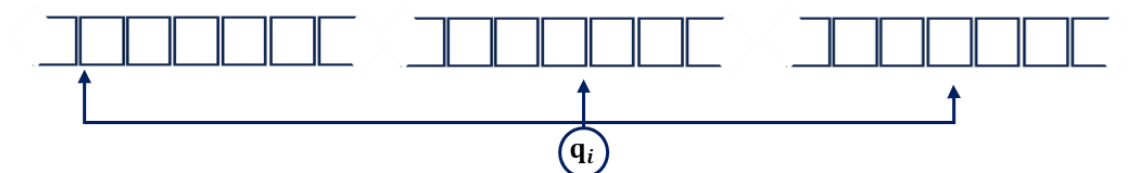


$$\delta: Q \times T^m \rightarrow Q \times T^m \times \{R, L\}$$

ماشین تورینگ با چند نوار مستقل:

در این ماشین به جای یک نوار از m نوار حافظه استفاده می‌شود که روی هر کدام یک هد می‌تواند مستقل از سایر هدها به چپ یا راست حرکت کند.

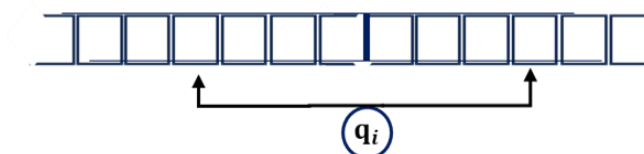
توان محاسباتی این ماشین با تورینگ استاندارد برابر است.



$$\delta: Q \times T^m \rightarrow Q \times T^m \times \{R, L\}^m$$

ماشین تورینگ با یک نوار اما چند هد مستقل:

نوار حافظه این ماشین، معمولی است اما روی همین نوار k هد به طور مستقل در حال کار هستند. هر هد می‌تواند امکان توقف هم داشته باشد. توان محاسباتی این ماشین با تورینگ استاندارد برابر است.

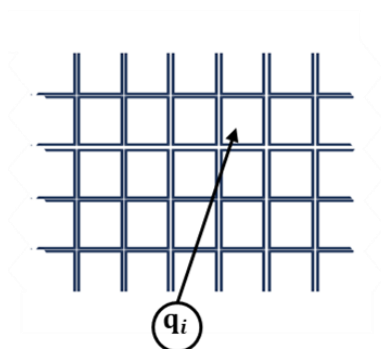


$$\delta: Q \times T^k \rightarrow Q \times T^k \times \{R, L, S\}^k$$

ماشین تورینگ چند بعدی:

در این ماشین به جای نوار حافظه یک بعدی، از دیسک حافظه دوبعدی یا شبکه سه بعدی استفاده می‌شود. در هر سلول، هد می‌تواند بخواند و بنویسد. تعداد جهت‌های حرکت هد بیشتر است. مثلاً در شبکه دو بعدی چهار جهت (و احتمالاً امکان توقف) را داریم.

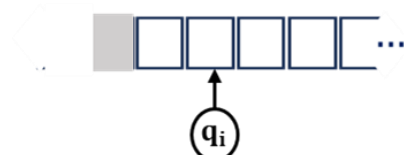
توان محاسباتی این ماشین با تورینگ استاندارد برابر است.



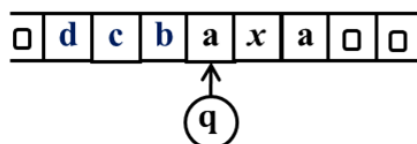
$$\delta: Q \times T \rightarrow Q \times T \times \{R, L, S, U, D\}$$

ماشین تورینگ با نوار نیمه متناهی:

نوار حافظه در این ماشین از یک طرف متناهی و از یک طرف نامتناهی است.



با استفاده از یک نوار نیمه متناهی دوشیاره میتوان همه حرکات ماشین تورینگ استاندارد را با استفاده از نوار نیمه متناهی شبیه سازی کرد. در نتیجه: توان محاسباتی این ماشین با تورینگ استاندارد برابر است.

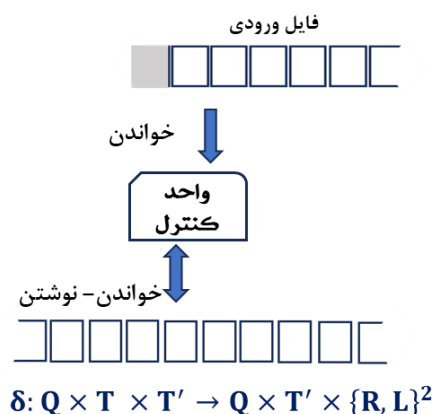


#	a	x	a			
#	b	c	d			

ماشین تورینگ برون-خط (off-line):

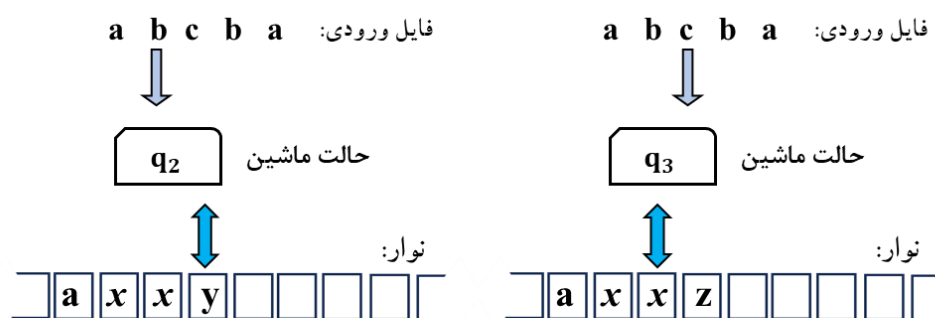
تورینگ استاندارد فایل ورودی ندارد و فقط یک نوار خواندنی-نوشتنی دارد که از ابتدا رشته روی آن درج شده و همه محاسبات روی همان نوار انجام می‌شود. اما تورینگ برون-خط علاوه بر این نوار خواندنی نوشتنی، یک نوار ورودی (نیمه متناهی) دارد که فایل ورودی را از آنجا می‌خواند. یک هد فقط خواندنی اطلاعات نوار ورودی را می‌خواند، هد خواندنی نوشتنی هم روی نوار اصلی محاسبات را انجام می‌دهد. نتیجه هر انتقال بستگی به ۳ چیز دارد: واحد کنترل در کدام حالت q_i باشد، هد خواندنی از فایل ورودی چه علامتی را خوانده باشد، و هد خواندنی نوشتنی روی نوار چه علامتی را خوانده باشد؟

ماشین برون خط را می‌توان با ماشین تورینگ چهار شیاره شبیه سازی کرد. پس: توان محاسباتی این ماشین با تورینگ استاندارد برابر است.



یک مثال:

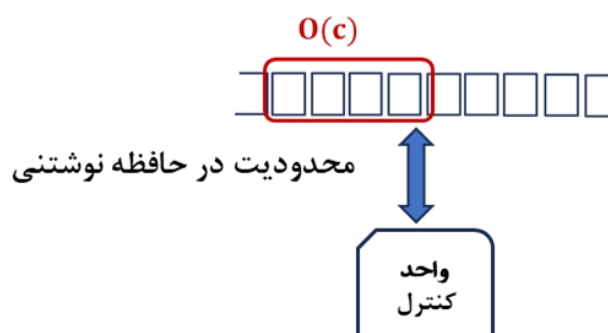
توضیح انتقال $\delta(q_2, b, y) = (q_3, z, R, L)$ در ماشین تورینگ آفلاین:
 وقتی ماشین در حالت q_2 است و از فایل ورودی الفبای b و از نوار جافظه،
 الفبای y را بخواند این اقدامات را انجام می‌دهد: به حالت q_3 می‌رود، به
 جای y ، z را می‌نویسد و خواندنی (روی فایل ورودی) را به راست و هد
 خواندن - نوشتن (روی نوار) را به چپ می‌برد.



$$\delta(q_2, b, y) = (q_3, z, R, L)$$

ماشین تورینگ فقط خواندنی یا با محدودیت در حافظه نوشتنی:

اگر ماشین تورینگ دارای هد فقط خواندنی باشد، (ناتوان از نوشتن) همچنین اگر فقط در تعدادی متناهی از سلول‌های نوار، امکان نوشتن داشته باشد، توان محاسباتی ماشین، به اندازه ماشین متناهی است.

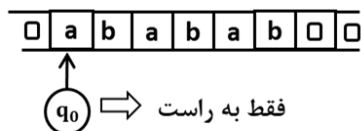


حافظه نوشتنی صفر یا از مرتبه $O(c)$ باشد $\Leftrightarrow$ توان ماشین با dfa معادل است $\Leftrightarrow$ زبان ماشین، منظم است.

حافظه نوشتنی از مرتبه $O(|w|)$ باشد $\Leftrightarrow$ ماشین تورینگ LBA است $\Leftrightarrow$ زبان ماشین، حساس به متن است.

ماشین تورینگ با محدودیت حرکت هد به یک سمت:

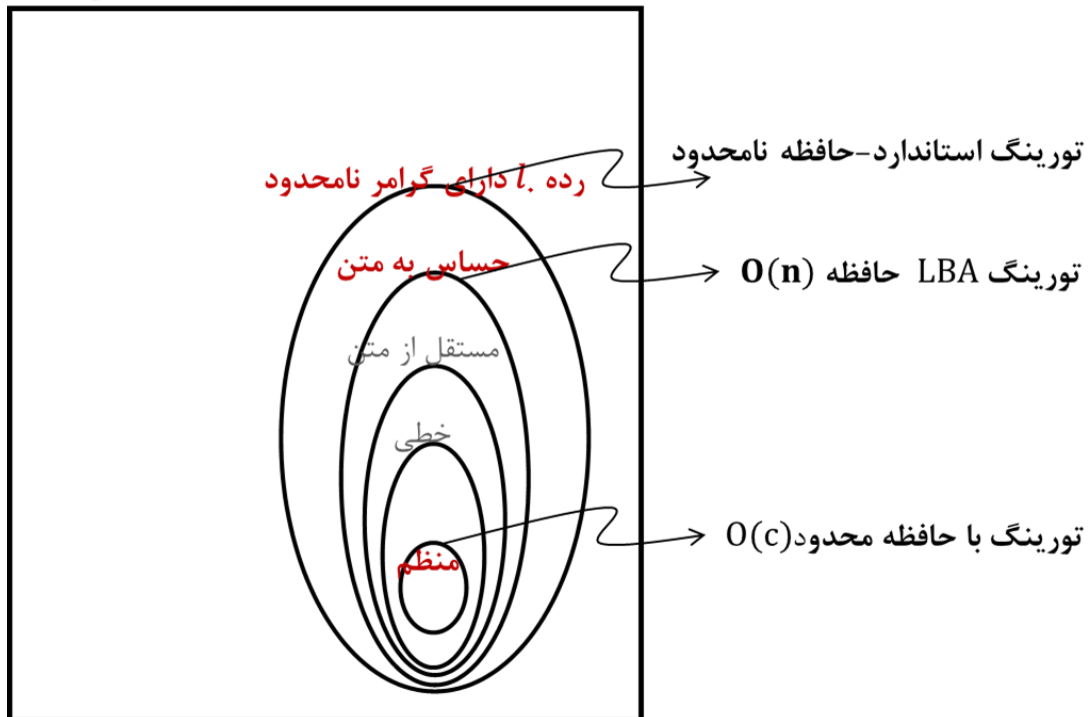
اگر حرکت هد فقط در یک جهت ممکن باشد (در تورینگ استاندارد که هد روی اولین الفبا قرار دارد، حرکت هد فقط به سمت راست مجاز باشد) آنگاه ماشین حتی در صورت یادداشت کردن چیزی، به چیزی که نوشته است دسترسی ندارد پس مانند آن است که حافظه نوشتنی محدود شده باشد، چنین ماشینی توان محاسباتی ماشین متناهی را دارد.



حرکت هد فقط به راست $\Leftarrow$ توان ماشین با dfa معادل است $\Leftarrow$ زبان ماشین، منظم است.

تأثیر مرتبه حافظه بر توان محاسبه تورینگ :

مجموعه همه زبان‌ها



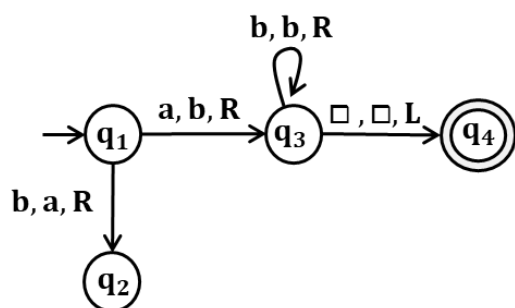
فصل نهم: ماشین تورینگ

جلسه بیستم بخش یک:

تورینگ جهانی (کد ماشین تورینگ) - روال شمارش

تورینگ جهانی: (A Universal Turing Machine)

شیوه ای برای کد کردن ماشین تورینگ به صورت یک رشته باینری وجود دارد. برای شروع، می‌دانیم ماشین تورینگ استاندارد را می‌توان طوری به شکل معادل طراحی کرد که فقط یک حالت نهایی (پذیرش) داشته باشد. برای مثال ماشین تورینگ مقابل را در نظر بگیرید:



۱- ابتدا حالات ماشین و مجموعه الفبا و نمادهای نوار را کد می‌کنیم:

این ماشین چهار حالت دارد. آنها را q_1, q_2, q_3, q_4 بنامیم. همیشه آخرین حالت، حالت پذیرش باشد.

به ترتیب آنها را با 0,00,000,0000 کد می‌کنیم:

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

$$\begin{array}{cccc} \downarrow & \downarrow & \downarrow & \downarrow \\ 0, & 00, & 000, & 0000 \end{array}$$

الفبا و سایر علامت‌های نوار را نیز به همین صورت کد کنیم. می‌توان فرض کرد blank آخرین نماد است. (B همان blank است.)

۲- ابتدا و انتهای کد، با 111 مشخص می‌شود.

111.....111

۳- برای جداکردن پیام‌ها از 11 استفاده می‌کنیم.

111.....11.....11.....11.....11.....111

۴- اولین پیام، تعداد حالات ماشین است. در این مثال اولین پیام 0000 است.

۵- دومین پیام، تعداد الفبا و نمادهای ماشین است. در این مثال دومین پیام 000 است.

111 0000 11 000 11.....11.....11...111

|Q| |Γ|

۶- در ادامه پیام انتقال‌های ماشین را کدگذاری می‌کنیم.

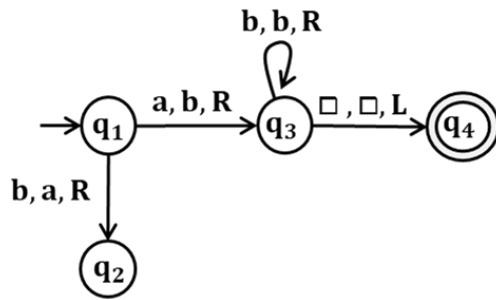
در مورد حالت‌ها و نمادها کدها را مشخص کردیم. حرکت به چپ و راست را نیز به این شکل کد می‌کنیم:

L, R
↓ ↓
0, 00

حالا می‌توانیم هر انتقال را به صورت یک پنج تایی نمایش دهیم.
مثلا انتقال $\delta(q_1, a) = (q_2, b, R)$ به این شکل رمز می‌شود:

$\delta(q_1, a) = (q_3, b, R)$
↓ ↓ ↓ ↓ ↓
11 0 1 0 1 000 1 00 1 00 11

نتیجه: این ماشین را M بنامیم که کامل آن چنین است:



انتقال از q_1 به q_3 انتقال از q_1 به q_2 ۳ علامت ۴ حالت

$\langle M \rangle =$ **111** 0000 **11** 000 **11** 010010010100 **11** 0101000100100
11 0001001000100100 **11** 000100010000100010 **111**

انتقال از q_3 به q_4 انتقال از q_3 به q_3

مساله - کد دودویی ماشین تورینگ M داده شده است. الفبای ماشین $\{a, b, B\}$ است (B همان blank است). زبان $L(M)$ کدام است؟

$\langle M \rangle$

$= 111\ 000\ 11\ 000\ 11\ 01010101001\ 11\ 01001001001001\ 11$
 $00100010001000101\ 111$

$\{a^n b^n : n \geq 0\}$ - 2

$\{b a^n b : n \geq 0\}$ - 1

$\{b^n a : n \geq 0\}$ - 4

$\{a^n b : n \geq 0\}$ - 3

پاسخ:

با ترجمه ی کد، تعداد حالات، الفبا، و انتقالهای ماشین را تشخیص دهیم:

$\{q_1, q_2, q_3\} \quad \{a, b, B\} \quad \delta(q_1, a) = (q_1, a, R) \quad \delta(q_2, B) = (q_3, B, L)$

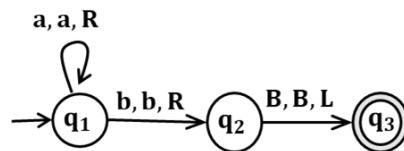
$\uparrow \quad \uparrow \quad \uparrow \quad \uparrow$

$111\ 000\ 11\ 000\ 11\ 01010101001\ 11\ 01001001001001\ 11$

$00100010001000101\ 111$

$\downarrow$

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



رشته هایی به توقف در حالت نهایی می‌رسند که به فرم $a^n b$ باشند:

$L(M) = \{a^n b : n \geq 0\}$

مساله - کد دودویی ماشین تورینگ M داده شده است. الفبای ماشین

$\{a, b, B\}$ است (B همان blank است). زبان $L(M)$ کدام است؟

$\langle M \rangle$

= 111 000 11 000 11 01010101001 11 01001001001001 11
00100010001000101 111

$\{a^n b^n : n \geq 0\}$ - 2

$\{ba^n b : n \geq 0\}$ - 1

$\{b^n a : n \geq 0\}$ - 4

$\{a^n b : n \geq 0\}$ - 3

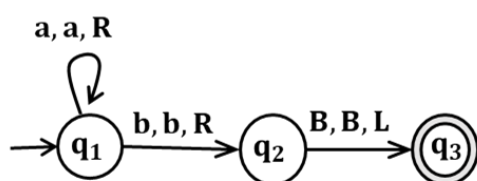
پاسخ: با ترجمه ی کد، تعداد حالات، الفبا، و انتقالهای ماشین را تشخیص دهیم:

$\{q_1, q_2, q_3\}$ $\{a, b, B\}$ $\delta(q_1, a) = (q_1, a, R)$ $\delta(q_2, B) = (q_3, B, L)$

↑ ↑ ↑ ↑
111 000 11 000 11 01010101001 11 01001001001001 11

00100010001000101 111

↓
 $\delta(q_1, b) = (q_2, b, R)$

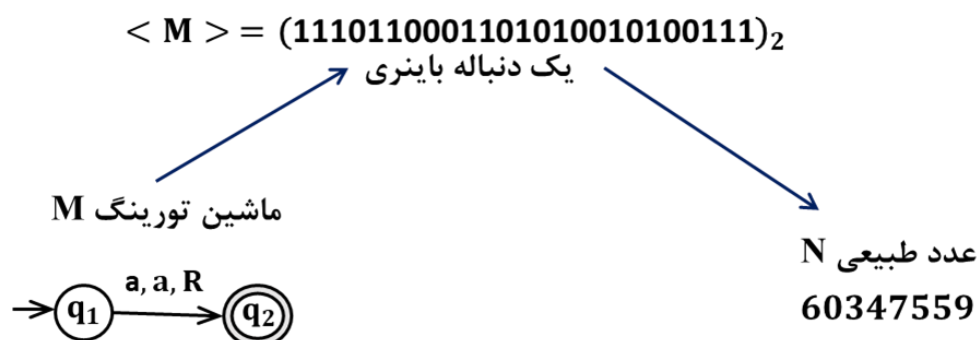


رشته هایی به توقف در حالت نهایی می‌رسند که به فرم $a^n b$ باشند:

$L(M) = \{a^n b : n \geq 0\}$

نتیجه مهم: شمارا بودن ماشین‌های تورینگ

هر ماشین تورینگ استاندارد را به شکل یکتا به یک رشته باینری تبدیل کردیم. هر رشته باینری اگر به مبنای ۱۰ بیاید یک عدد طبیعی را نشان می‌دهد. پس تابعی یک به یک از ماشین‌های تورینگ به مجموعه اعداد طبیعی داریم. در نتیجه تعداد ماشین‌های تورینگ غیرهم ارز، شمارای نامتناهی است.



نکته: نگاشت مورد اشاره که هر ماشین تورینگ را به یک عدد طبیعی نظیر می‌کند یک به یک است اما پوشا نیست. نظیر هر ماشین تورینگ، یک عدد طبیعی وجود دارد. اما نظیر هر عدد طبیعی، لزوماً یک ماشین تورینگ وجود ندارد. اعداد طبیعی زیادی مانند عدد طبیعی ۶ وجود دارند که وقتی در مبنای ۲ نوشته شوند دارای سه رقم ۱ در ابتدا و انتها نیستند پس نمی‌توانند کد ماشین تورینگ باشند.

روال شمارش:

فرض کنیم L مجموعه ای از رشته‌ها باشد که شمارا است پس متناهی یا شمارای نامتناهی است. عناصر هر مجموعه متناهی را می‌توان در یک صف قرار داد:

$$w_1, w_2, \dots, w_k$$

به طور مشابه برای مجموعه های شمارای نامتناهی تابع یک به یک و پوشای

$f: \mathbb{N} \rightarrow L$ وجود دارد که عناصر L را در یک دنباله قرار می‌دهد:

$$f(1) = w_1, f(2) = w_2, f(3) = w_3, f(4) = w_4, \dots$$

اگر یک ماشین تورینگ M بتواند تمام رشته های $w_1, w_2, w_3, w_4, \dots$

را به ترتیب روی نوار به وجود بیاورد می‌گوییم آن ماشین یک روال شمارش برای L

است. روال شمارش یک الگوریتم نیست زیرا هیچ گاه به پایان نمی‌رسد.

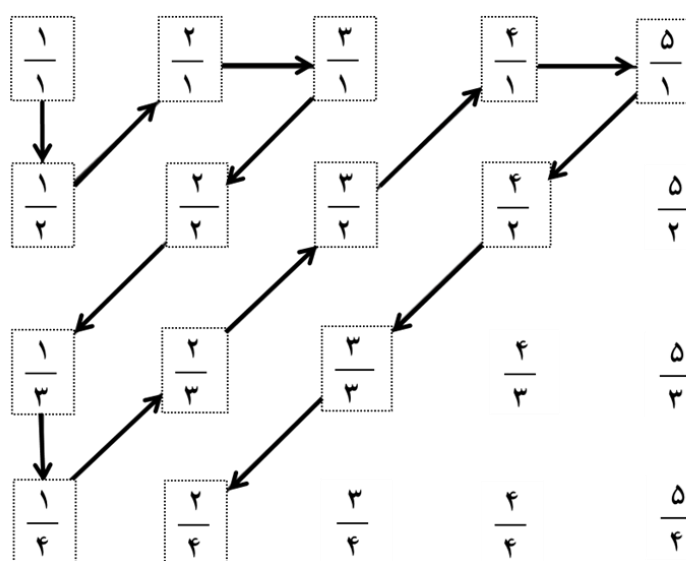
فصل نهم: ماشین تورینگ

جلسه بیستم بخش دوم:

روال شمارش - زبان‌های بازگشتی و بازگشتی شمارش‌پذیر

مثال:

اگر اعداد گویای مثبت ($\mathbb{Q}^+$) را به این ترتیب تولید کنیم یک روال شمارش برای آن خواهیم داشت:

$$\mathbb{Q}^+ = \left\{ \frac{m}{n} \mid n, m \in \mathbb{N} \right\}$$


مثال: اگر رشته های عضو زبان $L = \{a^n b : n \geq 0\}$ را به این ترتیب تولید کنیم، یک روال شمارش برای آن خواهیم داشت:

$$f(1) = b$$

$$f(2) = a^1 b$$

$$f(3) = a^2 b$$

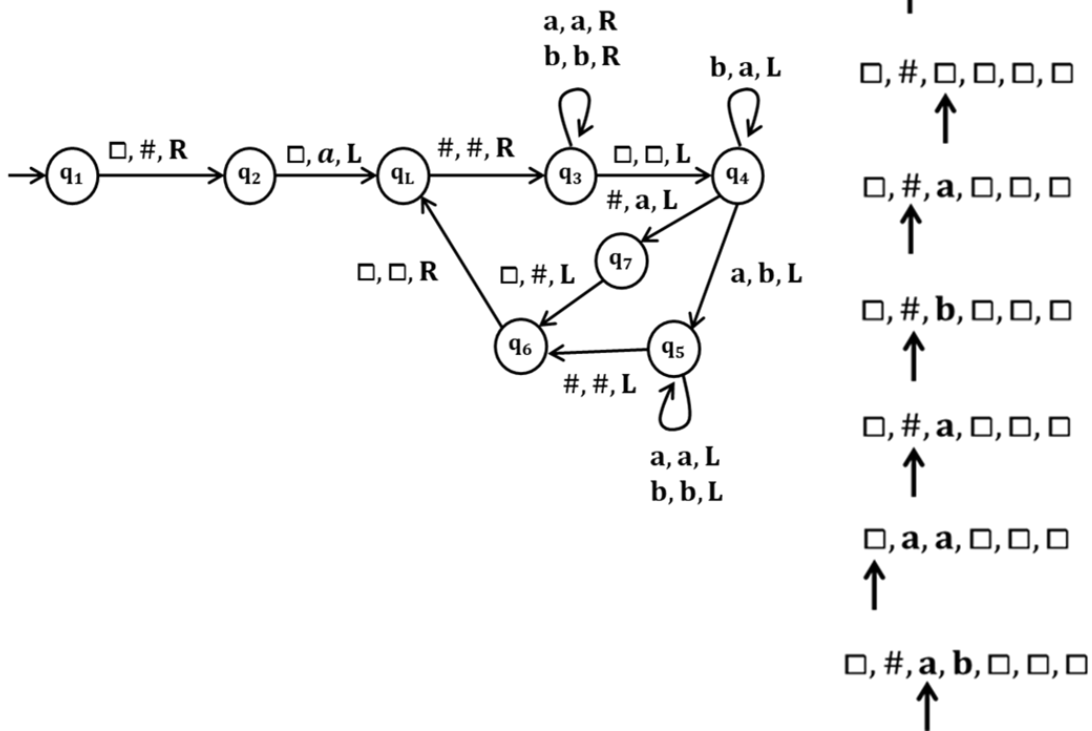
$$f(4) = a^3 b$$

...

مثال: ماشین تورینگ زیر یک روال شمارش برای زبان $\Sigma^+ = \{a,b\}^+$ ارائه می‌دهد.

ابتدا همه رشته‌های به طول یک، سپس همه رشته‌های به طول ۲، و ... به همین ترتیب به تولید رشته‌ها ادامه می‌دهد. این روال، هیچگاه متوقف نمی‌شود پس یک الگوریتم نیست اما تمام رشته‌های عضو Σ^+ را یکی یکی تولید می‌کند.

$$\Sigma^+ = \{ \mathbf{a}, \mathbf{b}, \mathbf{a^2}, \mathbf{ab}, \mathbf{b^2}, \mathbf{a^3}, \mathbf{a^2b}, \mathbf{ab^2}, \mathbf{b^3}, \dots \}$$



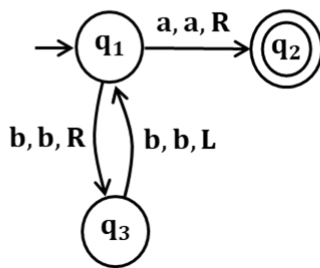
زبان‌های بازگشتی و بازگشتی - شمارش پذیر:

فرض کنیم M ماشین تورینگ پذیرنده زبان $L = L(M)$ باشد. دیدیم که برای هر رشته دلخواه w یکی از این حالات رخ می‌دهد:

الف: اگر $w \in L(M)$ باشد، توقف در حالت نهایی (پذیرش) رخ می‌دهد.

ب: اگر $w \notin L(M)$ یا توقف در حالت غیرنهایی یا افتادن در حلقه نامتناهی رخ می‌دهد.

مثال: این ماشین تورینگ پذیرنده زبان $L(M) = a(a + b)^*$ است. به رفتار آن به ازای سه ورودی مختلف توجه کنید:



$$L(M) = a(a + b)^* = \{ax \mid x \in \Sigma^*\}$$

$w = ab^2 \in L(M)$ $\Leftarrow$ توقف در حالت نهایی q_2

$w = ba \notin L(M)$ $\Leftarrow$ توقف در حالت غیرنهایی q_3

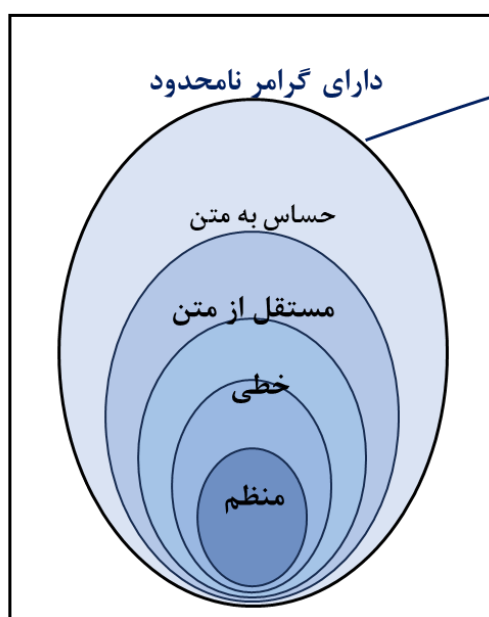
$w = b^2 \notin L(M)$ $\Leftarrow$ در حلقه گرفتار می‌شود

زبان‌های بازگشتی شمارش پذیر (R.E) : (تشخیص پذیر)

هر زبانی که دارای گرامر باشد، دارای ماشین تورینگ پذیرنده است. پس الگوریتمی وجود دارد که برای هر رشته‌ی $w \in L(M)$ ، عضویت w را تشخیص داده و متوقف می‌شود اما اگر $w \notin L(M)$ ممکن است متوقف نشود یعنی عدم عضویت ممکن است تشخیص داده نشود. این زبان‌ها را تشخیص پذیر می‌نامیم.

مجموعه‌های زیر با هم معادلند:

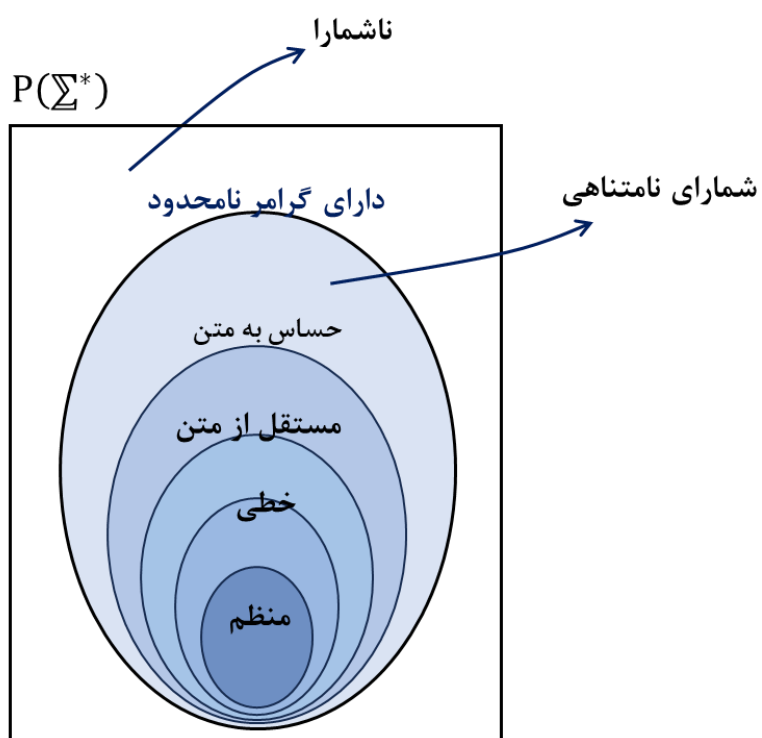
$P(\Sigma^*)$



رده l_0
 زبان‌های دارای گرامر نامحدود
 زبان‌های دارای تورینگ پذیرنده
 زبان‌های شمارش پذیر بازگشتی (R.E)
 زبان‌های تشخیص پذیر

- هر زبان رده l_0 دارای گرامر است.
- هر زبان دارای گرامر دارای ماشین تورینگ پذیرنده است.
- همه ماشین‌های تورینگ، تشخیص پذیر هستند.

رده l_0 شمارای نامتناهی است اما مکمل آن ناشماراست.

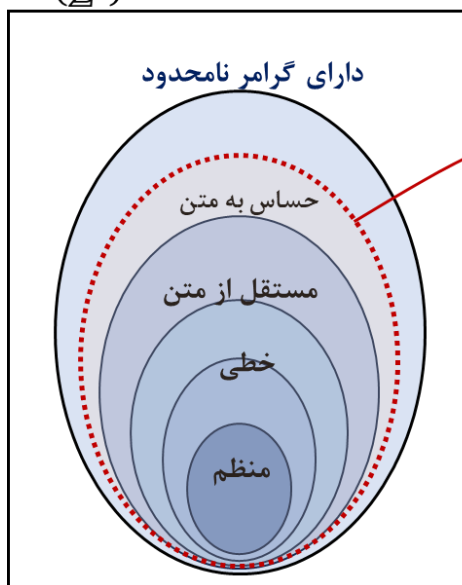


زبان‌های بازگشتی (R) (تصمیم‌پذیر) :

برخی از زبان‌های تشخیص‌پذیر، دارای ماشین تورینگ پذیرنده‌ای هستند که برای همه رشته‌های ورودی متوقف می‌شود و هیچ‌گاه در حلقه گرفتار نمی‌شود. برای هر رشته ورودی w که $w \in L(M)$ ، در حالت نهایی متوقف می‌شود و اگر $w \notin L(M)$ در حالت غیر نهایی متوقف می‌شود. چنین زبان‌هایی را بازگشتی (تصمیم‌پذیر) می‌نامیم.

به بیان دیگر: زبان بازگشتی زبانی است که دارای الگوریتم عضویت باشد، الگوریتمی که هم برای $w \in L$ و هم برای $w \notin L$ متوقف شود.

$P(\Sigma^*)$



→ Recursive language

there exists a Turing machine M that accepts L and that **halts on every w in Σ^+** .

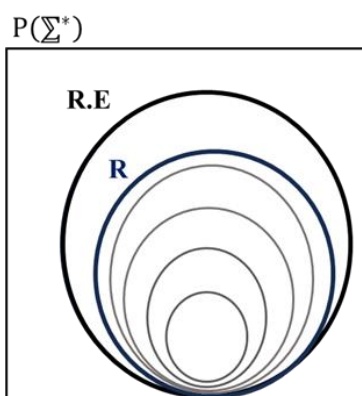
جمع‌بندی:

۱- مجموعه همه زبان‌ها ناشماراست. اما یک خانواده‌ی شمارا از آنها دارای گرامر هستند. هر زبان دارای گرامر، ماشین تورینگ پذیرنده دارد. این خانواده از زبان‌ها بازگشتی شمارش پذیر (R.E) (تشخیص پذیر) نامیده می‌شوند. برای رشته‌هایی که عضو زبان باشند، الگوریتمی وجود دارد که این عضویت را تشخیص می‌دهد. اما برای رشته‌هایی که عضو زبان نیستند، الگوریتمی که این عدم عضویت را تشخیص دهد، ممکن است وجود نداشته باشد.

۲- برخی از زبان‌ها دارای ماشین تورینگ پذیرنده ای هستند که همواره متوقف می‌شود و می‌تواند برای عضویت یا عدم عضویت هر رشته ورودی در آن زبان تصمیم بگیرد. این نوع ماشین‌های تورینگ را تصمیم گیرنده گوییم. این نوع زبان‌ها تصمیم پذیر و بازگشتی (R) نامیده می‌شوند.

۳- همه زبان‌های حساس به متن بازگشتی هستند. همه زبان‌های بازگشتی، بازگشتی شمارش پذیرند.

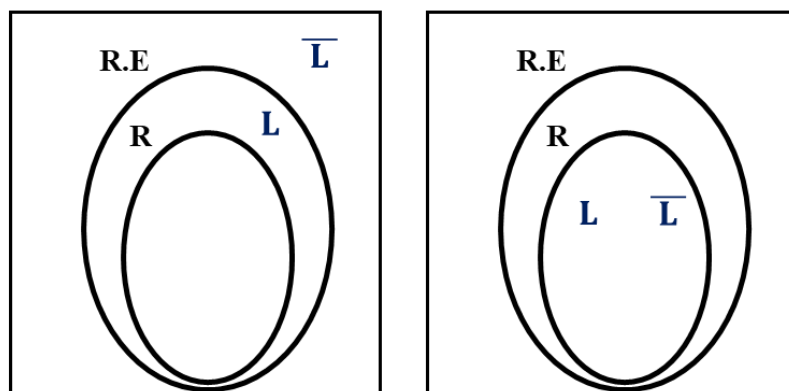
۴- همه این روابط به شکل اکید برقرار هستند:



$$\text{خانواده حساس به متن ها} \quad R \subset R.E$$

مقدمه جلسه بعدی:

خانواده زبان‌های بازگشتی شمارش‌پذیر، تحت مکمل بسته نیست.
 زبانی وجود دارد که خودش در رده R.E قرار دارد اما مکمل آن در R.E نیست.



خانواده زبان‌های بازگشتی، تحت مکمل بسته است.
 زبانی که خودش در رده R باشد مکملش نیز در R است.

در جلسه بعد ویژگیهای بستاری این دو خانواده را مطالعه می‌کنیم.

-21-

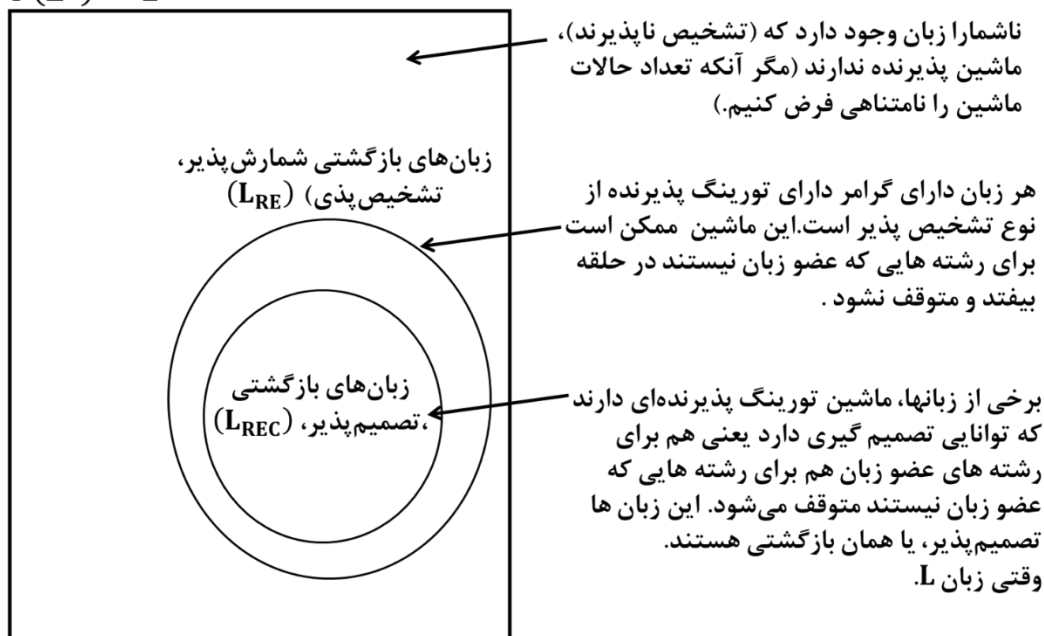
فصل نهم: ماشین تورینگ

جلسه بیستم و یکم بخش یک: زبان های بازگشتی و بازگشتی شمارش پذیر

- دسته بندی زبان های صوری

یادآوری :

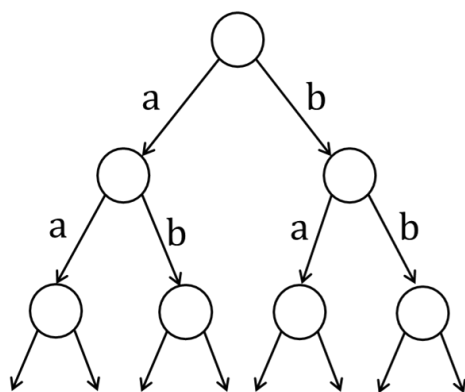
$$P(\Sigma^*) = 2^{\Sigma^*}$$



توضیح شکل:

طبق تعریف، در همه انواع ماشین‌ها، تعداد حالات ماشین متناهی است. اما اگر تعداد حالات ماشین (مثلاً ماشین DFA)

را بتوان نامتناهی گرفت، آنگاه همه زبان‌ها دارای ماشین پذیرنده می‌شوند. کافیت در شکل زیر حالاتی که با رشته‌های عضو زبان به آنها می‌رسیم را حالت پذیرش فرض کنیم.



مطالعه چند ویژگی زبان های بازگشتی شمارش پذیر، و زبان های بازگشتی:

درباره زبان های بازگشتی شمارش پذیر گفتیم که:

توان محاسباتی گرامرها = توان محاسباتی ماشین های تورینگ = خانواده زبان های بازگشتی شمارش پذیر = خانواده زبان های تشخیص پذیر = رده L_0

و در جلسات قبل دیده ایم که:

اگر L_1 و L_2 هر کدام دارای یک گرامر باشند آنگاه $L_1 \cup L_2$ و $L_1 \cap L_2$ و $L_1 L_2$ و $h(L_1)$ نیز دارای گرامر خواهند بود.

به عبارتی: خانواده R.E نسبت به اجتماع و اشتراک و الحاق و همریختی بسته است.

اما اکنون به این مساله می پردازیم:

سوال: آیا مکمل یک زبان تشخیص پذیر، تشخیص پذیر است؟

پاسخ: خیر، ممکن است L تشخیص پذیر باشد اما $\bar{L}$ تشخیص پذیر نباشد.

یک مثال معروف برای این مساله وجود دارد که مطالعه می کنیم.

زبانی وجود دارد که تشخیص پذیر است اما مکمل آن، تشخیص پذیر نیست.

فرض کنیم مجموعه الفبا، تک عضوی باشد

$$\Sigma = \{a\}$$

مجموعه ماشین های تورینگ، شمارای نامتناهی است.

پس همه ماشین های تورینگ را می توان با اعداد $\{0,1,2,3, \dots\}$ اندیس گذاری کرد:

$$\mathbb{T} = \{\mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2, \dots\}$$

تمام رشته های موجود در Σ^* عبارتند از :

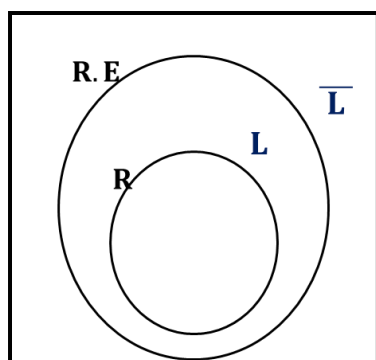
$$\Sigma^* = \{\lambda, a, a^2, a^3, \dots\}$$

زبان L و مکمل آن $\overline{L}$ را به این صورت معرفی می کنیم:

$$L = \{a^i \mid a^i \in L(\mathcal{M}_i)\}$$

$$\overline{L} = \{a^i \mid a^i \notin L(\mathcal{M}_i)\}$$

مثلا رشته a^5 را به ماشین تورینگ $\mathcal{M}_5$ می دهیم اگر پذیرفته شود عضو L است و در غیر اینصورت عضو $\overline{L}$ است. پس هر رشته ای که عضو L باشد توسط ماشین تورینگ مرتبط با آن پذیرفته خواهد شد یعنی عضویت آن حتما تشخیص داده می شود. به عبارتی زبان L تشخیص پذیر است.



حالا ثابت می کنیم $\overline{L}$ تشخیص پذیر نیست.

ثابت می‌کنیم $\overline{L}$ تشخیص‌پذیر نیست.

برهان خلف: اگر $\overline{L}$ تشخیص‌پذیر باشد، (فرض خلف) دارای ماشین تورینگ پذیرنده خواهد بود. همه ماشین‌های تورینگ در مجموعه $\{\mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2, \dots\}$ قرار دارند. پس یکی از آنها مثلاً $\mathcal{M}_j$ پذیرنده $\overline{L}$ است:

$$\overline{L} \in R.E \Rightarrow \exists j \quad \overline{L} = L(\mathcal{M}_j)$$

حالا به رشته a^j توجه می‌کنیم. به صورت منطقی a^j یا توسط $\mathcal{M}_j$ پذیرفته می‌شود یا نمی‌شود. نشان می‌دهیم در هر دو حالت تناقض رخ می‌دهد.

$$a^j \in L(\mathcal{M}_j) \Rightarrow a^j \in L \Rightarrow a^j \notin \overline{L} \Rightarrow a^j \notin L(\mathcal{M}_j)$$

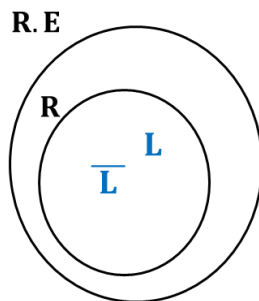
$$a^j \notin L(\mathcal{M}_j) \Rightarrow a^j \notin L \Rightarrow a^j \in \overline{L} \Rightarrow a^j \in L(\mathcal{M}_j)$$

پس در هر دو حالت تناقض رخ می‌دهد. در نتیجه $\overline{L}$ تشخیص‌پذیر نیست.

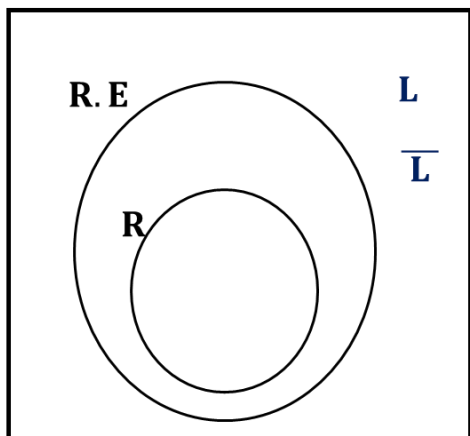
توجه: اگر L و $\overline{L}$ هر دو تشخیص‌پذیر باشند، آنگاه هر دو بازگشتی (تصمیم‌پذیر) هستند.

$$\left. \begin{array}{l} L \in R.E \\ \overline{L} \in R.E \end{array} \right\} \Rightarrow L \in R$$

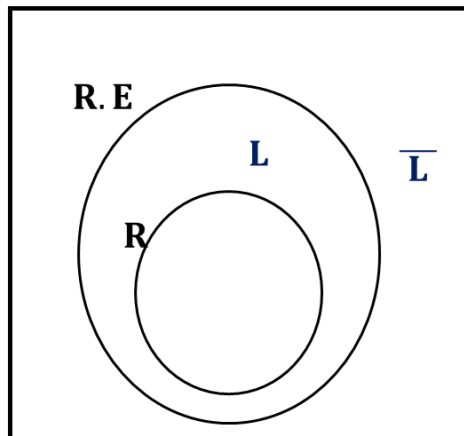
اگر عضویت در L و عضویت در $\overline{L}$ ، هر دو را بتوان تشخیص داد یعنی عضویت و عدم عضویت در L را می‌توان تشخیص داد. به عبارتی L بازگشتی است. وقتی L بازگشتی باشد $\overline{L}$ هم بازگشتی است.



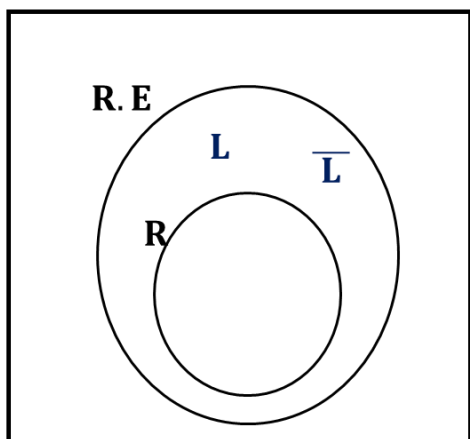
حالات ممکن برای L و $\overline{L}$



این وضعیت ممکن است. زبانی وجود دارد که خودش و مکملش هیچکدام تشخیص پذیر نیستند.

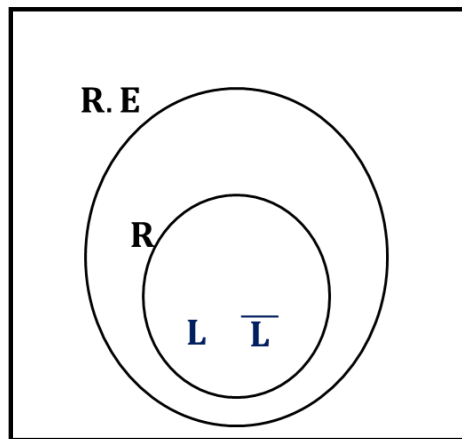


این وضعیت ممکن است. زبانی وجود دارد که تشخیص پذیر است اما مکملش تشخیص پذیر نیست. در این حالت می‌گوییم عضو $\overline{L}$ عضو $CO - RE$ است.



این حالت امکان ندارد مگر آنکه L و $\overline{L}$ هر دو عضو R باشند.

$\Rightarrow$



L و $\overline{L}$ هر کدام بازگشتی باشد دیگری هم بازگشتی است

خانواده‌ی $\text{CO} - \text{RE}$:

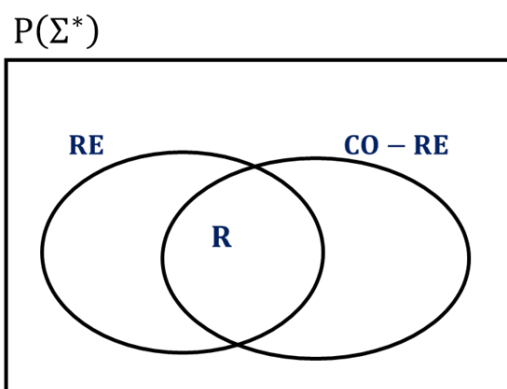
اگر مکمل یک زبان، تشخیص پذیر (R.E) باشد، خود آن زبان را عضو CO-RE می‌دانیم.

$$\bar{L} \in \text{RE} \Leftrightarrow L \in \text{CO} - \text{RE}$$

$$L \in \text{RE} \Leftrightarrow \bar{L} \in \text{CO} - \text{RE}$$

نتیجه:

اشتراک رده های RE و $\text{CO} - \text{RE}$ خانواده زبان های بازگشتی (R) است.



مسئله- (کارشناسی ارشد ۸۹)

مجموعه زبان های بازگشتی را با R و مجموعه زبان های بازگشتی شمارش پذیر را با RE نشان می دهیم. زبان L مفروض است. در کدام حالت زیر، ماشین تورینگ پذیرنده L وجود دارد که برای تمام رشته های ورودی به حالت توقف برسد؟

$$L \in RE, \bar{L} \notin R - ۲$$

$$L \in RE, \bar{L} \notin RE - ۱$$

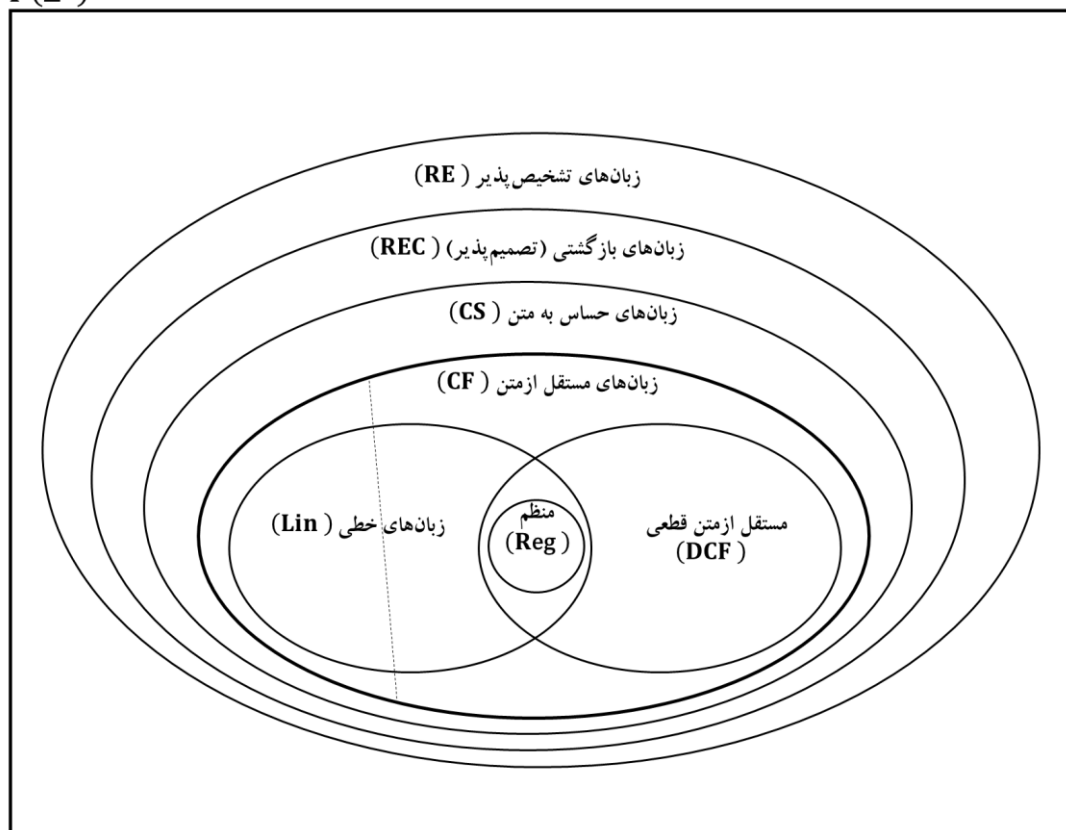
$$- ۴ - \text{هیچکدام}$$

$$L \in RE, \bar{L} \in RE - ۳$$

پاسخ: گزینه ۳

اگر L و $\bar{L}$ هر دو تشخیص پذیر (بازگشتی شمارش پذیر) باشند، آنگاه هر دو تصمیم پذیر (بازگشتی) هستند. یعنی دارای تورینگ پذیرنده از نوع تصمیم گیرنده هستند که برای هر رشته متوقف خواهد شد.

دسته بندی چامسکی برای زبان های صوری:

$P(\Sigma^*)$


ماشین پذیرنده	گرامر توصیف کننده	زبان
ماشین متناهی (FA)	گرامر منظم	زبان‌های منظم
ماشین پشته ای یک نوبته	گرامر خطی	زبان‌های خطی
ماشین پشته ای قطعی	گرامر LR(1)	زبان‌های مستقل از متن قطعی
ماشین پشته ای (PDA)	گرامر مستقل از متن	زبان‌های مستقل از متن
ماشین کران دار خطی (LBA)	گرامر حساس به متن	زبان‌های حساس به متن
ماشین تورینگ تصمیم گیرنده	----	زبان‌های بازگشتی
ماشین تورینگ تشخیص دهنده	گرامر بدون محدودیت	زبان‌های بازگشتی شمارش پذیر

فصل نهم: ماشین تورینگ

جلسه بیستم و یکم بخش دو: حل تست و چند نکته

علوم کامپیوتر - ۱۳۹۵

- ۱۱۷- فرض کنید $T(M, w)$ زمان اجرای پردازش ماشین M بر روی ورودی w است که برابر با تعداد گذرهای M در محاسبه w تعریف می‌شود. کدام گزینه صحیح است؟
- (۱) اگر M یک ماشین تورینگ باشد، آنگاه $T(M, w)$ متناهی است.
 - (۲) اگر M یک اتوماتون متناهی قطعی باشد، آنگاه $T(M, w)$ متناسب با $|w|$ است.
 - (۳) اگر M یک اتوماتون خطی محدود (LBA) باشد، آنگاه $T(M, w)$ متناسب با $|w|$ است.
 - (۴) اگر M یک اتوماتون پشته‌ای قطعی باشد، آنگاه $T(M, w)$ متناسب با بیشترین اندازه پشته هنگام پردازش w است.

پاسخ: گزینه ۲

گزینه ۲ صحیح است و سایر گزینه‌ها نادرست هستند:

در گزینه ۱ اگر رشته ورودی w پذیرفته نشود، ممکن است در حلقه نامتناهی گرفتار شود و $T(M, w)$ نامتناهی شود.

در گزینه ۲ ماشین متناهی dfa دقیقاً با n گام متوقف می‌شود که $n = |w|$.

در گزینه ۳ برای ماشین تورینگ LBA مرتبه حافظه مصرفی $S(n) \leq an + b = O(n)$ است. اما تعداد گامها (مرتبه زمانی) حداکثر نمایی است. رابطه بین مرتبه حافظه و مرتبه زمانی برای توقف تورینگ را در ادامه همین جلسه خواهیم گفت.

در گزینه ۴ نیز برای ماشین پشته‌ای قطعی، مرتبه زمانی به تعداد گامها مربوط است نه به اندازه پشته.

۶۴- اگر مجموعه زبان‌های پذیرفته شده به وسیله مدل‌های A کاملاً بزرگ‌تر از مجموعه زبان‌های پذیرفته شده به وسیله مدل‌های B باشد، می‌گوییم، مدل‌های محاسباتی A قوی‌تر از مدل‌های محاسباتی B هستند. اکنون فرض کنید یک $K-PDA$ نمایانگر یک خودکاره (اوتوماتون) پوش‌دان با k نوار پشته باشد. کدام یک از گزاره‌های زیر درست است؟

(۱) $1-PDA$ ها تنها مجموعه زبان‌های تصمیم‌پذیر را می‌پذیرند.

(۲) $2-PDA$ ها تنها مجموعه زبان‌های تصمیم‌پذیر را می‌پذیرند.

(۳) $3-PDA$ ها قوی‌تر از $2-PDA$ ها هستند.

(۴) $2-PDA$ ها قوی‌تر از $1-PDA$ ها هستند.

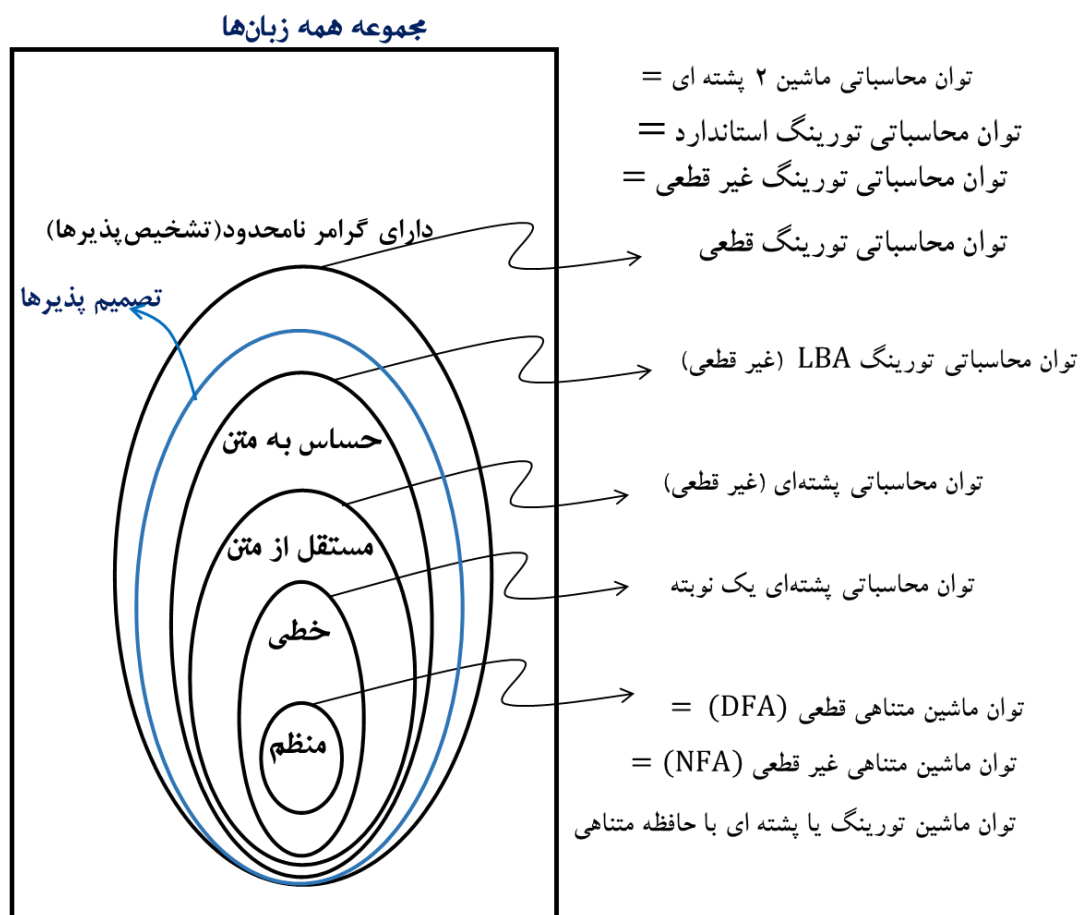
پاسخ: گزینه ۴

برای $K = 1$ توان محاسباتی $1-PDA$ ، خانواده زبان‌های مستقل از متن است که زیر مجموعه‌ای محض از تصمیم‌پذیرهاست.

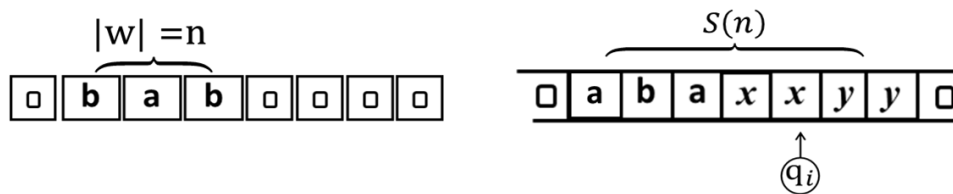
برای $K \geq 2$ توان محاسباتی $K-PDA$ همه زبان‌های تشخیص‌پذیر است که خانواده‌ای بزرگ‌تر از تصمیم‌پذیرها است.

در واقع برای $K \geq 2$ توان محاسباتی $K-PDA$ با ماشین تورینگ استاندارد برابر است.

توان محاسباتی ماشین‌ها در یک نگاه:



رابطه بین مرتبه حافظه مصرفی ماشین تورینگ با مرتبه زمانی (تعداد گامها برای توقف)



$$T(M, w) \leq k S(n) \Gamma^{S(n)}$$

فرض کنیم: تعداد حالات ماشین $|Q| = k$ و تعداد الفبای ماشین $|\Gamma|$ باشد، ماشین برای رشته ورودی w متوقف شود یعنی $T(M, w)$ متناهی باشد، طول رشته w برابر با $n = |w|$ باشد.

در این صورت اگر اندازه حافظه مصرفی $S(n)$ باشد. تعداد پیکربندی‌ها حداکثر $k S(n) |\Gamma|^{S(n)}$ است. در مسیر محاسبه این رشته، پیکربندی تکراری رخ نمی‌دهد زیرا ماشین در حلقه قرار نگرفته است. بنابراین تعداد گامها (مرتبه زمانی توقف) حداکثر به تعداد پیکربندی‌ها است:

نتیجه مهم:

اگر تعداد گامها در مسیر محاسبه w از $k S(n) \Gamma^{S(n)}$ بیشتر شود نتیجه می‌شود که ماشین متوقف نخواهد شد و $T(M, w) = \infty$ است.

$$T(M, w) > k S(n) \Gamma^{S(n)} \Rightarrow T(M, w) = \infty$$

ماشین T برای هر ورودی تعداد ثابتی پیکربندی اگر ماشین تورینگ قطعی داده شده T برای هر ورودی w با طول n فقط از n^{15} خانه از نوار حافظه استفاده نماید آن گاه: (علوم کامپیوتر - ۸۹)

(۱) ماشین T حتماً پس از یک تعداد متناهی مرحله (ثابت) برای هر ورودی داده شده متوقف می شود.

(۲) ماشین T حتماً در زمان متناهی (وابسته به طول ورودی) برای هر ورودی متوقف می شود.

(۳) تابع $f(x)$ وجود دارد که اگر T برای یک ورودی به طول n بیش از f مرحله کار کند آن گاه حتماً متوقف نمی شود.

(۴) ماشین T برای هر ورودی تعداد ثابتی پیکربندی (مستقل از طول ورودی) است.
(مستقل از طول ورودی) است.

پاسخ: گزینه ۳

فرض کنیم تابع $f(w)$ همان عبارت $\Gamma^{S(n)}$ باشد. پس گزینه ۳ درست است .

$$S(n) = n^{15} \begin{cases} \nearrow T(M, w) \leq k n^{15} \Gamma^{n^{15}} \\ \text{OR} \\ \searrow T(M, w) = \infty \end{cases}$$

۵۴- از میان چهار جمله زیر، چه تعداد از آن‌ها صحیح است؟

الف- اشتراک دو زبان بازگشتی، لزوماً یک زبان بازگشتی است.

ب- اگر $h(L)$ (تصویر همومورفیک L) منظم باشد می‌توان نتیجه گرفت خود L نیز منظم است.

ج- اجتماع دو زبان مستقل از متن قطعی، خود یک زبان مستقل از متن قطعی است.

د- زبان‌های شمارش‌پذیر بازگشتی تحت عملیات مکمل‌گیری بسته هستند.

۱ (۱)	۲ (۲)
۳ (۳)	۴ (۴)

پاسخ: گزینه ۱

فقط جمله الف درست است.

خانواده زبان‌های بازگشتی (رده R) نسبت به همه اعمال اجتماع، اشتراک، الحاق، مکمل، (به جز تصویر هم‌ریختی) بسته است.

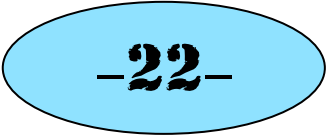
خانواده زبان‌های منظم نسبت به تصویر هم‌ریختی بسته است. اگر L منظم باشد آنگاه $h(L)$ منظم است. اما عکس این مطلب درست نیست. مثلاً اگر $L = \{a^n b^n : n \geq 1\}$ و هم‌ریختی h چنان باشد که $h(a) = a, h(b) = a$ آنگاه $h(L) = \{a^{2n} : n \geq 1\}$ منظم است اما L منظم نیست. اجتماع دو زبان مستقل از متن قطعی حتماً مستقل از متن است اما لزوماً قطعی نیست مثال نقض را قبلاً دیده ایم:

$$L_1 = \{a^n b^n : n \geq 1\}, L_2 = \{a^n b^{2n} : n \geq 1\}$$

زبان‌های بازگشتی شمارش‌پذیر (تشخیص‌پذیر - رده RE) تحت مکمل بسته نیستند. مثال را در همین جلسه دیدیم.

تز چرچ و تورینگ

هر محاسبه‌ای که بتواند توسط یک کامپیوتر یا توسط وسایل مکانیکی انجام شود، توسط یک ماشین تورینگ نیز قابل انجام است. در ضمن، این تز هنوز اثبات نشده است و همچنین هیچکس تاکنون قادر نبوده تا یک مسئله را طراحی کند که توسط یک الگوریتم قابل حل باشد، اما توسط یک ماشین تورینگ قابل پیاده‌سازی نباشد.



-22-

فصل نهم: ماشین تورینگ
جلسه بیست و دوم بخش یک: نکات تکمیلی - تورینگ محاسبه گر

نکات مربوط به ماشین‌های تورینگ

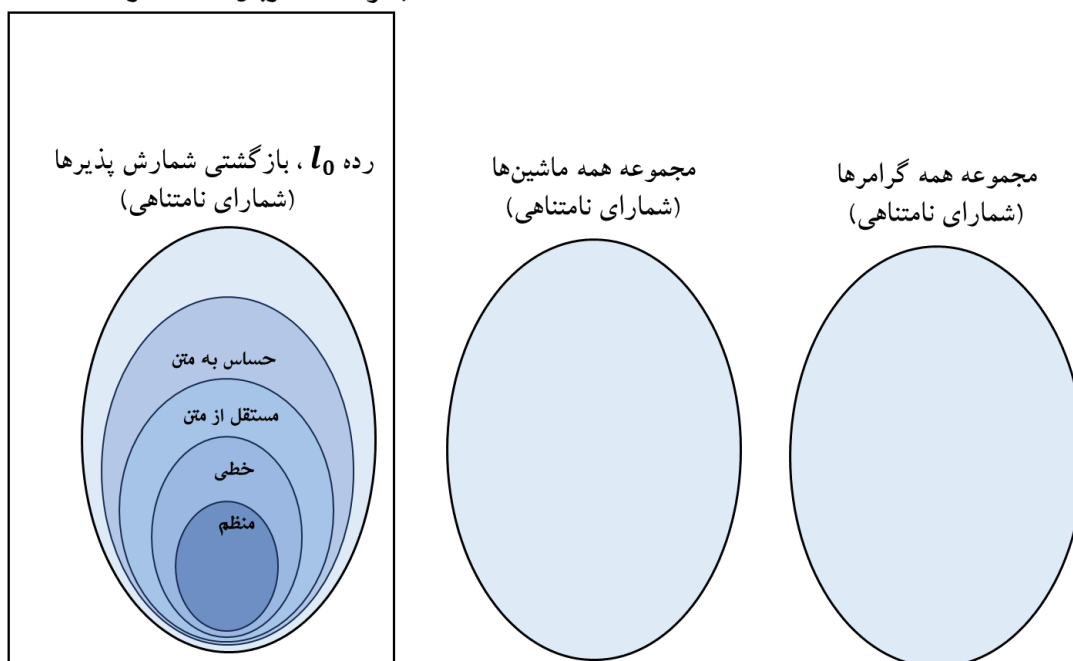
- (۱) اگر یک ماشین تورینگ با این محدودیت روبرو شود که هد آن فقط می‌تواند به یک سمت حرکت کند، آنگاه دیگر نوشتن روی آن مهم نیست و توان ماشین به توان یک ماشین متناهی کاهش می‌یابد.
- (۲) اگر ماشین تورینگ با این محدودیت روبرو شود که حافظه آن، عددی ثابت از سلول‌های نوار را در اختیار داشته باشد، آنگاه قدرت ماشین به قدرت یک ماشین متناهی کاهش می‌یابد. دقت کنید که این مطلب نباید با ماشین‌های LBA ، اشتباه گرفته شود. زیرا در ماشین‌های LBA ، طول نوار، محدود به طول رشته ورودی می‌باشد.
- (۳) برای هر ماشین تورینگ، یک ماشین تورینگ معادل، حداکثر با پنج حالت وجود دارد. البته این در صورتی است که افزایش تعداد نمادهای نوار ماشین مجاز باشد.
- (۴) یک ماشین تورینگ نمی‌تواند فقط شامل یک حالت باشد. زیرا نمی‌توان برای یک وضعیت نهایی، تابع انتقال نوشت.
- (۵) برای هر ماشین تورینگ، یک ماشین تورینگ معادل تنها با یک حالت نهایی وجود دارد.
- (۶) برای هر ماشین تورینگ غیرقطعی، یک ماشین تورینگ قطعی معادل با آن وجود دارد.
- (۷) زبان‌هایی وجود دارند که ماشین تورینگ آنها را نمی‌پذیرد.
- (۸) تعداد نمادهای نوار یک ماشین تورینگ را می‌توان به دو نماد کاهش داد. البته این در صورتی است که افزایش تعداد حالات ماشین، مجاز باشد.

نکات مربوط به زبان‌های بازگشتی و برشمردنی

- (۱) برخی زبان‌ها وجود دارند که بازگشتی برشمردنی نیز نمی‌باشند.
- (۲) مجموعه تمام اعداد گویا $Q = \{\frac{a}{b} \mid a, b \in Z, b \neq 0\}$ و Z مجموعه اعداد صحیح است) شمارش‌پذیر می‌باشد.
- (۳) مجموعه تمام اعداد حقیقی $R = \{ \}$ (مجموعه اعداد گویا و گنگ) شمارش‌ناپذیر می‌باشد.
- (۴) مجموعه تمام اعداد گنگ، شمارش‌ناپذیر می‌باشد.
- (۵) مجموعه تمامی ماشین‌های تورینگ، نامتناهی و در عین حال شمارش‌پذیر می‌باشد.
- (۶) هر مجموعه متناهی M ، یک مجموعه شمارش‌پذیر می‌باشد و مجموعه توانی آن (2^M) نیز شمارش‌پذیر خواهد بود.
- (۷) اگر مجموعه نامتناهی M ، یک مجموعه شمارش‌پذیر باشد، در آن صورت مجموعه توانی آن (2^M) شمارش‌پذیر نخواهد بود.
- (۸) $\sum^*$ (مجموعه تمامی رشته‌های یک الفبا) شمارش‌پذیر می‌باشد.
- (۹) مجموعه تمام زبان‌هایی که بازگشتی برشمردنی نیستند، شمارش‌ناپذیر می‌باشند.
- (۱۰) هر زبان حساس به متن، بازگشتی نیز می‌باشد. اما هر زبان بازگشتی، لزوماً حساس به متن نیست.

مجموعه همه زبان ها ناشماراست. اما مجموعه گرامرها و ماشین های تورینگ شمارا هستند. فقط برای زبان های رده l_0 گرامر و ماشین تورینگ پذیرنده وجود دارد. همه زبان ها دارای گرامر نیستند. همه زبان ها ماشین تورینگ پذیرنده ندارند.

مجموعه همه زبان ها ناشماراست $P(\Sigma^*)$



خصوصیات بستاری زبان های بازگشتی و بازگشتی برشمردنی

خواص بستاری این زبان ها، به قرار جدول زیر می باشد:

نوع زبان	اجتماع	اشتراک	الحاق	تفاضل	معکوس	متمم	بستار ستاره	هم ریختی	هم ریختی معکوس
REC	✓	✓	✓	✓	✓	✓	✓	×	✓
RE	✓	✓	✓	×	✓	×	✓	✓	✓

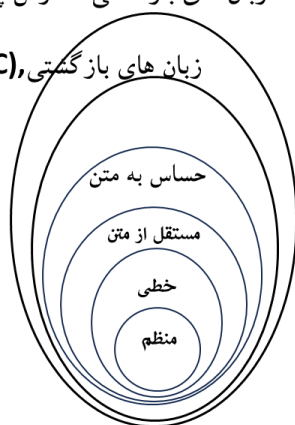
خصوصیات بستاری زبان‌های حساس به متن

خواص بستاری زبان‌های حساس به متن به قرار جدول زیر می‌باشد:

اجتماع	اشتراک	الحاق	تفاضل	معکوس	متمم	بستارستاره	همریختی	همیریختی معکوس
✓	✓	✓	✓	✓	✓	✓	✗	✓

زبان‌های بازگشتی شمارش پذیر (R.E)

زبان‌های بازگشتی (R), (REC),



علوم کامپیوتر - ۱۳۸۳

۵- مجموعه‌های بازگشتی شمارا (*recursively enumerable*) نسبت به کدام یک از مجموعه عملیات زیر بسته هستند؟

- (۱) اجتماع - اشتراک
- (۲) اجتماع - اشتراک - مکمل
- (۳) اجتماع - مکمل - الحاق (*concatenation*)
- (۴) اشتراک - مکمل - الحاق

پاسخ: گزینه ۱

زبان‌های بازگشتی شمارش‌پذیر (R.E) نسبت به همه این اعمال به جز مکمل و تفاضل بسته اند.

نوع زبان	اجتماع	اشتراک	الحاق	تفاضل	معکوس	مکمل	بستار ستاره	هم‌ریختی	هم‌ریختی معکوس
REC	✓	✓	✓	✓	✓	✓	✓	✗	✓
RE	✓	✓	✓	✗	✓	✗	✓	✓	✓

مهندسی کامپیوتر - ۱۳۸۰

۳- کدام یک از گزاره‌های زیر صحیح است؟

- (۱) مجموعه همه ماشین‌های تورینگ (*Turing Machines*) روی یک الفبا، شمارش‌پذیر (*countable*) است.
- (۲) مجموعه همه ماشین‌های تورینگ (*Turing Machines*) روی یک الفبا، شمارش‌ناپذیر (*uncountable*) است.
- (۳) مجموعه همه زبان‌های نامنظم (*non-regular*) روی یک الفبا، شمارش‌پذیر (*countable*) است.
- (۴) مجموعه تمامی رشته‌های تعریف شده روی یک الفبا، شمارش‌ناپذیر (*uncountable*) است.

پاسخ: گزینه ۱

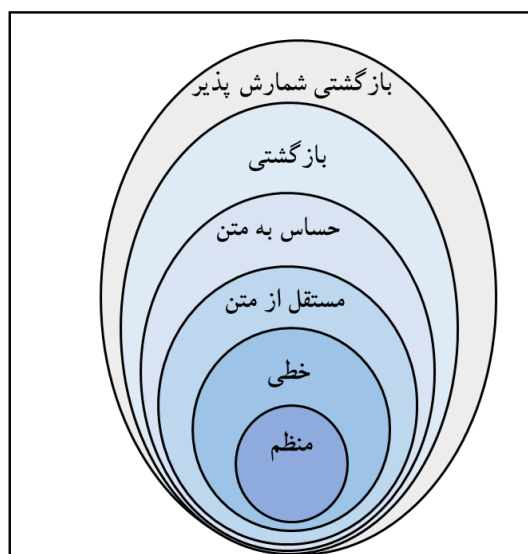
توجه: همه خانواده های درون بیضی ها شمارا هستند. ناحیه بیرون بیضی ها ناشماراست.

ناشمارا

خانواده زبان های نامنظم
خانواده زبان های غیرخطی
خانواده زبان های غیرمستقل از متن
خانواده زبان های غیرحساس به متن
خانواده زبان های غیربازگشتی
خانواده زبان های تصمیم ناپذیر
خانواده زبان های تشخیص ناپذیر

شمارا

خانواده زبان های منظم
خانواده زبان های خطی
خانواده زبان های مستقل از متن
خانواده زبان های حساس به متن
خانواده زبان های بازگشتی
خانواده زبان های تصمیم پذیر
خانواده زبان های تشخیص پذیر

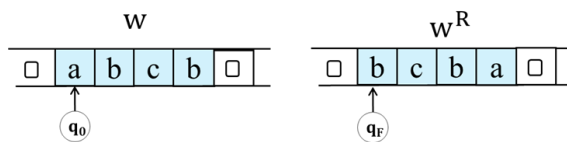


فصل نهم: ماشین تورینگ

جلسه بیست و دوم بخش دو: تورینگ محاسبه گر

ماشین تورینگ محاسبه گر:

می‌گوییم ماشین M تورینگ محاسبه کننده تابع است اگر w را روی نوار ورودی گرفته و در نهایت $f(w)$ را روی نوار تولید کند. در این صورت تابع f را یک تابع تورینگ-محاسبه پذیر گوییم.
مثال: تابع $f(w) = w^R$ تورینگ محاسبه پذیر است.



در پایان ضمن تولید $f(w)$ روی نوار باید ماشین در حالت نهایی باشد و هد در ابتدای رشته باشد.

$$q_0 w \vdash q_F f(w)$$

نمونه‌ها: این توابع محاسبه پذیر هستند.

$$f(w) = w^R$$

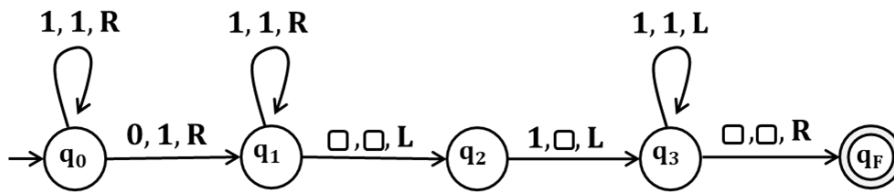
$$f(w) = w^2 = ww$$

$$f(n, m) = n + m$$

$$f(n, m) = n \cdot m$$

مثال: ماشین تورینگ محاسبه گر جمع دو عدد صحیح مثبت: $f(n, m) = n + m$

اعداد n و m را به صورت 1^n و 1^m با یک 0 جدا کننده به ماشین می دهیم. مثلاً ورودی 11110111 بیانگر $(n = 4, m = 3)$ است و در پایان رشته 1111111 روی نوار تولید می شود که عدد 7 است.



#	1	1	1	1	0	1	1	1	#
---	---	---	---	---	---	---	---	---	---

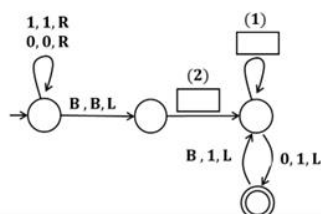
رشته ورودی

#	1	1	1	1	1	1	1	#	#
---	---	---	---	---	---	---	---	---	---

رشته خروجی

مهندسی کامپیوتر - ۱۳۹۳

۵۷- ماشین تورینگ مطابق شکل زیر با ورودی $x = (\phi + 1)^*$ مفروض است. به منظور پیاده سازی تابع $F(x) = x \text{div} 2 + 1$ آن را تکمیل نمایید. مقصود از - صفر (ϕ) یا یک (1) است.



2: - / ϕ / R	1: ϕ / 1 / R (۱)
2: 1 / B / L	1: ϕ / 1 / L (۲)
2: - / B / L	1: 1 / ϕ / L (۳)
2: 1 / ϕ / R	1: 1 / ϕ / L (۴)

پاسخ: گزینه ۳

در هر مبنایی که باشیم، برای تقسیم اعداد صحیح در همان مبنا باید رقم یکان حذف شود. مثلاً در مبنای ۱۰ تقسیم عدد صحیح ۲۴۵۷ بر ۱۰ برابر است با عدد صحیح ۲۴۵ است. پس در مبنای ۲ نیز برای تقسیم بر ۲ باید رقم یکان حذف شود، پس دستورالعمل کادر (۲) چنین است: رقم یکان هرچه که بود به جایش B قرار بده و به سمت چپ برو.

(2)
..., B, L

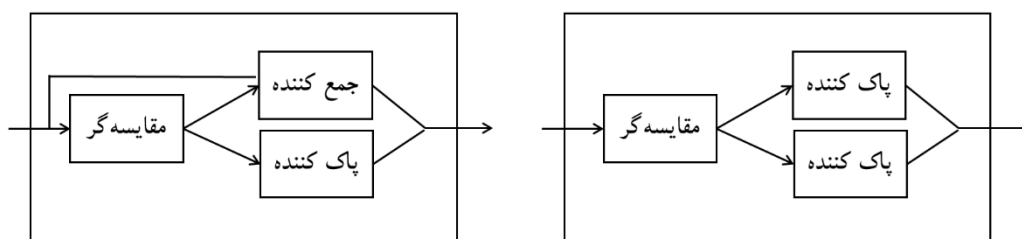
حالا عملیات تقسیم بر ۲ انجام شده. باید یک واحد به این عدد اضافه کنیم. اگر رقم آخر صفر باشد کافیست به ۱ تبدیل شود و کار تمام است. اگر رقم آخر یک باشد ضمن حرکت به چپ، تا جایی که ۱ ها ادامه دارند به صفر تبدیل کنیم تا به اولین صفر برسیم و آن را به ۱ تبدیل کنیم. پس داریم:

(1)
1, 0, L

(البته این ماشین برای زمانی که همه رقم ها تا آخر ۱ باشند پیش بینی نکرده. اما این مشکل هم قابل حل است)

ترکیب ماشین‌های تورینگ محاسبه‌گر:

با ترکیب ماشین‌های تورینگ که اعمال ساده و مقدماتی را انجام می‌دهند می‌توان ماشین تورینگ طراحی کرد که فرایندهای پیچیده‌تر مانند مرتب سازی اعداد صحیح را انجام دهد.



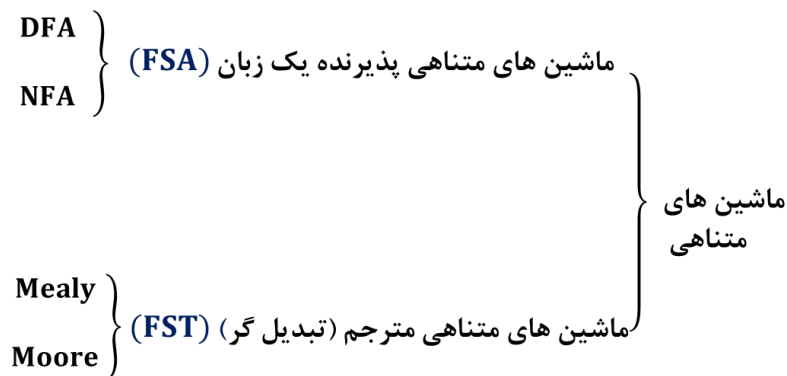
$$f(x, y) = \begin{cases} x + y & x \geq y \\ 0 & x < y \end{cases}$$

$$g(x, y) = \max(x, y) = \begin{cases} x & x \geq y \\ y & x < y \end{cases}$$

-23-

فصل نهم: ماشین تورینگ

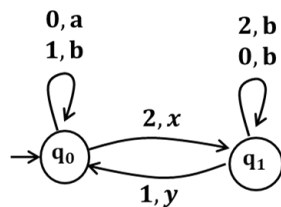
جلسه بیست و سوم بخش یک: ماشین های متناهی مترجم یا محاسبه گر



ماشین میلی (Mealy):

ماشین میلی نوعی ماشین متناهی مترجم (محاسبه گر) است که رشته ورودی w یا ورودی هایی با چند مولفه را گرفته و در خروجی $f(w)$ را به وجود می آورد.

$$M = (Q, \Sigma, \Gamma, \delta, \sigma, q_0)$$



input	0	2	2	1
output	a	x	b	y

Q مجموعه متناهی حالات ماشین

Σ مجموعه متناهی الفبای ورودی

Γ مجموعه متناهی الفبای خروجی

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

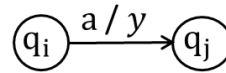
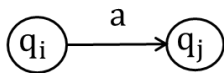
$\sigma: Q \times \Sigma \rightarrow \Gamma$ تابع خروجی

q_0 حالت اولیه

گاه توابع انتقال و خروجی را با هم نمایش می دهیم: $(\delta, \sigma): Q \times \Sigma \rightarrow Q \times \Gamma$

خروجی گرفتن از ماشین میلی:

در ماشین میلی با انجام هر انتقال δ ، یک علامت هم در خروجی نمایش داده می‌شود اینکه چه علامتی در خروجی قرار بگیرد، بستگی به حالت فعلی و الفبای ورودی دارد. خروجی را تابع σ نشان می‌دهد.



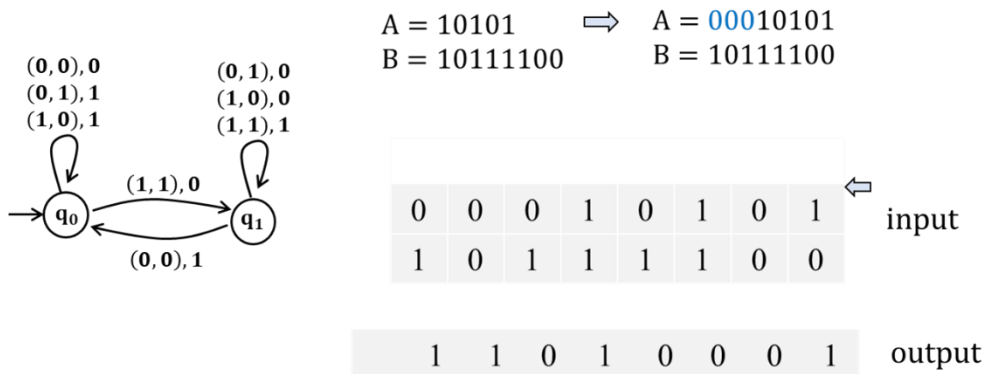
$$\delta(q_i, a) = q_j$$

$$\sigma(q_i, a) = y$$

$$\underbrace{(\delta, \sigma)}_T(q_i, a) = (q_j, y)$$

مثال ماشین میلی: مجموع دو عدد در مبنای ۲

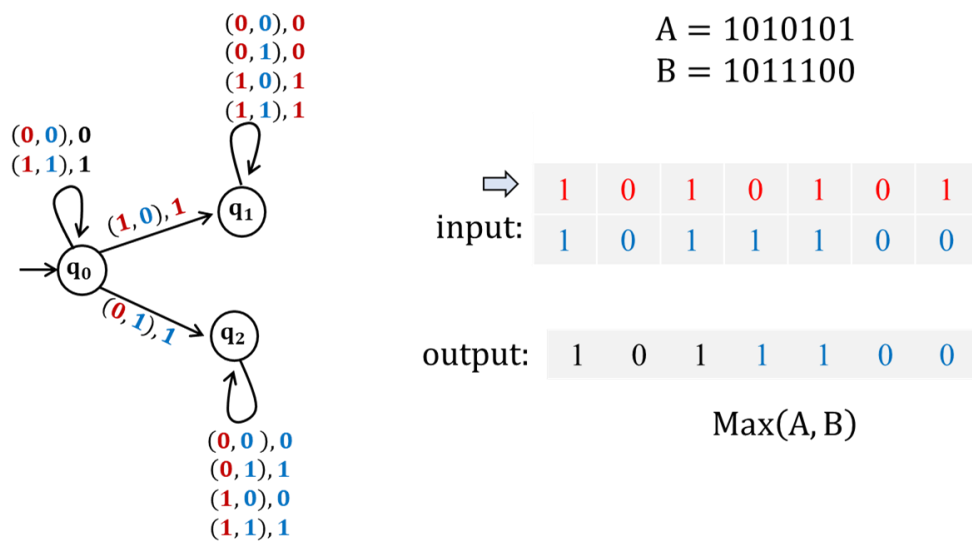
فرض کنید می‌خواهیم دو عدد باینری را با هم جمع کنیم. با افزودن بیت های بی تاثیر صفر، کاری می‌کنیم که هر دو رشته دارای طول مشترک n باشند. محاسبه جمع از ارقام یکان آغاز می‌شود. پس فرض کنیم این دو عدد از راست به چپ (با شروع از رقم یکان) در ورودی خوانده می‌شوند.



ماشین مورد نظر به دو حالت نیاز دارد یکی برای حالت عادی مجموع بیت ها (بدون وجود رقم ۱ اضافی) و دیگری برای محاسبه مجموع بیت ها با حضور یک رقم ۱ سرریز شده از مرحله قبل.

ماشین میلی Max: محاسبه ماکزیمم دو عدد در مبنای ۲

برای یافتن ماکزیمم (یا می‌نیمم) باید بررسی را از بیت‌های باارزش تر آغاز کنیم. پس برخلاف عمل جمع که از رقم یکان شروع می‌شد، این بار خواندن ورودی از ارقام با ارزش تر (چپ به راست) شروع می‌شود.

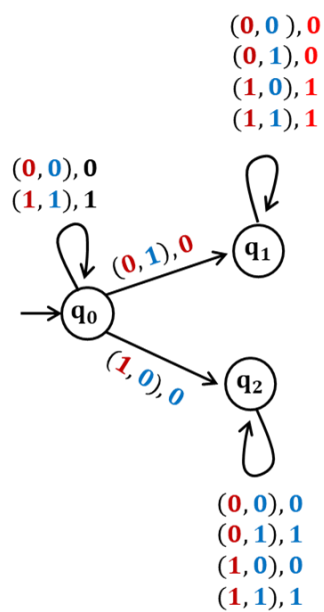


q_0 : تا وقتی که $A = B$ است.

q_1 : وقتی $A > B$ باشد.

q_2 : وقتی $B > A$ باشد.

ماشین میلی Min : محاسبه می نیمم دو عدد در مبنای ۲



$A = 1010101$

$B = 1011100$

input:

1	0	1	0	1	0	1
1	0	1	1	1	0	0

output:

1	0	1	0	1	0	1
---	---	---	---	---	---	---

Min(A, B)

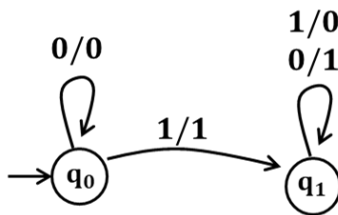
q_0 : تا وقتی که $A = B$ است.

q_1 : وقتی $A < B$ باشد.

q_2 : وقتی $B < A$ باشد.

ماشین میلی مکمل دو: محاسبه مکمل دوی عدد باینری n بیتی A

مکمل دوی عدد باینری n بیتی A برابر است با $2^n - A$. مثلاً اگر در یک سیستم برای نمایش اعداد از $n = 4$ بیت استفاده شود، مکمل دوی عدد A برابر است با $2^4 - A$. اگر فرض کنیم رشته ورودی به ترتیب از کم ارزش ترین رقم تا پر ارزش ترین باشد (از راست به چپ) آنگاه ماشین میلی برای محاسبه مکمل دوی A چنین عمل می کند: صفرهای سمت راست، حفظ می شوند. اولین رقم ۱ نیز حفظ می شود. اما از آن به بعد همه رقم ها تغییر می کنند:



صفرها به یک و یک ها به صفر تبدیل می شوند.

$n = 4, A = 0011$

0	0	1	1	← input
عدد ۳				
1	1	0	1	output
عدد ۱۳				

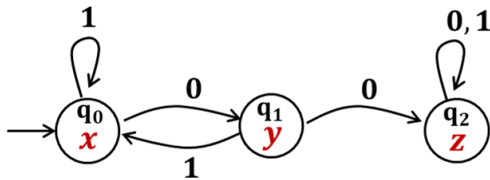
$n = 4, B = 1010$

1	0	1	0	← input
عدد ۱۰				
0	1	1	0	output
عدد ۶				

ماشین مور (Moore):

ماشین میلی نوعی ماشین متناهی مترجم (محاسبه گر) است که رشته ورودی w یا ورودی هایی با چند مولفه را گرفته و در خروجی $f(w)$ را به وجود می آورد.

$$M = (Q, \Sigma, \Gamma, \delta, \sigma, q_0)$$



Q مجموعه متناهی حالات ماشین

Σ مجموعه متناهی الفبای ورودی

Γ مجموعه متناهی الفبای خروجی

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

$\sigma: Q \rightarrow \Gamma$ تابع خروجی

q_0 حالت اولیه

input: w

1	0	0	1
---	---	---	---

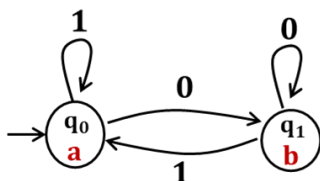
output: $f(w)$

x	y	z	z
-----	-----	-----	-----

خروجی گرفتن از ماشین مور:

در ماشین میلی تابع σ به هر حالت ماشین یک علامت (نماد) نسبت داده است. در شروع کار که در q_0 هستیم علامتی در خروجی درج نمی شود. اما با انجام هر انتقال δ و با ورود به هر حالت q_j علامت $\sigma(q_j)$ در خروجی قرار می گیرد.

مثلا ماشین مور روبه رو رشته باینری 1001 را به رشته abba ترجمه می کند.



input: w

1	0	0	1
---	---	---	---

output: $h(w)$

a	b	b	a
---	---	---	---

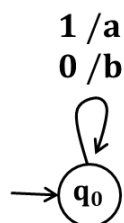
تمرین: همریختی h داده شده است. ماشین مور و ماشین میلی برای ترجمه رشته w به رشته $h(w)$ را طراحی کنید.

$$h: \{0,1\}^* \rightarrow \{a,b\}^*$$

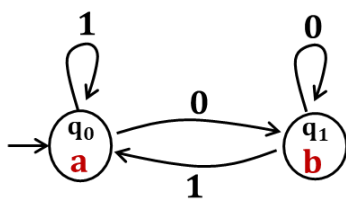
$$h(1) = a$$

$$h(0) = b$$

ماشین میلی:



ماشین مور:



input:

1	0	0	1
---	---	---	---

output:

a	b	b	a
---	---	---	---

input:

1	0	0	1
---	---	---	---

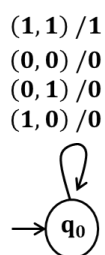
output:

a	b	b	a
---	---	---	---

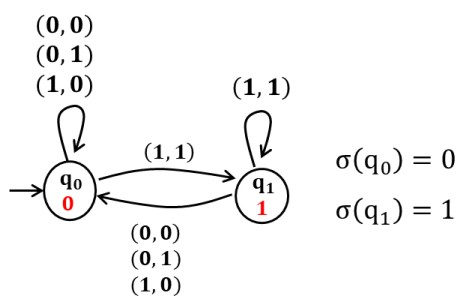
مثال: برای عملگر AND ماشین مور و ماشین میلی طراحی کنید:

با افزودن ارقام صفر بی تاثیر به سمت چپ، می توان هر دو عدد را n رقمی نوشت. حالا ورودی را از چپ به راست می خوانیم.

ماشین میلی:



ماشین مور:



$\Rightarrow$

input:	0	0	1	0	1	1	1
	1	0	1	1	1	0	0

output:	0	0	1	0	1	0	0
---------	---	---	---	---	---	---	---

AND(A, B)

$\Rightarrow$

input:	0	0	1	0	1	1	1
	1	0	1	1	1	0	0

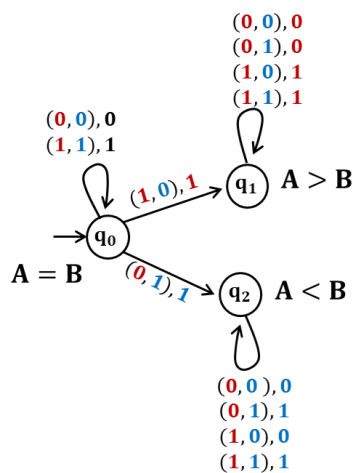
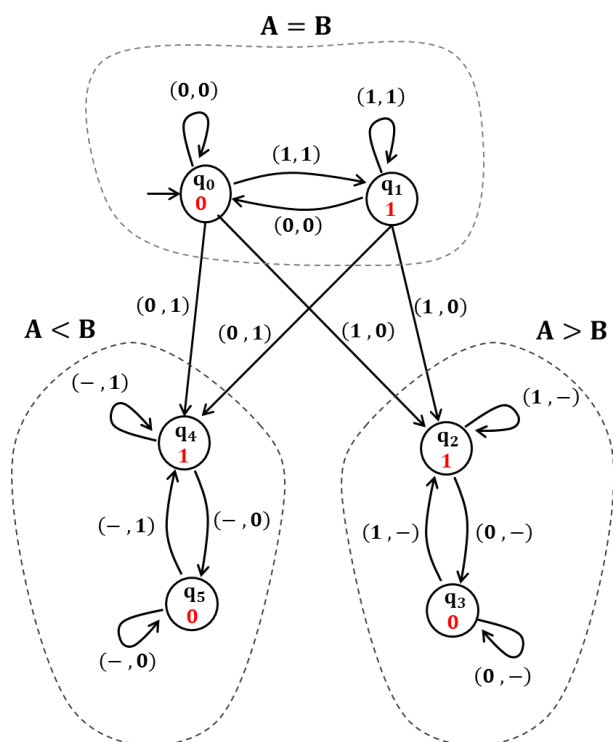
output:	0	0	1	0	1	0	0
---------	---	---	---	---	---	---	---

AND(A, B)

ماشین مور و ماشین میلی برای عملگر Max :

ماشین مور:

ماشین میلی:



پایان فصل نهم: ماشین تورینگ

جلسه بیست و سوم بخش دو:

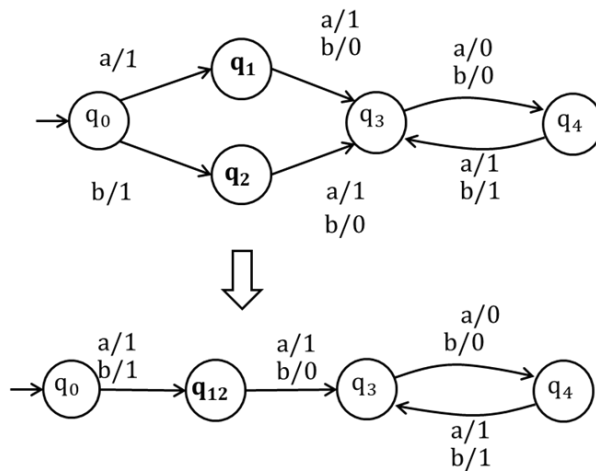
کمینه کردن ماشین میلی و مور ماشین های، مبدل تاخیر ماشین تورینگ محاسبه گر

کمینه کردن ماشین Mealy :

حالات قابل ادغام (هم‌ارز) در ماشین های میلی و مور حالاتی هستند که با شروع از هر کدام و با ورودی یکسان، خروجی یکسان به دست آید. به بیان دقیق تر دو حالت q_i و q_j قابل ادغام هستند هرگاه برای هر رشته ورودی w داشته باشیم:

$$\sigma^*(q_i, w) = \sigma^*(q_j, w)$$

که σ تابع خروجی است. در شکل یک ماشین میلی را می بینید که حالات q_1 و q_2 در آن قابل ادغام هستند. برای رشته ورودی $w = aab$ چه از q_1 شروع کنیم چه از q_2 فرقی ندارد و رشته خروجی یکسان 101 به دست می آید. یکسان بودن خروجی، برای همه ورودی ها دیده می شود. پس این دو حالت هم‌ارز و قابل ادغام هستند. با ادغام آنها در یک حالت، به ماشین معادلی می‌رسیم که کمینه است چون دیگر جفت قابل ادغام ندارد.



EXERCISES

ماشین متناهی ایجاد کننده تاخیر (از تمرینات فصل اول لینز)

8. We assumed that an automaton works in a framework of discrete time steps, but this aspect has little influence on our subsequent discussion. In digital design, however, the time element assumes considerable significance. In order to synchronize signals arriving from different parts of the computer, delay circuitry is needed. A unit-delay transducer is one that simply reproduces the input (viewed as a continual stream of symbols) one time unit later. Specifically, if the transducer reads as input a symbol a at time t , it will reproduce that symbol as output at time $t + 1$.

At time $t = 0$, the transducer outputs nothing. We indicate this by saying that the transducer translates input $a_1 a_2 a_3 \dots$ into output $\lambda a_1 a_2 a_3 \dots$

(a) Draw a graph showing how such a unit-delay transducer might be designed for $\Sigma = \{a, b\}$.

9. An n -unit delay transducer is one that reproduces the input n time units later; that is, the input $a_1 a_2 \dots$ is translated into $\lambda n a_1 a_2 \dots$, meaning again that the transducer produces no output for the first n time slots.

(a) Construct a two-unit delay transducer on $\Sigma = \{a, b\}$.

(b) Show that an n -unit delay transducer must have at least $|\Sigma|^n$ states. ($|Q| \geq |\Sigma|^n$)

در ادامه محتوای این دو تمرین و ویژگی‌های این ماشین را مطالعه می‌کنیم.

ماشین مبدل برای ایجاد n واحد تاخیر (Dealy transducer):

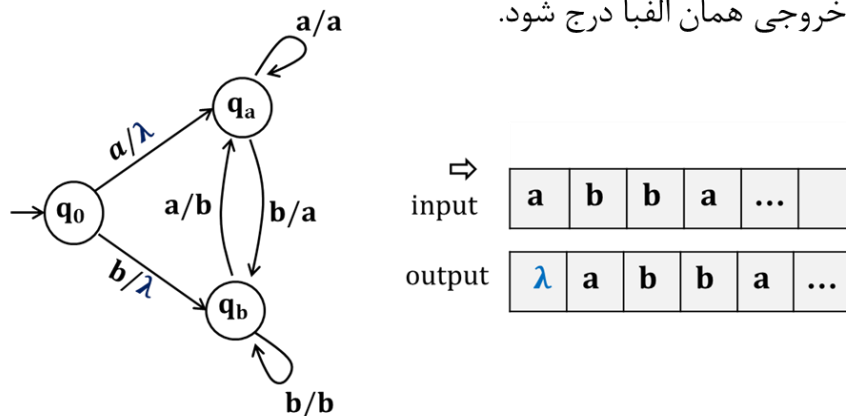
به منظور همگام سازی سیگنال هایی که از بخش های مختلف سیستم کامپیوتر به واحد پردازش می رسند، مدار ایجاد تاخیر مورد نیاز است. برای ایجاد n واحد تاخیر در رشته ورودی $a_1 a_2 a_3 \dots$ (که نشان دهنده جریان پیوسته ای از سیگنال ها است)، به مداری نیاز داریم که این رشته را به رشته $\lambda^n a_1 a_2 a_3 \dots$ تبدیل کند یعنی به ابتدای آن n گام تهی اضافه کند.

$$a_1 a_2 a_3 \dots \Rightarrow \lambda^n a_1 a_2 a_3 \dots$$

این کار توسط یک ماشین متناهی انجام می شود. دقت کنید که رشته ورودی بی انتها فرض شده است.

مثال: مبدل یک واحد تاخیر:

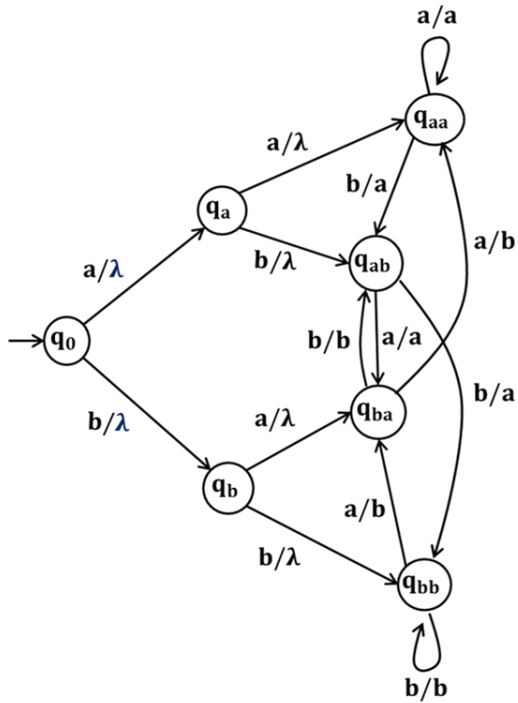
در شروع، با خواندن اولین الفبا چیزی در خروجی درج نمی شود (λ). اما اگر این الفبا باشد به حالت q_a می رویم و اگر باشد به حالت q_b می رویم. نام این حالات نشان می دهد که یک گام قبل، چه الفبایی داشتیم، اکنون باید در خروجی همان الفبا درج شود.



مثال: مبدل ۲ واحد تاخیر:

برای دو الفبای ابتدایی چیزی در خروجی درج نمی‌شود ($\lambda\lambda$).

اگر این دو الفبا aa, ab, ba, bb باشند به ترتیب به حالت های $q_{aa}, q_{ab}, q_{ba}, q_{bb}$ می‌رسیم. نام این حالات نشان می‌دهد که در دو گام قبل، چه الفبایی داشتیم، اکنون باید در خروجی همان الفباها به ترتیب درج شوند.



توجه: مبدل تاخیر n واحدی دستکم دارای $|\Sigma|^n$ حالت است: $|\mathbb{Q}| \geq |\Sigma|^n$

همانطور که در دو مثال قبل برای ۱ واحد و ۲ واحد تاخیر مشاهده کردید، برای ایجاد n واحد تاخیر علاوه بر حالت، برای هر رشته به طول یک، دو، سه، ...، n نیاز به یک حالت داریم.

تعداد رشته‌های به طول n برابر $|\Sigma|^n$ است. پس $|\mathbb{Q}| \geq |\Sigma|^n$.

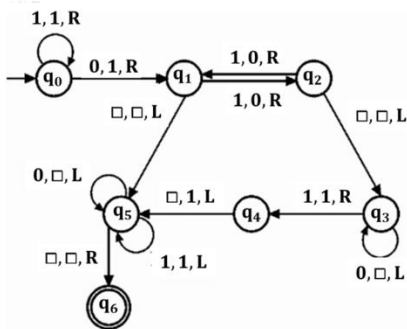
البته با در نظر گرفتن نحوه ساخت ماشین‌ها در دو مثال قبل می‌توان گفت:

$$|\mathbb{Q}| = 1 + |\Sigma|^1 + |\Sigma|^2 + \dots + |\Sigma|^n$$

حل چند تست از تورینگ محاسبه گر و تورینگ پذیرنده:

23- ماشین تورینگ مطابق شکل زیر با ورودی $\{w = x0y \mid x = 1^*, y = 1^*\}$ مفروض است، کدام تابع توسط آن حساب می‌شود؟

(مهندس کامپیوتر - 92)

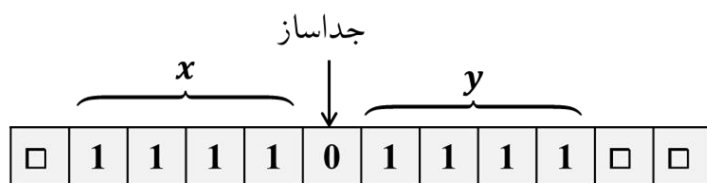


$$f(x,y) = x + 1 + (y \bmod 2) \quad (1)$$

$$f(x,y) = x + (y \bmod 2) \quad (2)$$

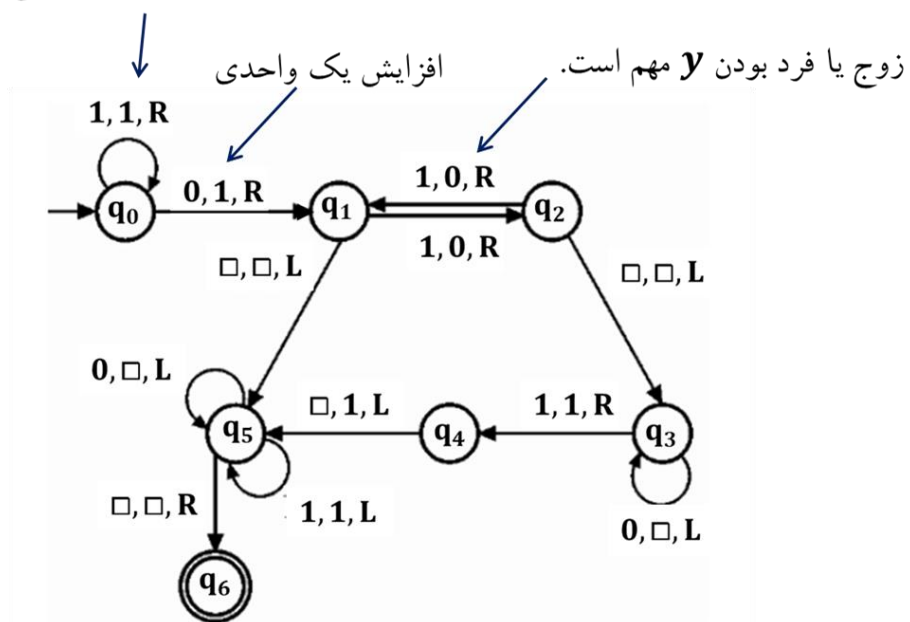
$$f(x,y) = (x + y + 1) \bmod 2 \quad (3)$$

$$f(x,y) = (x + y) \bmod 2 \quad (4)$$

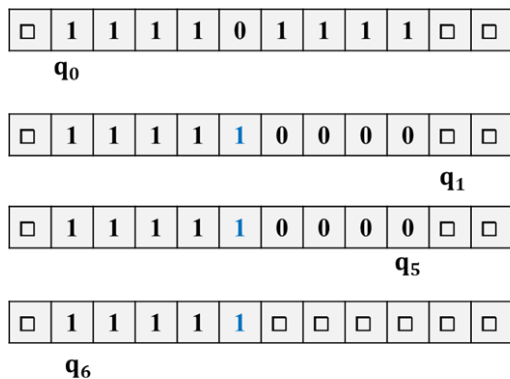


پاسخ: گزینه ۱

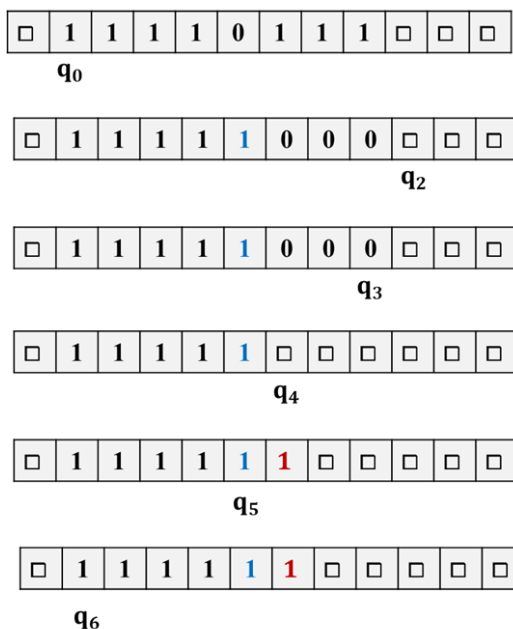
x به طور کامل حفظ می شود



y زوج باشد:

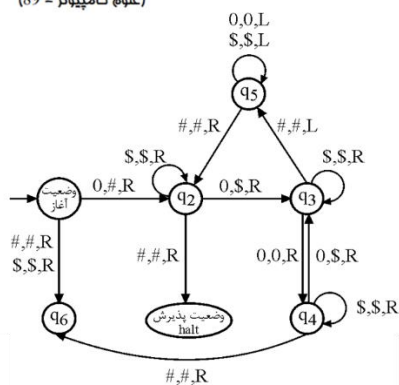


y فرد باشد:



34- در ماشین تورینگ زیر نماد # نشان دهنده خانه خالی، و نماد \$ یکی از حروف نواری ماشین است. منظور از انتقال به شکل x, y, R این است که حرف y به جای x جایگزین شده و هد به سمت راست حرکت می‌کند. در انتقال به شکل x, y, L نیز حرف y به جای x جایگزین شده و هد به سمت چپ حرکت می‌کند. زبان ماشین تورینگ زیر عبارت است از:

(علوم کامپیوتر - 89)



$$\{0^{2n} \mid n \geq 0\} \quad (1)$$

$$\{0^{2n} \mid n \geq 0\} \quad (2)$$

$$\{0^{n^2} \mid n \geq 0\} \quad (3)$$

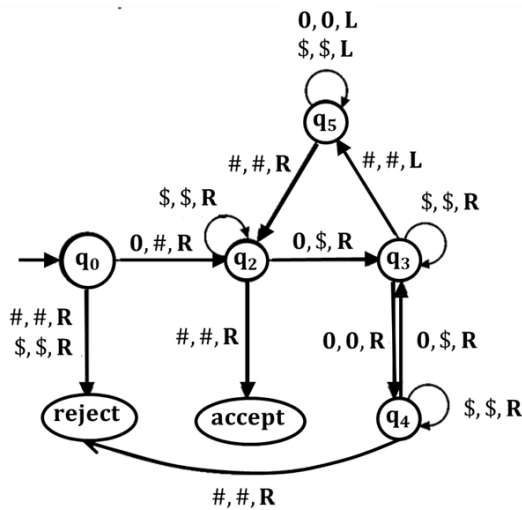
$$\{0^{n^n} \mid n \geq 0\} \quad (4)$$

پاسخ: گزینه ۲

ماشین تورینگ داده شده را به این صورت هم می‌توان نشان داد.

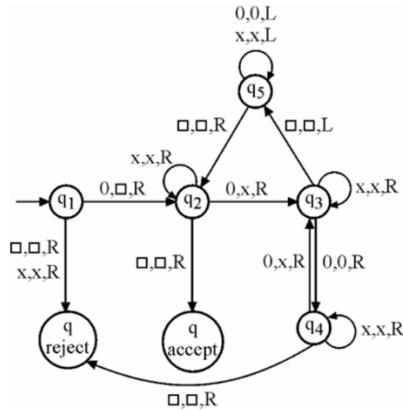
این ماشین اعداد 2^n را می‌پذیرد که به صورت 0^{2^n} به ماشین داده می‌شوند.

حلقه ایجاد شده توسط q_3 و q_4 طول رشته را نصف می‌کند در هر مرحله اگر عدد باقیمانده، عدد فرد بزرگتر از یک باشد به حالت $q_6: \text{reject}$ وارد می‌شود و رد می‌شود. اما اگر زوج باقی بماند از طریق q_5 وارد تکرار بعدی می‌شود. تنها اعدادی در نهایت وارد حالت پذیرش می‌شوند که تا رسیدن به عدد ۱، همواره زوج باقی بمانند.



86- فرض کنید M ماشین تورینگ زیر باشد: اگر $\alpha = 0^{16}$ و $\beta = 0^{17}$ کدام یک از این دو رشته توسط M پذیرفته می شود؟

(علوم کامپیوتر - 1401)



(1) فقط α

(2) فقط β

(3) هر دو

(4) هیچ کدام

پاسخ: گزینه ۱

همان ماشین تورینگ مساله قبل است. اعداد 2^n را می پذیرد که به صورت 0^{2^n} به ماشین داده می شوند.

موضوع جلسه بعد:

مسائل تصمیم:

برخی از سوالات پاسخ (yes – no) دارند. آنها را مسائل تصمیم می نامیم.

نمونه هایی از مسائل تصمیم در این درس:

آیا $w \in L(G)$ ؟

آیا $L(G) = \emptyset$ ؟

آیا $L(G)$ متناهی است ؟

آیا $L(G_1) = L(G_2)$ ؟



-24-

ف.، کاهش مساله

جلسه بیست و چهارم بخش یک: مساله تصمیم، ارتباط مساله و زبان

مسائل تصمیم:

برخی از سوالات پاسخ (yes – no) دارند. آنها را مسائل تصمیم می‌نامیم.

نمونه‌هایی از مسائل تصمیم در این درس:

آیا $w \in L(G)$ ؟

آیا $L(G) = \emptyset$ ؟

آیا $L(G)$ متناهی است ؟

آیا $L(G_1) = L(G_2)$ ؟

برخی از این مسائل تصمیم پذیر هستند و برخی خیر. ابتدا معنای تصمیم‌پذیری را بیان کنیم.

معنای تصمیم پذیر بودن یک مساله از دیدگاه الگوریتمی:

در این دیدگاه، مساله را تصمیم‌پذیر گوییم هرگاه الگوریتمی وجود داشته باشد که پاسخ آن را در زمان متناهی تعیین کند. یعنی چه پاسخ مثبت بود، چه منفی، الگوریتم پس از مدت زمان متناهی متوقف شود و به جواب برسد.

یک مساله معروف تصمیم‌ناپذیر: مساله صحت الگوریتم (Algorithm correctness problem)

آیا الگوریتمی وجود دارد که صحت الگوریتم‌ها را تایید کند؟
یعنی الگوریتم دلخواه A و ورودی α را به آن بدهیم و تعیین کند که آیا الگوریتم A به ازای این ورودی، خروجی درست خواهد داشت یا خیر؟
پاسخ: خیر چنین الگوریتمی وجود ندارد.

معنای تصمیم پذیر بودن یک مساله از دیدگاه نظریه زبانها:

هر مساله تصمیم را می‌توان با یک مجموعه (یعنی یک زبان) نظیر کرد. برای مثال این مساله را در نظر بگیرید:

مساله P: آیا گراف G همیلتونی است؟، پاسخ این مساله برای هر گراف G یا (yes) یا (no) است. حالا فرض کنیم L مجموعه تمام گراف‌هایی باشد که همیلتونی هستند. یعنی همه گراف‌هایی که پاسخ مساله برای آنها (yes) است:

$$\text{زبان } L: \{ \text{گراف } G \text{ همیلتونی است} \mid G \}$$

مساله P تصمیم‌پذیر است اگر و تنها اگر زبان L تصمیم‌پذیر باشد. مثال دیگر:

مساله P: آیا ۲۷۱ عددی اول است؟

$$\text{زبان } L: \{ n \in \mathbb{N} \mid n \text{ عدد اول است} \}$$

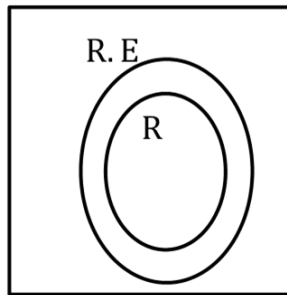
مساله P تصمیم‌پذیر است اگر و تنها اگر زبان L تصمیم‌پذیر باشد.

نتیجه:

هر مساله تصمیم‌پذیر، با یک زبان تصمیم‌پذیر در تناظر است.

تصمیم پذیر بودن زبان ها و الگوریتم عضویت :

زبان L تصمیم پذیر است اگر و تنها اگر الگوریتم عضویت داشته باشد. یعنی الگوریتمی وجود داشته باشد که برای هر رشته $w \in \Sigma^*$ عضویت یا عدم عضویت w در L را مشخص کند و در هر صورت متوقف شود.



$L \in R \Leftrightarrow L$ الگوریتم عضویت دارد

مسائل تصمیم پذیر و تصمیم ناپذیر معروف برای رده های مختلف زبان ها:

۱- A_{DFA} تصمیم پذیر است.

$$A_{DFA} = \{ \langle M, w \rangle \mid M \text{ is DFA and } w \in L(M) \}$$

برای هر جفت $\langle M, w \rangle$ داریم:

$$M \Leftrightarrow \langle M, w \rangle \in A_{DFA} \text{ ماشین DFA باشد و } w \text{ توسط ماشین } M \text{ پذیرفته شود.}$$

اولا DFA بودن یا نبودن M قابل تشخیص است. اگر M ماشین DFA باشد هر رشته w به طول n را دقیقا در n گام مصرف می کند و عضویت یا عدم عضویت w در $L(M)$ را مشخص می کند. پس الگوریتمی (ابزاری) وجود دارد که برای هر جفت $\langle M, w \rangle$ عضو بودن یا عضو نبودن آن را در A_{DFA} مشخص می کند. (در هر دو حالت متوقف می شود).

۲- A_{CFG} تصمیم‌پذیر است.

$$A_{CFG} = \{ \langle G, w \rangle \mid G \text{ is CFG and } w \in L(G) \}$$

برای هر جفت $\langle G, w \rangle$ داریم:

$$G \Leftrightarrow \langle G, w \rangle \in A_{CFG} \iff G \text{ گرامری مستقل از متن باشد و } w \text{ توسط } G \text{ تولید شود.}$$

مستقل از متن بودن یا نبودن یک گرامر به سادگی قابل تشخیص است. (در هر دو صورت).
و اگر G مستقل از متن باشد، مساله عضویت یا عدم عضویت w در $L(G)$ تصمیم‌پذیر است.

توجه:

هر گرامر مستقل از متن با یک ماشین پشته‌ای معادل است و بالعکس. پس به طور مشابه برای ماشین‌های پشته‌ای هم A_{PDA} تصمیم‌پذیر است.

$$A_{CFG} = \{ \langle M, w \rangle \mid M \text{ is PDA and } w \in L(M) \}$$

۳- A_{TM} تصمیم‌پذیر نیست.

$$A_{TM} = \{ \langle M, w \rangle \mid M \text{ is turing and } w \in L(M) \}$$

برای هر جفت $\langle M, w \rangle$ داریم:

$$M \Leftrightarrow \langle M, w \rangle \in A_{TM} \text{ کد یک ماشین تورینگ باشد و } w \text{ توسط } M \text{ پذیرفته شود.}$$

ثابت می‌کنیم A_{TM} تصمیم‌ناپذیر است.

برهان خلف: فرض کنیم A_{TM} تصمیم‌پذیر باشد. در این صورت یک الگوریتم (یا ماشین) به نام H وجود دارد که برای هر زوج $\langle M, w \rangle$ عضویت یا عدم عضویت در A_{TM} را مشخص می‌کند و متوقف می‌شود.

$$H(\langle M, w \rangle) = \begin{cases} \text{yes} & w \in L(M) \\ \text{no} & w \notin L(M) \end{cases}$$

در حالت خاص، اگر رشته w همان کد باینری ماشین M باشد، الگوریتم می‌تواند تعیین کند که

آیا $\langle M \rangle \in L(M)$ یا $\langle M \rangle \notin L(M)$. به عبارتی زبان‌های زیر تصمیم‌پذیر هستند:

$$l_1 = \{ \langle M \rangle \mid M \text{ is turing and } \langle M \rangle \in L(M) \}$$

$$l_2 = \{ \langle M \rangle \mid M \text{ is turing and } \langle M \rangle \notin L(M) \}$$

:

$$I_2 = \{ \langle M \rangle \mid M \text{ is turing and } \langle M \rangle \notin L(M) \}$$

I_2 تصمیم پذیر است پس دارای ماشین تورینگ تصمیم گیرنده است. ماشینی (الگوریتمی) به نام وجود دارد که برای هر کد $\langle M \rangle$ می تواند به این صورت عمل کند :

$$D(\langle M \rangle) = \begin{cases} \text{accept} & \langle M \rangle \notin L(M) \\ \text{reject} & \langle M \rangle \in L(M) \end{cases}$$

حالا ماشین D خودش هم یک ماشین تورینگ است پس می توانیم کد خودش را به عنوان ورودی در نظر بگیریم:

$$D(\langle D \rangle) = \begin{cases} \text{accept} & \langle D \rangle \notin L(D) \\ \text{reject} & \langle D \rangle \in L(D) \end{cases}$$

این یک تناقض است. ماشین D اگر $\langle D \rangle$ را بپذیرد باید نپذیرد و اگر آن را نپذیرد باید بپذیرد! این تناقض چرا رخ داد؟ چون ما فرض کردیم که A_{TM} تصمیم پذیر باشد.

نتیجه: A_{TM} تصمیم پذیر نیست.

۴- می‌دانیم مساله تهی بودن برای زبان مستقل از متن تصمیم پذیر است.

ما نشان می‌دهیم در حالت کلی برای زبان های دارای گرامر بدون محدودیت یعنی زبان های دارای ماشین تورینگ، مساله تهی بودن تصمیم پذیر نیست.

حکم: مجموعه $EMPTY_{TM}$ تصمیم ناپذیر است.

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

اثبات:

نشان می‌دهیم اگر $EMPTY_{TM}$ تصمیم پذیر باشد آنگاه A_{TM} نیز تصمیم پذیر می‌شود.

فرض کنیم $EMPTY_{TM}$ تصمیم پذیر باشد پس ماشین تورینگی (الگوریتمی) به نام S وجود دارد که می‌تواند برای هر ماشین تورینگ M

تهی یا ناتهی بودن $L(M)$ را تعیین کرده و متوقف شود:

$$S(\langle M \rangle) = \begin{cases} \text{accept} & L(M) = \emptyset \\ \text{reject} & L(M) \neq \emptyset \end{cases}$$

اکنون جفت دلخواه $\langle M, w \rangle$ را در نظر بگیرید. می‌توانیم با افزودن انتقالهایی به ابتدای ماشین M ,

ماشین تورینگ M_1 را به دست آوریم که هر رشته به جز w را به حالت عدم پذیرش ببرد و فقط به

رشته w اجازه ورود به M را بدهد. که ممکن است پذیرفته بشود یا نه.

$$S(\langle M \rangle) = \begin{cases} \text{accept} & L(M) = \emptyset \\ \text{reject} & L(M) \neq \emptyset \end{cases}$$

اکنون جفت $\langle M, w \rangle$ را در نظر بگیرید. می‌توانیم با افزودن انتقالهایی به ابتدای ماشین M ، ماشین تورینگ M_1 را به دست آوریم که هر رشته به جز w را به حالت عدم پذیرش ببرد و فقط به رشته w اجازه ورود به M را بدهد. که ممکن است پذیرفته بشود یا نه. (*یک نمونه از تبدیل M به M_1 را در اسلاید بعدی مشاهده کنید) بنابراین خواهیم داشت:

$$L(M_1) = \{w\} \cap L(M)$$

به بیان بهتر $L(M_1)$ دو حالت دارد: یا تهی است یا $\{w\}$ است.

$L(M_1) = \emptyset$ اگر و تنها اگر $w \notin L(M)$.

$L(M_1) \neq \emptyset$ اگر و تنها اگر $w \in L(M)$.

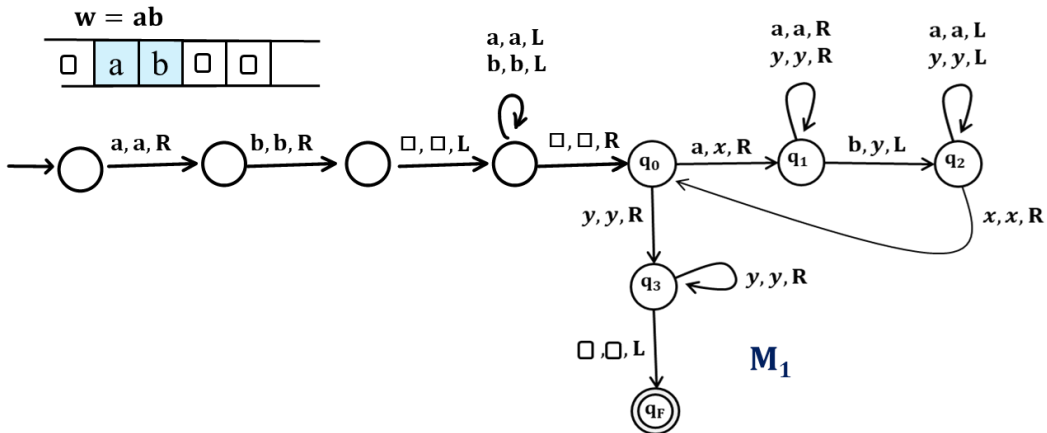
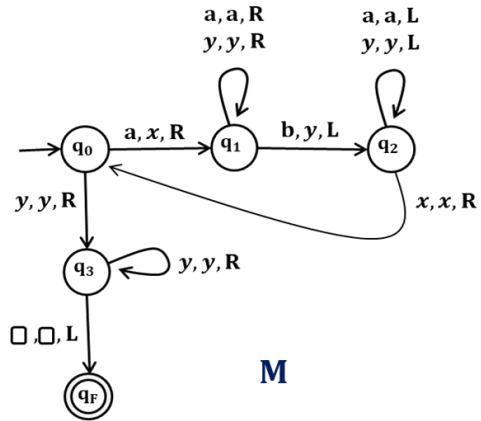
طبق فرض خلف، مساله تهی یا ناتهی بودن $L(M_1)$ تصمیم‌پذیر است پس مساله عضویت یا عدم

عضویت w در $L(M)$ تصمیم‌پذیر است. به عبارتی A_{TM} تصمیم‌پذیر است!

این تناقض از آنجا رخ داد که فرض کردیم $EMPTY_{TM}$ تصمیم‌پذیر باشد.

نتیجه: $EMPTY_{TM}$ تصمیم‌پذیر نیست.

* می‌دانیم ماشین تورینگ M زبان $\{a^n b^n | n \geq 1\}$ را می‌پذیرد. حالا با افزودن چند انتقال ماشین M_1 را به دست آوردیم که فقط رشته $w = ab$ را می‌پذیرد.



تصميم پذير $A_{\text{DFA}} = \{ \langle M, w \rangle \mid M \text{ is DFA and } w \in L(M) \}$

تصميم پذير $A_{\text{CFG}} = \{ \langle G, w \rangle \mid M \text{ is CFG and } w \in L(G) \}$

تصميم ناپذير $A_{\text{TM}} = \{ \langle M, w \rangle \mid M \text{ is turing and } w \in L(M) \}$

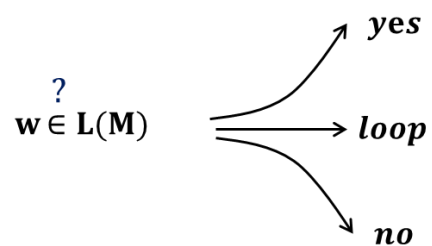
تصميم ناپذير $\text{EMPTY}_{\text{TM}} = \{ \langle M \rangle \mid M \text{ is turing and } L(M) = \emptyset \}$

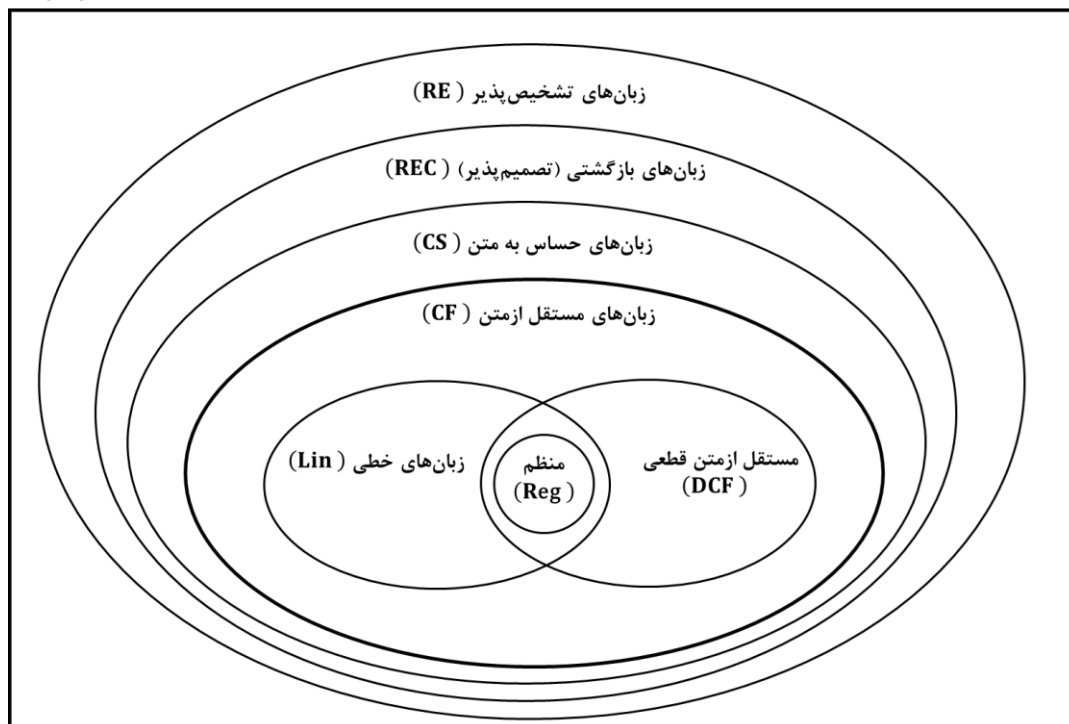
تصميم ناپذير $\text{HALT}_{\text{TM}} = \{ \langle M, w \rangle \mid M \text{ is turing and halts for } w \}$

فصل دهم: تصمیم‌پذیری، کاهش مساله

جلسه بیست و چهارم بخش دو:

مساله **Halt** مسائل معروف تصمیم‌پذیر در هر رده از زبان‌ها



$P(\Sigma^*)$


قضیه رایس: (Rices theorem)

زبان هایی که دارای تورینگ پذیرنده هستند، همان زبان های دارای گرامر بدون محدودیت هستند، همان زبان های تشخیص پذیر هستند. این رده را به عنوان مجموعه مرجع در نظر بگیرید. ویژگی (property) غیر بدیهی مانند P را در نظر بگیریم.

قضیه :

مجموعه زبان های تشخیص پذیری که ویژگی P را دارند، مجموعه ای تصمیم ناپذیر است:

$$A_P = \{ \langle M \rangle \mid M \text{ is turing and } L(M) \text{ has property } P \}$$

مثال: مجموعه $\langle M, w \rangle$ هایی که M ماشین تورینگ و w یک رشته است زبانی تشخیص پذیر است. ویژگی P را ویژگی عضویت w در $L(M)$ در نظر بگیرید. مجموعه زوج مرتب هایی که ویژگی را دارند تصمیم ناپذیر است. یعنی این مجموعه تصمیم ناپذیر است.

$$A_{TM} = \{ \langle M, w \rangle \mid M \text{ is turing and } w \in L(M) \}$$

نتایج مهم قضیه رایس:

در خانواده زبان های دارای ماشین تورینگ پذیرنده ، این مساله ها تصمیم پذیر نیستند:

- مساله عضویت W.
- مساله تهی بودن.
- مساله متناهی بودن.
- مساله جامع بودن.
- مساله منظم بودن.
- مساله مستقل از متن بودن.
- مساله حساس به متن بودن.
- مساله مساوی بودن دو زبان.
- مساله شمول (زیرمجموعه بودن).

کاهش مساله A به مساله B:

اگر تابع محاسبه‌پذیری مانند f موجود باشد که هر نمونه از A را به یک نمونه از B تبدیل کند،

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

می‌گوییم مساله A به مساله B قابل کاهش است:

$$A \leq_m B$$

نتایج حاصل از کاهش مساله A به مساله B:

اگر مساله A به مساله B قابل کاهش باشد ($A \leq_m B$) آنگاه در یک بیان ساده، B حداقل به اندازه A سخت است.

در نتیجه:



B تصمیم‌پذیر باشد آنگاه A تصمیم‌پذیر است.

A تصمیم‌ناپذیر باشد آنگاه B تصمیم‌ناپذیر است.

مساله توقف: (halting theorem)

می‌خواهیم نشان دهیم مجموعه $HALT_{TM}$ تصمیم‌ناپذیر است. یعنی الگوریتمی وجود ندارد که توقف یا عدم توقف ماشین تورینگ M برای ورودی w را تشخیص دهد.

برای اثبات این مطلب نشان می‌دهیم A_{TM} به $HALT_{TM}$ قابل کاهش است.

$$A_{TM} \leq_m HALT_{TM}$$

در واقع ارتباط بین مساله‌ها چنین است:

$$A_{TM} = \{ \langle M, w \rangle \mid M \text{ is turing and } w \in L(M) \}$$

$$HALT_{TM} = \{ \langle M, w \rangle \mid M \text{ is turing and halts for } w \}$$

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

$$A_{TM} \leq_m HALT_{TM}$$

$$A_{TM} \leq_m EMPTY_{TM}$$

برهان خلف: فرض کنیم $HALT_{TM}$ تصمیم‌پذیر باشد. پس الگوریتم یا ابزاری مانند S وجود دارد که برای هر جفت $\langle M, w \rangle$ مشخص می‌کند که آیا M برای ورودی w متوقف می‌شود یا خیر.

$$S(\langle M, w \rangle) = \begin{cases} \text{accept} & \langle M \text{ متوقف می‌شود برای } w \rangle \\ \text{reject} & \langle M \text{ متوقف نمی‌شود برای } w \rangle \end{cases}$$

اکنون یک الگوریتم برای مساله عضویت w در $L(M)$ به این شکل طراحی می‌کنیم:
ابتدا $\langle M, w \rangle$ را به S می‌دهیم، اگر S نشان دهد که ماشین متوقف نمی‌شود، معلوم می‌شود که $w \notin L(M)$

اگر S نشان دهد که ماشین متوقف می‌شود، رشته w را به ماشین می‌دهیم در مدت زمان متناهی معلوم می‌شود که $w \in L(M)$ یا $w \notin L(M)$.
به این ترتیب A_{TM} تصمیم‌پذیر می‌شود که تناقض است.

:

چند مساله معروف و نوع آنها:

ردیف	زبان	توصیف	کلاس
۱	EQ_{CFG}	$(i,j) \subseteq N \times N$ s.t. $L(PDA_i) = L(PDA_j)$ برابری دو گرامر مستقل از متن	Co-RE
۲	$Co - EQ_{CFG}$	$(i,j) \notin EQ_{CFG}$ نابرابری دو گرامر مستقل از متن	RE
۳	A_{TM}	$(i,j) \subseteq N \times N$ s.t. $M_i(j)$ halts accepts تعلق یک رشته به یک ماشین تورینگ	RE
۴	$Co - A_{TM}$	$(i,j) \notin HALT$ عدم تعلق یک رشته به یک ماشین تورینگ	Co-RE
۵	$HALT_{TM}$	$(i,j) \subseteq N \times N$ s.t. $M_i(j)$ halts توقف یک ماشین تورینگ روی رشته ورودی	RE
۶	$Co - HALT_{TM}$	$(i,j) \notin HALT$ عدم توقف یک ماشین تورینگ روی رشته ورودی	Co-RE
۷	E_{TM}	$i \subseteq N$ s.t. $L(M_i) = \emptyset$ تهی بودن زبان یک ماشین تورینگ	Co-RE
۸	$Co - E_{TM}$	$i \notin E_{TM}$ تهی نبودن زبان یک ماشین تورینگ	RE
۹	EQ_{TM}	$(i,j) \subseteq N \times N$ s.t. $L(M_i) = L(M_j)$ برابری دو ماشین تورینگ	Non-RE
۱۰	$Co - EQ_{TM}$	$(i,j) \notin EQ_{TM}$ نابرابری دو ماشین تورینگ	Non-RE