مدرسان شریف

فصل اول

«زبان‌های منظم»

نظریه زبان‌ها و ماشین‌ها، چگونگی ساخت یک زبان صوری شامل قواعد آن و ویژگی‌ها و امکانات زبانی که ارائه می‌دهد، را بررسی می‌کند. منظور از زبان صوری، شکل ظاهری و مستقل از گفتار یک زبان است. نمونه این نوع زبان را می‌توان در دستگاه‌های ریاضی یافت. به‌طور مثال دستگاه اعداد طبیعی یک زبان صوری است که دارای قواعد ساخت رشته‌های عددی می‌باشد. در این زبان هر عدد طبیعی با یکی از قواعد زیر تولید می‌شود:

$1:0\in L_N \qquad 2:x\in L_N \text{ for all } x\in \text{Variable} \qquad 3:s(x)\in L_N \text{ for all } x\in L_N \qquad 4:x+y\in L_N \text{ for all } x,y\in L_N$

برای نمونه، رشته‌های عددی s(s(s(0))) یا s(s(0))+s(s(s(0))) با این قواعد تولید می‌شوند. دقت کنید در اینجا ما می‌خواهیم اعداد طبیعی را با یک ساختار رشته‌ای بسازیم، از این رو رشته 0 که کمترین طول یعنی یک دارد را رشته ابتدایی گرفته و زبان خود را با 0 شروع می‌کنیم. در اینجا 0 رشته‌ای با طول یک است و مفهوم عددی صفر را ندارد. از آنجا که طبق قاعدهٔ 1 رشته $0\in L_N$ است قاعده 3 می‌تواند روی آن اجرا شود که با سه بار تکرار این قاعده رشته s(s(s(0))) ساخته می‌شود. ساخت رشته دوم نیز قاعده 4 را روی s(s(s(0))) و s(0) استفاده می‌کند و پس از آن قاعده 3 را مجدد روی s(0)+s(s(s(0))) بکار می‌برد. یک زبان می‌تواند پردازش‌های متنوعی روی رشته‌ها تعریف کند. به نظر می‌رسد برای یک زبان طبیعی مانند فارسی، انگلیسی، چینی و غیره نیز می‌توان یک توصیف صوری ارائه داد. جمله در یک زبان طبیعی از تعدادی کلمه و کلمه از تعدادی حرف ساخته می‌شود. یک واحد بزرگتر مانند پاراگراف، شامل جملات متعددی است. پس هر زبان طبیعی هم برخوردار از یک مجموعه قواعد است که توسط این قواعد آن زبان تولید می‌شود. اگر یک زبان را به محدوده ساختار صوری آن کاهش دهیم بسیاری از امکانات و ابزارهای آن زبان را از دست می‌دهیم ولی درعوض دقت و تقویت بافت‌های زبانی را در اختیار خواهیم داشت. در این فصل، ابتدا مفاهیم پایه‌ای نظریه زبان‌ها و ماشین‌ها را بررسی کرده، سپس زبان‌های منظم که ساده‌ترین نوع زبان‌های صوری هستند را معرفی می‌کنیم. در پایان ماشین متناهی قطعی مورد بحث قرار می‌گیرد که می‌تواند به‌طور خودکار زبان‌های منظم را بسازند.

مفاهیم پایه

الف ـ مجموعه‌ها: گروهی از اعضا است که هیچ نوع ساختاری ندارد بلکه صرفاً عضو مجموعه هستند. مجموعه‌ها به دو گروه تقسیم می‌شوند: شمارا (قابل شمارش) و ناشمارا (غیرقابل شمارش). مجموعه A را قابل شمارش می‌گویند اگر یک تابع یک‌به‌یک مانند f از A به N (مجموعه اعداد طبیعی) وجود داشته باشد، در غیر این صورت A ناشمارا (غیرقابل شمارش) است. خود مجموعه‌های شمارا (قابل شمارش) نیز در دو دسته متناهی و نامتناهی قرار می‌گیرند. برای مثال، مجموعه الفبای فارسی، مجموعه‌ای قابل شمارش و متناهی است؛ اعداد طبیعی (N) قابل شمارش و نامتناهی بوده و در نهایت اعداد حقیقی (R) غیرقابل شمارش هستند.

ب ـ الفبای زبان: الفبا مجموعه متناهی و ناتهی از نمادهایی است که هیچ کدام به عناصر دیگری در الفبا تجزیه نمی‌شوند. حروف الفبا، اجزا بنیادین ساخت رشته‌ها هستند. مجموعه الفبای یک زبان را با Σ نشان می‌دهیم. در زبان‌های کامپیوتری می‌توان الفبای ترکیبی هم داشت. این عناصر الفبا را به خاطر غیر قابل تجزیه بودن، عناصر پایانی نیز می‌نامند.

ج ـ رشته‌ها: رشته دنباله‌ای متناهی از عناصر الفبای یک زبان است. مثلاً 1001 رشته‌ای است که روی الفبای $\sum=\{0,1\}$ تعریف می‌شود. مجموعه رشته‌ها روی الفبای Σ را با Σ^* نشان می‌دهیم. اگر Σ تک عضوی باشد Σ^* مجموعه‌ای نامتناهی شمارا و اگر Σ بیش از یک عضو داشته باشد، ناشماراست. از u ، v و w برای نام بردن عبارات و رشته‌ها استفاده می‌شود. یک رشته که بخشی از آن با یک متغیر مانند x جایگزین شود، شبه رشته نامیده می‌شود. شبه رشته با شبه جمله تفاوت دارد. به طور مثال 1001x01 یک شبه رشته است اما متغیر u و v که به یک رشته اشاره می‌کنند را شبه جمله می‌گویند. طول رشته w با |w| و معکوس آن را با w^R نشان می‌دهند. رشته تهی رشته‌ای به طول صفر است که با λ مشخص می‌گردد.

نکته ۱: با فرض متناهی بودن الفبای Σ، مجموعه‌های Σ^* و Σ^+ همواره نامتناهی است؛ زیرا طول رشته‌ها در این مجموعه‌ها محدودیتی روی Σ ندارد.

د ـ زبان: به‌طور کلی یک زبان روی الفبای Σ به‌صورت زیرمجموعه‌ای از Σ^* تعریف شده و به هر رشته در زبان L، «جمله» یا «عبارت» می‌گویند. یک زبان با مجموعه‌ای از رشته‌ها که جمله‌های زبان هستند، تعریف می‌شود. برای الفبای Σ یک زبان L، زیر مجموعه‌ای از Σ^* است. مجموعه زبان‌های ممکن

روی الفبای Σ با Λ نشان می‌دهیم که در اینجا $\Lambda = \text{Power}(\Sigma^*)$ می‌باشد. مجموعه زبان‌های ممکن روی الفبای ناتهی Σ یعنی Λ ، مجموعه‌ای نامتناهی و غیرقابل شمارش است.

برای الفبای $\Sigma = \{a,b\}$ داریم: $\Sigma^* = \{\lambda, a, ab, aa, aab, abb, aaa, \cdots\}$. در اینجا مجموعه $A = \{a, ab, bb\}$ یک زبان روی الفبای Σ است. از آن‌جا که تعداد رشته‌های زبان A متناهی است، به A زبان متناهی می‌گویند. اما در همین نمونه فوق مجموعه‌ی $X = \{a^n b^n \mid n \geq 0\}$ یک زبان نامتناهی روی اَلفبای Σ است.

❖ **تعریف ۱: عملگرهای رشته**

عملگرهای مختلفی روی هر رشته تعریف می‌شود؛ از جمله: الحاق $(u \bullet v)$، توانی (u^n)، وارون (u^R) و بستار یا closure (u^*) که هر کدام عمل خاصی را روی رشته‌ها تعریف می‌کنند.

الحاق y به x به معنای وقوع y بلافاصله پس از x است. عمل الحاق، خاصیت شرکت‌پذیری داشته اما خاصیت جابجایی ندارد.

منظور از (u^n) کنار هم قرار دادن u به تعداد n بار است و این عملگر، خاصیت پخشی ندارد یعنی $(ur)^n \neq u^n r^n$.

عملگر وارون به صورت بازگشتی به شکل زیر تعریف می‌شود:

۱ـ اگر $u = a \in \Sigma$ باشد در این صورت $u^R = a^R = a$ است.

۲ـ اگر $u = av$ که $a \in \Sigma$ و $v \in \Sigma^*$ باشد در این صورت $u^R = v^R a$ است.

برای دو رشته v و u ، خواهیم داشت: $(uv)^R = v^R u^R$.

عملگر x^* عبارت است از تکرار صفر مرتبه یا بیشتر رشته یا الفبای x ، که به شکل $\bigcup_{i=0}^{\infty} x^i$ بیان می‌شود. عملگر x^+ تکرار یک مرتبه یا بیشتر عبارت x است. عملگرهای $*$ و $+$ نیز خاصیت پخشی روی الحاق ندارند.

نکته ۲: اگر $v = uxy$ باشد آن‌گاه x یک زیررشته از v است. اگر $u = \lambda$ باشد x پیشوندی برای v است و اگر $y = \lambda$ باشد x یک پسوند برای v محسوب می‌شود. هر شبه‌جمله فقط یک پیشوند دارد و هر جمله به طول n دارای $n+1$ پیشوند است. به‌طور مثال برای رشته v پیشوندهای λ، u و ux وجود دارند.

نکته ۳: هرگاه داشته باشیم $W = W^R$ در این صورت رشته W از دو طرف یکسان نوشته می‌شود و به آن رشته آیینه‌ای یا متقارن (Palindrome) می‌گویند؛ مانند رشته "RADAR"و "KADAK".

عملگرهای زبان

روی زبان‌های ساخته شده از الفبا Σ، می‌توان عملگرهای زیادی را تعریف نمود. این عملگرها با پردازش کلمات زبان، کلمات جدیدی را می‌سازند و در نتیجه ممکن است زبان جدیدی ایجاد کنند. از آنجا که هر زبان، در واقع یک مجموعه است، عملگرهای مجموعه‌ها مانند اجتماع، اشتراک، تفاضل نامتقارن و از این قبیل عملگرها نیز روی زبان‌ها قابل تعریف می‌باشند؛ اما تمام عملگرهای زبان این گونه نیستند. برای مثال، عملگر تقسیم (/)، رشته‌های یک زبان را بر اساس پسوندهایی که در زبان دیگری دارند، برش می‌زند و خارج قسمت‌ها را در زبان جدیدی قرار می‌دهد. نمونه‌های متنوعی از عملگرهای زبانی وجود دارد که پردازش‌های مختلفی را روی زبان انجام می‌دهند.

❖ **تعریف ۲: عملگرهای زبان**

عملگرهای پایه‌ای زبان‌های صوری را در ادامه معرفی می‌نماییم.

الف) عملگر اجتماع و اشتراک: اگر L_1 و L_2 دو زبان روی الفبای Σ باشند، عملگر اجتماع و اشتراک روی این دو به صورت زیر است:

$$L_1 \cup L_2 = \{w \in \Sigma^* \mid w \in L_1 \text{ or } w \in L_2\}$$

$$L_1 \cap L_2 = \{w \in \Sigma^* \mid w \in L_1 \text{ and } w \in L_2\}$$

ب) عملگر الحاق: زبان جدید با الحاق کلمات زبان L_1 در ابتدای کلمات زبان L_2 ساخته می‌شود:

$$L_1 \bullet L_2 = \{w \in \Sigma^*, w = u.v \mid u \in L_1 \text{ and } v \in L_2\}$$

ج) عملگر متمم: مجموعه کلمات غیر عضو در زبان را به صورت یک زبان جدید عرضه می‌کند: $\overline{(L_1)} = \{w \in \Sigma^* \mid w \notin L_1\}$

مثال ۱: زبان $L = \{aa, bb\}$ را روی الفبای $\sum = \{a,b\}$ در نظر بگیرید. زبان $\bar{L}$ را مشخص کنید؟

☑ **پاسخ:** با توجه به اینکه L دو عضوی بوده و Σ^* تعداد نا متناهی عضو دارد، لذا رشته‌هایی با طول بیش از ۲ و همین طور ab و ba و a و b و λ در متمم L وجود دارد. $\bar{L} = \{\lambda, a, b, ab, ba\} \cup \{w \in \{a,b\}^* \mid \mid w \mid > 2\}$

د) عملگر تقسیم: نسبت راست دو زبان L_1 و L_2 یا عمل تقسیم دو زبان، شامل مجموعه پیشوندهایی از L_1 است که بخش پایانی آن‌ها در L_2 قرار دارد. این مجموعه به‌صورت یک زبان جدید ارائه می‌شود:

$$L_1 / L_2 = \{w \in \Sigma^* \mid \exists v \in \Sigma^* \text{ that } wv \in L_1 \text{ and } v \in L_2\}$$

هـ) عملگر وارون: از مجموعه کلمات وارون‌سازی شده زبان جدیدی می‌سازد: $L^R = \{w \in \Sigma^* \mid w^R \in L\}$

و) عملگر بستار، بعلاوه و توانی: عملگر بستار تمام کلمات ممکن از طریق بستار ترکیبی با طول مختلف را روی کلمات زبان ایجاد می‌کند. عملگر توانی این کار را تنها در طول خاص، مثلاً ۳ مرتبه الحاق تمامی کلمات یک زبان، انجام می‌دهد. عملگر به‌علاوه هم فقط روی حالت تکرار صفر یک عبارت تعریف نمی‌شود:

$L_1^* = \{w \in \Sigma^* \mid w = u_1 u_2 ... u_n \text{ s.t. } u_i \in L_1 \text{ and } n \geq 0\}$ (منظور از s.t. همان Such that می‌باشد)

زبان L_1^+ به‌صورت $L_1^+ = L_1^* - \{\lambda\}$ تعریف می‌شود. حتی زمانی که $L = \varnothing$ باشد باز هم این تعریف کارائی دارد (به مفهوم تفاضل مجموعه‌ها توجه کنید).

تعریف L_1^t به صورت $\{\omega \in \Sigma^* \mid \omega = u_1 u_2 ... u_n \text{ s.t } u_i \in L_1 \text{and } n \geq 1\}$ نیز عیناً معادل دیگری برای آن ارائه می‌دهد. عملگر توانی نیز به‌صورت زیر است.

$$L_1^n = \{w \in \Sigma^* \mid w = u_1 u_2 ... u_n \text{ s.t. } u_i \in L_1\}$$

اگر $L = \{a^n b^n \mid n \geq 0\}$ فرض شود، زبان L^R و L^2 به این صورت بیان می‌شود:

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

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

ز) عملگر تفریق: این عملگر، اشتراک یک زبان و متمم زبانی دیگر را در زبان جدیدی می‌آورد.

$$L_1 \setminus L_2 = \{w \in \Sigma^* \mid w \in L_1 \text{ and } w \notin L_2\}$$

➢ **توجه:** هر یک از علائم $-$ و $\setminus$ به عنوان نماد عملگر تفریق استفاده می‌شوند.

مثال ۲: با توجه به دو زبان $L_2 = \{0,1,11\}$ و $L_1 = \{0,01,111\}$ برای عملگرهای تقسیم و تفاضل خواهیم داشت:

☑ **پاسخ:**

$$L_1 \setminus L_2 = \{01,111\} \qquad L_1 / L_2 = \{\lambda, 0, 1, 11\}$$

$$L_2 / L_1 = \{\lambda\} \qquad L_2 \setminus L_1 = \{1,\ 11\}$$

مثال ۳: زبان L را روی الفبای $\sum = \{a,b\}$ بیابید بطوری‌که $(\bar{L})^+ = \bar{L}^+$ باشد و سپس این موضوع را ثابت کنید؟

☑ **پاسخ:** فرض کنید $L = \phi$ باشد، در این‌صورت تساوی‌های روبه‌رو برقرار است.

$$\left.\begin{matrix} \bar{L} = \Sigma^* \\ L^+ = \phi \end{matrix}\right\} \Rightarrow (\bar{L})^+ = \bar{L}^+ = \Sigma^*$$

هر یک از زبان‌های $\{a^+\}, \{b^*\}, \{a^*\}, \Sigma^*, \lambda, \Sigma^+$ و $\{b^+\}$ نیز می‌توانند پاسخ‌های دیگر این سوال باشند.

عملگرهای زبانی نقش مهمی در پردازش کلمات یک زبان دارند. ممکن است یک عملگر ویژگی‌هایی چون شرکت‌پذیری، جابه‌جایی، عضو خنثی و عضو وارون را داشته باشد و یا از آن برخوردار نباشد. این موضوع روی قدرت عملگر اثر گذاشته و ممکن است کارایی آن را محدود کند. در ادامه مشخص می‌کنیم هر عملگر چه ویژگی‌ها و خواصی دارد.

مثال ۴: کدام‌یک از عملگرهای زبان، ویژگی شرکت‌پذیری، جابجایی، عضو خنثی و عضو وارون دارند؟

☑ **پاسخ: الف ــ** اگر L_1، L_2 و L_3 عضو Λ باشند برای عملگرهای اجتماع و اشتراک خاصیت شرکت‌پذیری و جابه‌جایی برقرار است. همچنین برای عضو خنثی داریم:

$L_1 \cup \varnothing = L_1$ عضو خنثی اجتماع $\varnothing$

$L_1 \cap \Sigma^* = L_1$ عضو خنثی اشتراک Σ^*

$L_1 \cup ? = \varnothing$ تحت اجتماع هر رشته عضو وارون وجود ندارد

$L_1 \cap ? = \Sigma^*$ همینطور عضو وارون تحت اجتماع نیز وجود ندارد.

ب ــ عملگر الحاق، شرکت‌پذیر است ولی ویژگی جابه‌جایی ندارد. برای مثال اگر $L_1 = \{0,10\}$ و $L_2 = \{11,1\}$ داریم:

$$\left.\begin{matrix} L_1 \bullet L_2 = \{011,01,1011,101\} \\ L_2 \bullet L_1 = \{110,1110,10\} \end{matrix}\right\} \Rightarrow L_1 \bullet L_2 \neq L_2 \bullet L_1$$

همچنین عضو خنثی برای عملگر الحاق λ است، با این حال عملگر الحاق عضو وارون ندارد.

$$L_1 \bullet \{\lambda\} = L_1$$

$$L_1 \bullet ? = \{\lambda\}$$

ج ــ عملگر متمم به تنهایی هیچ ویژگی ندارد. چرا که یک عملگر تک‌موضعی است و فقط یک پارامتر دارد.

د ـ عملگر تقسیم نه شرکت‌پذیر است نه خاصیت جابه‌جایی دارد. برای مثال با فرض $L_1=\{1101,1010\}, L_2=\{01,10\}$ و $L_3=\{10,1\}$ داریم:

$$\left.\begin{array}{l} L_1/L_2=\{11,10\} \\ (L_1/L_2)/L_3=\{\lambda,1\} \\ L_2/L_3=\{\lambda,0\} \\ L_1/(L_2/L_3)=\{1101,1010,101\} \end{array}\right\} \Rightarrow L_1/(L_2/L_3)\neq(L_1/L_2)/L_3 \Rightarrow \text{پس شرکت‌پذیر نیست}$$

همین‌طور این عملگر جابجایی نیز نمی‌باشد، زیرا به‌طور مثال داریم: $L_1/L_2=\{11,10\}\neq\varnothing=L_2/L_1$

برای عملگر تقسیم زبان $\{\lambda\}$ عضو خنثی به حساب می‌آید، زیرا برای هر زبان $L\in A$ داریم $L/\{\lambda\}=L$ است. یافتن عضو وارون عملگر تقسیم، وقتی‌که L متناهی باشد، راحت است. اگر $w\in L$ که برای هر $w'\in L$ داشته باشیم $|w'|\leq|w|$ آن‌گاه $\{w\}$ یک وارون برای زبان L تحت عمل تقسیم می‌سازد، زیرا $L/\{w\}=\{\lambda\}$ است. طبیعی است که وارون تقسیم منحصربه‌فرد نیست و می‌تواند انتخاب‌های متعددی داشته باشد.

لذا به‌طور قطع وارون‌های زیادی برای یک زبان متناهی وجود دارد. با این حال، وقتی زبان L نامتناهی است وجود عضو وارون زبانی برای عملگر تقسیم قطعی نیست. برای مثال، زبان‌های L_1 و L_2 که در زیر تعریف شده‌اند، نامتناهی هستند اما L_1 وارون داشته ولی L_2 وارون ندارد. زبان $\{a\}$ برای زبان L_1 یک وارون می‌سازد که با تقسیم L_1 بر $\{a\}$ عضو همانی تقسیم، یعنی $\{\lambda\}$ ساخته می‌شود. اما برای زبان L_2 چنین زبانی وجود ندارد.

$L_1=\{ab^n\mid n\geq 0\}\quad L_1/\{a\}=\{\lambda\}$

$L_2=\{a^nb\mid n\geq 0\}\quad L_2/?=\{\lambda\}$

ز ـ عملگر تفریق، شرکت‌پذیر نیست؛ چرا که بنا بر تعریف داریم:

$$w\in L_1\setminus(L_2\setminus L_3)\Rightarrow \left.\begin{array}{l} w\in L_1 \\ w\notin L_2\setminus L_3 \end{array}\right\}\Rightarrow \text{ممکن است } w\in L_1\cap L_3 \text{ باشد ولی عضو } L_2 \text{ نباشد}$$

در صورتی‌که برای حالت دیگر داریم $w\in(L_1\setminus L_2)\setminus L_3$ از این رو $w\notin L_3$ است. پس داریم: $L_1\setminus(L_2\setminus L_3)\neq(L_1\setminus L_2)\setminus L_3$

ویژگی جابه‌جایی نیز برای عملگر تفریق برقرار نیست؛ چرا که این حالت با مفهوم تفریق، تناقض دارد. عضو همانی تفریق هم $\varnothing$ است؛ چرا که برای هر L داریم $L\setminus\varnothing=L$. از طرفی عضو وارون هر زبان L خود آن زبان است؛ چراکه $L\setminus L=\varnothing$ و به طور طبیعی هر زبان دیگر L_1 که $L\subset L_1$ نیز یک وارون برای زبان L به حساب می‌آید.

هـ ـ عملگر وارون تک‌موضعی است و یک پارامتر بیشتر ندارد لذا شرکت‌پذیری و جابه‌جایی برای آن تعریف نمی‌شود.

و ـ عملگرهای بستار، به‌علاوه و توانی نیز تک‌موضعی هستند و به همین ترتیب این ویژگی‌ها برای آن‌ها تعریف نمی‌شود.

مثال ۵: کدام‌یک از عملگرهای تک‌موضعی متمم، وارون، بستار، به‌علاوه و توانی، روی عملگرهای اجتماع، اشتراک، الحاق و تقسیم، توزیع‌پذیر هستند؟ توضیح: اگر $F(E(x,y))=E(F(x),\ F(y))$ باشد، عملگر $F(x)$ را روی $E(x,y)$ توزیع‌پذیر می‌گویند.

☑ **پاسخ: الف ـ** بنابر قانون دمورگان، عملگر متمم روی اجتماع و اشتراک توزیع‌پذیر نیست، زیرا متمم اشتراک‌ها، اجتماع متمم‌ها است $\overline{(A_1\cap A_2\cap A_3)}=(\bar{A}_1\cup\bar{A}_2\cup\bar{A}_3)$ و متمم اجتماع‌ها، اشتراک متمم‌ها $\overline{(A_1\cup A_2\cup A_3)}=(\bar{A}_1\cap\bar{A}_2\cap\bar{A}_3)$ است. همچنین عملگر وارون، توزیع‌پذیر است اما عملگر بستار، نه روی اشتراک و نه روی اجتماع توزیع‌پذیر نیست. مثال زیر این موضوع را نشان می‌دهد:

$L_1=\{101,010\}$

$L_2=\{11,00\}$

$$\begin{cases} 10111\in(L_1\cup L_2)^* \\ 10111\notin(L_1^*\cup L_2^*) \end{cases}\Rightarrow(L_1\cup L_2)^*\neq(L_1^*\cup L_2^*)$$

و برای اشتراک فرض کنید $L_1=\{0,01\}$ و $L_2\{10,00)$ باشد لذا:

$L_1\cap L_2=\varnothing\rightarrow(L_1\cap L_2)^*=\{\lambda\}$

$L_1^*\cap L_2^*=\{10,00\}^*\cap\{0,01\}^*=\{0010,\ldots\}\neq\{\lambda\}$

وضعیت عملگر به‌علاوه و توانی روی اجتماع و اشتراک، مشابه عملگر بستار است و روی اشتراک و اجتماع توزیع‌پذیر نیست.

ب ـ عملگر متمم، روی عملگر الحاق توزیع‌پذیر نیست. برای مثال برای زبان‌های $L_1=\{\lambda\}$ و $L_2=\{0\}$ داریم:

$0\in L_1\bullet L_2\rightarrow 0\notin\overline{L_1\bullet L_2}$

$0\in\overline{L_1}\ \text{and}\ \lambda\in\overline{L_2}\Rightarrow 0\in\overline{L_1}\bullet\overline{L_2}\Rightarrow\overline{L_1\bullet L_2}\neq\overline{L_1}\bullet\overline{L_2}$

ج ـ عملگر وارون نیز روی عملگر الحاق توزیع‌پذیر نیست؛ چراکه اگر $w=uv$ که $w\in L_1\bullet L_2$ آن‌گاه $w^R=u^Rv^R$ که لزوماً عضو $L_1^R\bullet L_2^R$ نیست، از این‌رو که قسمت اول خود را از L_2^R می‌گیرد.

عملگر بستار، عملگر به‌علاوه و عملگر توانی هر سه توزیع‌پذیری روی عملگر الحاق را ندارند. اگر $L_1 = \{1\}$ و $L_2 = \{0\}$ در نظر گرفته شود:

$$(L_1 \bullet L_2)^* = \{10\}^* = \{\lambda, 10, 1010, \ldots\} \neq \{1,11,\ldots\}.\{0,00,\ldots\}$$

د ـ عملگر تقسیم:

توزیع‌پذیری عملگر متمم روی عمل تقسیم به راحتی رد می‌شود. برای این موضوع فرض کنید $L_1 = \Sigma^+$ و $L_2 = \{\lambda\}$ باشد، در این صورت $L_1 / L_2 = \Sigma^+$. حال داریم $(\overline{L_1 / L_2}) = \{\lambda\}$ در صورتی‌که چون $\overline{L_1} = \{\lambda\}$ و $\overline{L_2} = \Sigma^+$ داریم $\overline{L_1} / \overline{L_2} = \varnothing$ و نتیجه می‌گیریم تساوی برقرار نیست.

برای اینکه نشان دهیم عملگر وارون روی عملگر تقسیم توزیع‌پذیر نیست، در ادامه یک مثال را ارائه می‌کنیم:

$$\begin{cases} L_1 = \{1,10,100,1010\} \\ L_2 = \{1\} \end{cases} \Rightarrow L_1 / L_2 = \{\lambda\} \Rightarrow (L_1 / L_2)^R = \{\lambda\}$$

$$\begin{cases} L_1^R = \{1,01,001,0101\} \\ L_2^R = \{1\} \end{cases} \Rightarrow L_1^R / L_2^R = \{\lambda, 0, 00, 010\}$$

هـ ـ عملگرهای بستار، بعلاوه و توانی نیز روی عملگر تقسیم توزیع‌پذیر نیستند:

$$\begin{cases} L_1 = \{1\} \\ L_2 = \{11\} \end{cases} \Rightarrow \frac{L_1}{L_2} = \varnothing \Rightarrow (\frac{L_1}{L_2})^* = \{\lambda\}$$

$$\begin{cases} L_1^* = \{1,11,111,\ldots\} \\ L_2^* = \{11,1111,\ldots\} \end{cases} \Rightarrow \frac{L_1^*}{L_2^*} = \{\lambda, 1, 11, \ldots\}$$

و ـ عملگرهای تک موضعی $\overline{()}$، $()^R$، $()^*$، $()^+$ و $()^n$ روی تفریق $(\setminus)$ در معادلات زیر برقرار هستند. فرض کنید L_1 و L_2 همان تعریف قبلی را دارند، اکنون عملگرهای تک‌موضعی روی عملگر تفریق معادلات زیر را تشکیل می‌دهند.

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

$$(L_1 \setminus L_2)^R = L_1^R \setminus L_2^R$$

$$(L_1 \setminus L_2)^* = L_1^* \setminus (L_1 \cap L_2)^* \cup \{\lambda\}$$

نشان دادن این موضوع برای عملگر توانی و عملگر به‌علاوه مشابه عملگر بستار است، اثبات هر یک، از طریق نمودار وِن برای مجموعه‌ها قابل انجام است.

مثال ۶: آیا عملگرهای اجتماع، اشتراک، الحاق و تقسیم روی هم توزیع‌پذیرند؟ توضیح: یک عملگر دو موضعی $F(x,y)$ روی عملگر دو موضعی $E(x,y)$ توزیع‌پذیر است اگر داشته باشیم: $F(x,E(y,z)) = E(F(x,y),F(x,z))$.

پاسخ:

۱ـ توزیع‌پذیری عملگر اجتماع روی عملگر اشتراک و برعکس: بنابر قانون دمورگان برقرار است.

۲ـ توزیع‌پذیری عملگر الحاق روی عملگرهای اجتماع و اشتراک: ثابت می‌شود که عمل الحاق روی اجتماع، توزیع‌پذیر بوده اما روی اشتراک توزیع‌پذیر نیست. به این صورت که اگر L_1، L_2 و L_3 سه زبان روی الفبای Σ باشند پس:

$$L_1 \bullet (L_2 \cup L_3) = (L_1 \bullet L_2 \cup L_1 \bullet L_3)$$

$$w \in (L_1 \bullet (L_2 \cup L_3) \leftrightarrow w = uv \;\; u \in L_1 \;\; \& \;\; v \in L_2 \;\; or \;\; L_3 \leftrightarrow w \in L_1 \bullet L_2 \, or \, w \in L_1 \bullet L_3$$

برای نقض توزیع‌پذیری روی اشتراک فرض کنید:

$$L_1 = \{1,10\} \;\; L_2 = \{01\} \;\; L_3 = \{1,0\}$$

$$L_2 \cap L_3 = \varnothing \rightarrow L_1 \bullet (L_2 \cap L_3) = L_1 \bullet \varnothing = \varnothing$$

$$\begin{cases} L_1 \bullet L_2 = \{101,1001\} \\ L_1 \bullet L_3 = \{11,10,101,100\} \end{cases} \Rightarrow L_1 \bullet L_2 \cap L_1 \bullet L_3 = \{101\} \neq \varnothing$$

پس داریم:

$$L_1 \bullet (L_2 \cap L_3) \neq (L_1 \bullet L_2) \cap (L_1 \bullet L_3)$$

۳ـ توزیع‌پذیری عملگر تقسیم روی اجتماع و اشتراک نیز مشابه الحاق در حالت اجتماع، توزیع‌پذیر و در حالت اشتراک توزیع‌ناپذیراست. به این صورت که برای سه زبان L_1، L_2 و L_3 روی الفبای Σ داریم:

$$L_1 / (L_2 \cup L_3) = (L_1 / L_2) \cup (L_1 / L_3)$$

اگر داشته باشیم $w \in L_1 / (L_2 \cup L_3)$ پس باید $z = wu \in L$ وجود داشته باشد که $u \in L_1$ یا $u \in L_3$ باشد. یعنی $Z \in L_1 / L_2$ یا $Z \in L_1 / L_3$ باشد. ولی برای اشتراک داریم:

$$L_1 / (L_2 \cap L_3) \neq (L_1 / L_2) \cap (L_1 / L_3)$$

برای تأیید این موضوع فرض کنید L_1، L_2 و L_3 به شکل زیر باشند:

$$L_1=\{10,101\}\quad L_2=\{0\}\quad L_3=\{01\}$$

آنگاه خواهیم داشت:

$$\begin{cases} L_1/(L_2\cap L_3)=L_1/\varnothing=\varnothing \\ L_1/L_2\cap L_1/L_3=\{1\}\cap\{1\}=\{1\}\end{cases}\Rightarrow \varnothing\neq\{1\}$$

۴ـ توزیع‌پذیری عملگر تقسیم روی عملگر الحاق و برعکس:

توزیع‌پذیری تقسیم روی الحاق برقرار نیست؛ چرا که اگر فرض کنیم $L_1=\{10,1\}$ و $L_2=\{0\}$ و $L_3=\{1\}$ در این‌صورت:

$$L_1/(L_2\bullet L_3)=\{10,1\}/\{01\}=\varnothing\neq L_1/L_2\ \bullet\ L_1/L_3=\{1\}\bullet\{\lambda\}=\{1\}$$

برای اثبات برقرار نبودن توزیع عمل الحاق روی تقسیم، می‌توان با فرض L_1، L_2 و L_3 به‌صورت زیر، نتیجه را اثبات کرد:

$$L_3=\{0\},L_2=\{1\},L_1=\{10,1\}$$
$$L_1\bullet(L_2/L_3)=L_1\bullet\varnothing=\varnothing$$
$$(L_1\bullet L_2)/(L_1\bullet L_3)=\{101,11\}/\{1001,101\}=\{\lambda\}$$

مثال ۷: طول کوچک‌ترین رشته متعلق به مجموعهٔ $1^*0+0\{0^i10^{2i}1\,|\,i\geq 0\}^*\cap 01\{0^i10^{2i}1\,|\,i\geq 1\}^*$ کدام است؟

۱) 12 ۲) 18 ۳) 24 ۴) 19

☑ **پاسخ:** گزینه «۴» کوچک‌ترین رشته متعلق به این مجموعه عبارت است از 010010^410^81 که طول آن 19 می‌باشد. دقت کنید در اینجا دو زبان $\{0^i10^{2i}1|i\geq 1\}^*0^*1$ و $01\{0^i10^{2i}1|i\geq 0\}^*$ هر کدام اجرا و بعد اشتراک این دو زبان محاسبه می‌گردد. نتیجه زبانی است که طول کمترین عنصر این زبان باید به‌دست آید. این عنصر باید در هر دو زبان وجود داشته باشد.

مثال ۸: طول دومین رشته کوچک متعلق به مجموعه مثال بالا کدام است؟

۱) 38 ۲) 69 ۳) 48 ۴) 60

☑ **پاسخ:** گزینه «۲» دومین رشته کوچک به فرم مقابل و به طول 69 خواهد بود:

$$S_1:\boxed{010010^410^810^{16}10^{32}1}$$

$$\text{طول رشته } S_1=19+16+1+32+1=69$$

روشن است که برای حل این‌گونه سؤالات باید پیشوندها یا پسوندهای مشترک را از روی ساختار آنها حدس بزنید و بعد محاسبه مشخص رشته‌ای که خواسته شده را دنبال کنید.

مثال ۹: آیا عملگرهای متمم، وارون، بستار، به‌علاوه و توانی، خاصیت جابجایی دارند؟ توضیح: اگر $F(E(x))=E(F(x))$ باشد دو عملگر $E(x),F(x)$ را جابه‌جاپذیر می‌گوییم.

☑ **پاسخ: ۱ـ جابجایی عملگرهای متمم و وارون:**

برای اینکه نشان دهیم برای زبان L، تساوی $(\overline{L})^R=(\overline{L^R})$ برقرار است؛ بنابر تعریف مجموعه‌ها اگر $w\in(\overline{L})^R$ باشد، پس باید $w^R\in\overline{L}$ و در نتیجه $w^R\notin L$ لذا $(w^R)^R\notin L^R$ یعنی $w\notin L^R$ پس $w\in\overline{L^R}$ است. برعکس اگر $w\in\overline{L^R}$ باید $w\notin L^R$ لذا $w^R\notin L$ پس $w^R\in\overline{L}$ و همین‌طور $(w^R)^R\in\overline{L}^R$ یعنی $w\in\overline{L}^R$ است. پس زبان دو طرف با هم یکسان بوده و لذا این دو عملگر جابه‌جاپذیر هستند.

۲ـ جابجایی عملگرهای متمم و بستار (متمم و به‌علاوه، متمم و توانی)

این دو جابه‌جاپذیر نیستند چراکه اگر داشته باشیم $L=\{0,1\}^*$ یعنی $L=\Sigma^*$ در این صورت داریم:

$$\begin{cases}\overline{L}=\varnothing\rightarrow\overline{L}^*=\{\lambda\}\\ L^*=\{0,1\}^*=\Sigma^*\rightarrow\overline{L^*}=\varnothing\end{cases}\Rightarrow\overline{L}^*\neq\overline{L^*}$$

با همین مثال جابه‌جاپذیر نبودن دو حالت دیگر نیز می‌تواند تأیید شود. این کار را به‌عنوان تمرین می‌توانید انجام دهید.

۳ـ وارون و بستار

این دو عملگر برعکس بقیه جابه‌جاپذیر هستند یعنی تساوی $(L^R)^*=(L^*)^R$ برقرار است. برای اثبات برقراری این تساوی، نشان می‌دهیم هر عضو زبان سمت چپ در زبان سمت راست قرار دارد و برعکس. ابتدا طرف اول:

$$w\in(L^R)^*\Rightarrow w=v_1v_2\ldots v_n,v_i\in L^R\quad i\geq n\Rightarrow$$
$$v_i^R\in L\Rightarrow w^R=v_n^R\ldots v_1^R\in L^*\Rightarrow(w^R)^R\in(L^*)^R\Rightarrow w\in L^R$$

و حال برعکس:

$$w \in (L^*)^R \Rightarrow w^R \in L^* \Rightarrow w^R = (v_1 \dots v_n)^R \in L^*$$
$$\Rightarrow v_n^R, \dots, v_1^R \in L \Rightarrow v_1, \dots, v_n \in L^R \Rightarrow w = v_1 \dots v_n \in L^R \Rightarrow w^* \in (L^R)^*$$

مشابه همین روش برای اثبات جابه‌جاپذیر بودن وارون و به‌علاوه و همچنین وارون و توانی انجام می‌شود و از این طریق می‌توان درستی این دو حالت را تأیید کرد.

مثال ۱۰: کدام‌یک از تساوی‌های زیر برقرار است؟

الف) $L_1/(L_2 \cap L_3) = L_1/L_2 \cup L_1/L_3$　　ب) $L_1/(L_2 \bullet L_3) = L_1/L_2 \bullet L_1/L_3$　　ج) $L_1 \bullet (L_2/L_3) = (L_1 \bullet L_2)/L_1 \bullet L_3$

۱) الف و ب　　۲) ب و ج　　۳) الف و ج　　۴) هیچ‌کدام.

☑ **پاسخ:** گزینه «۴» برای رد تساوی عبارت الف، فرض کنید $L_1 = \{10,11\}$ و $L_2 = \{0\}$ و $L_3 = \{1\}$ باشد.

$$L_1/(L_2 \cap L_3) = \{10,11\}/\varnothing = \varnothing \neq \{1\} = \{1\} \cup \{1\} = L_1/L_2 \cup L_1/L_3$$

برای رد تساوی ب نیز با فرض بالا، عبارت روبرو را داریم: $L_1/(L_2 \bullet L_3) = \{10,11\}/\{01\} = \varnothing \neq \{11\} = \{1\} \cup \{1\} = L_1/L_2 \bullet L_1/L_3$

در مثال۶ قسمت ۴ عدم تساوی ج اثبات شده است.

مثال ۱۱: اگر زبان $L_1 = \{11,00\}$ و $L_2 = \{10,01\}$ را داشته باشیم کدام تساوی زیر برقرار خواهد بود؟

۱) $\bullet^R(L_1,L_2) = \bullet(L_1^R, L_2^R)$　　۲) $\bullet^R(L_1,L_2) = \bullet(L_2^R, L_1^R)$

۳) $\bar{\bullet}(L_1,L_2) = \bullet(\bar{L}_1, \bar{L}_2)$　　۴) $\bar{\bullet}(L_1,L_2) = \bullet(\bar{L}_1 \setminus \bar{L}_2, \bar{L}_2 - L_1)$

☑ **پاسخ:** گزینه «۲» تساوی ۱ برای هیچ دوزبانی مگر حالت‌های خیلی خاص برقرار نیست. گزینه درست، تساوی ۲ است زیرا برای هر دوزبانی، از جمله زبان‌های این سؤال برقرار است. گزینه۳ و ۴ تنها در زبان‌های خاصی برقرار است ولی برای هیچ‌کدام از زبان‌های این سؤال، برقرار نمی‌باشد چرا که داریم $\bullet(L_1,L_2) = \{1101,1110,0010,0001\}$ که در نتیجه $1101 \notin \bar{\bullet}(L_1,L_2)$ است اما $1101 \in \bar{L}_1 \setminus L_2$ و $\lambda = \bar{L}_2 - L_1$ می‌باشد، لذا $1101 \in \bullet(\bar{L}_1 \setminus L_2, \bar{L}_2 \setminus L_1)$ است که نشان می‌دهد تساوی ۴ برقرار نیست. همین مثال نیز نامساوی بودن تساوی ۳ را نشان می‌دهد.

مثال ۱۲: کدام‌یک از تساوی‌های زیر برقرار است؟ عملگر \ در اینجا به معنای تفریق است یعنی اگر L_1 و L_2 دو زبان روی الفبای Σ باشند. آن‌گاه:

$$L_1 \setminus L_2 = \{w \in \Sigma^* \mid w \in L_1 \text{ and } w \notin L_2\}$$

۱) $L_1 \setminus (L_2 \bullet L_3) = (L_1 \setminus L_2) \bullet (L_1 \setminus L_3)$　　۲) $L_1 \setminus (L_2 \bullet L_3) = (L_1 \setminus L_2)/(L_1 \setminus L_3)$

۳) $L_1 \setminus (L_2 \cap L_3) = (L_1 \setminus L_2) \cap (L_1 \setminus L_3)$　　۴) $L_1 \setminus (L_2 \cup L_3) = (L_1 \setminus L_2) \cap (L_1 \setminus L_3)$

☑ **پاسخ:** گزینه «۴» این سؤال که توزیع‌پذیری عملگر تقسیم روی عملگرها را بررسی کرده است. برای رد هر یک از تساوی‌ها، یک مثال نقض ارائه می‌شود. با معرفی سه زبان L_1، L_2 و L_3 و ساخت هر یک از طرفین برقرار نبودن تساوی نشان داده می‌شود.

الف)
$$L_1 = \{11,00\}\ L_2 = \{11\}\ L_3 = \{00\}$$
$$L_1 \setminus (L_2 \bullet L_3) = \{11,00\} \neq (L_1 \setminus L_2) \bullet (L_1 \setminus L_3) = \{0011\}$$

ب)
$$L_1 = \{101,010\}\ L_2 = \{101\}\ L_3 = \{010,1\}$$
$$L_1 \setminus (L_2/L_3) = \{101\} \neq (L_1 \setminus L_2)/(L_1 \setminus L_3) = \{\lambda\}$$

ج)
$$L_1 = \{11,00\}\ L_2 = \{11\}\ L_3 = \{00\}$$
$$L_1 \setminus (L_2 \cap L_3) = \{11,00\} \neq (L_1 \setminus L_2) \cap (L_1 \setminus L_3) = \varnothing$$

د) این معادله برقرار است چرا که اگر داشته باشیم:
$$L_1 = \{11,00,10,01\}\ L_2 = \{11,00\}\ L_3 = \{10,01\}$$
$$L_1 \setminus (L_2 \cup L_3) = \varnothing = (L_1 \setminus L_2) \cap (L_1 \setminus L_3)$$

دقت شود که اگر سمت راست معادله، به جای عمل $\cap$ عمل $\cup$ را بگذاریم تساوی دیگر برقرار نیست.

گرامرهای زبان

گرامر زبان، مجموعه‌ای است از قوانین که ساخت رشته‌های متعلق به زبان را تعریف نموده و روند ایجاد رشته‌های زبان را کنترل می‌کند. گرامر فقط نحو (دستور، ساختار) زبان را کنترل می‌کند و روی معنی زبان هیچ کنترلی ندارند. به فرایند ساخت رشته‌های زبان توسط گرامر زبان اشتقاق می‌گویند.

گرامر زبان به‌صورت چهارتایی مرتب $G = (\Sigma, V, R, S)$ بیان می‌شود. در اینجا Σ نشانه الفبای زبان، V علامت مجموعه متغیرها، R شامل قواعد زبان است که به شکل $R \subset (V \cup \Sigma)^+ \times (V \cup \Sigma)^*$ نشان داده می‌شود و S نماد آغازگر اشتقاق زبان است که $S \in V$ می‌باشد. همچنین $L(G)$ زبان تعریف شده توسط گرامر مشخص G می‌باشد. اگر دو گرامر G_1 و G_2 زبان‌های یکسان تولید نمایند آن را معادل می‌گویند، به عبارت دیگر $L(G_1) = L(G_2)$ می‌باشد.

مثال ۱۳: گرامر مقابل با شماره قوانین مشخص شده مفروض است. (علامت | به معنی «یا» می‌باشد)

(1) $E \to E + T.$

(2) $E \to T.$

(3) $T \to (E) \mid 1 \mid 2 \mid 3.$

الف ـ مجموعه‌های Σ، V، R و S را تعیین کنید.

ب ـ آیا رشته $((3)+2)$ متعلق به $L(G)$ است؟

پاسخ: الف) $\Sigma = \{+,(,),1,2,3\} \qquad V = \{E,T\} \qquad S = \{E\}$

ب) $E \overset{2}{\Rightarrow} T \overset{3}{\Rightarrow} (E) \overset{1}{\Rightarrow} (E+T) \overset{3}{\Rightarrow} (E+(E)) \overset{2}{\Rightarrow} (T+(E)) \overset{3}{\Rightarrow} (2+(E)) \overset{2}{\Rightarrow} (2+(T)) \overset{3}{\Rightarrow} (2+(3))$

این رشته، متعلق به زبان گرامر است. از این رو: $E \overset{*}{\Rightarrow} (2+(3))$

این مسأله که چگونه براساس ساختار یک گرامر می‌توان تعداد مراحل اشتقاق یک جمله از گرامر داده شده را پیش از ساخت آن مشخص کرد، بسیار اهمیت دارد. برای این منظور سعی می‌شود گرامرها در شکل‌های استاندارد ایجاد شوند تا تعیین مراحل اشتقاق برای رشته‌ها راحت‌تر صورت گیرد. این شکل‌های استاندارد که به طبقه‌بندی گرامرها ارتباط پیدا می‌کند را در ادامه بیان می‌کنیم.

طبقه‌بندی گرامرها

گرامر زبان‌ها براساس قوانین توصیف‌کننده آن به چهار گروه تقسیم می‌شوند. اگر برای زبانی حداقل یک توصیف گرامری متعلق به گروه‌های زیر وجود داشته باشد، زبان ایجاد شده توسط گرامر مذکور متعلق به آن گروه شناخته می‌شود. آنچه در ادامه به‌عنوان دسته‌بندی زبان‌ها ارائه شده است را طبقه‌بندی چامسکی برای زبان‌های صوری می‌نامند.

الف ـ گرامرهای منظم (Regular) (نوع سه)

گرامرهای منظم یا باقاعده براساس قالب‌های تولید قوانین گرامری که در مقابل آمده است ساخته می‌شوند.

$A \to a \quad ; A \in V, a \in \Sigma$

$A \to aA' ; A, A' \in V, a \in \Sigma$

زبان‌هایی که از این گروه از گرامرها قابل استخراج هستند تماماً توسط ماشین‌های متناهی (DFA) که رشته‌های زبان را به‌طور خودکار می‌سازند، پذیرفته می‌شوند.

$S \to \lambda$ (اگر و فقط اگر $\lambda \in L$ باشد)

ب ـ گرامرهای مستقل از متن (Context Free): (نوع دو)

گرامرهای مستقل از متن به شکل مقابل تولید شده و ماشین‌های پشته‌ای (PDA) پذیرنده زبان این نوع از گرامرها هستند.

$S \to \lambda$ (اگر و فقط اگر $\lambda \in L$)

$A \to u \ ; A \in V, u \in (\sum \cup V)^+$

ج ـ گرامرهای حساس به متن (Context Sensitive): (نوع یک)

$G = (\Sigma, V, R, S).$

قوانین گرامر در این زبان چنین تعریف می‌شود: $R = \{u \to v \mid u \in (V \cup \Sigma)^+, v \in (V \cup \Sigma)^+, Length(u) \le length(v)\}$

زبان‌های وابسته به متن توسط ماشین کراندار خطی پذیرفته می‌شوند. این گرامرها فاقد قانون λ هستند.

د ـ گرامرهای بدون محدودیت (Unrestricted) : (نوع صفر)

گرامر زبان‌های بدون محدودیت این‌گونه تعریف می‌شوند: $G = (\Sigma, V, R, S).$

$R = \{u \to v \mid u \in (V \cup \Sigma)^+, v \in (V \cup \Sigma)^*\}$

سمت راست این گرامر، برخلاف گرامر وابسته (حساس) به متن، می‌تواند λ هم باشد. دقت کنید که هر زبانی یک زبان بدون محدودیت است. زبان گرامرهای بدون محدودیت توسط ماشین تورینگ ساخته می‌شود.

نکته ۴: رابطه‌های زیرمجموعه‌ای زیر بین گروه‌های مختلف از زبان‌ها برقرار هستند:

زبان منظم $\subset$ زبان مستقل از متن $\subset$ زبان حساس به متن $\subset$ زبان بدون محدودیت

مثال ۱۴: گرامر مقابل با حذف کدام قانون، وابسته به متن می‌شود؟

$S \to a \mid bSBC$
$T \to b\ TB \mid C$
$TCB \to bT$
$CT \to CB$
$B \to bB \mid C$

۱) $T \to CB$

۲) $CT \to CB$

۳) $TCB \to bT$

۴) هیچ‌کدام

پاسخ: گزینه «۳» در گرامر وابسته به متن، هر قانون به‌شکل $u \to v$ است که باید طول u کمتر یا مساوی طول v باشد.

مثال ۱۵: برای زبان‌های L_1 روی $\sum_1=\{a\}$ و L_2 روی $\sum_2=\{b\}$، دو تبدیل زبانی روی $\sum=\{0\}$ به شکل زیر تعریف می‌گردد:

$$L_1 \oplus L_2 = \{0^{m+n} \mid a^m \in L_1 \ \ AND \ \ b^n \in L_2\}$$

$$L_1 \ominus L_2 = \{0^{m-n} \mid a^m \in L_1 \ \ AND \ \ b^n \in L_2\}$$

الف) اگر $A=\{aa\}$ و $B=\{bb\}$، $A\oplus B$ و $A\ominus B$ را مشخص کنید.

ب) اگر A و B دو زبان متناهی باشد، کاردینالیتی $A\oplus B$ و $A\ominus B$ را تعیین کنید.

☑ **پاسخ:** الف ـ $A\ominus B=\{\lambda\}$ و $A\oplus B=\{0000\}$

ب ـــ فرض کنید $A=\{a^{m1},...,a^{mk}\}$ و $B=\{b^{n1},...,b^{np}\}$ باشد به طوریکه $m_i,n_j\in\mathbb{N}$ $1\le i\le k$ و $1\le j\le p$، در این صورت $|A\oplus B|\le kp$ و $|A\ominus B| \le kp$ است و این زبان‌ها نیز متناهی هستند. به طور دقیق داریم:

$|A\oplus B|\le \max(m_i)+\max(n_j)$, $|A\ominus B|\le \max(m_i)-\min(n_j)$ است.

مثال ۱۶: به کمک الفبای $\sum=\{0\}$، دو زبان زیر را روی الفبای {a, b} تعریف می‌کنیم:

$$SP(L)=\{a^m b^n \mid 0^{m+n}\in L\}$$

$$SM(L)=\{a^m b^n \mid 0^{m-n}\in L\}$$

اگر $L_1=\{00,0000\}$ باشد حاصل $SP(L_1)$ و $SM(L_1)$ را به‌دست آورید؟

☑ **پاسخ:** روشی است که عناصر L_1 شامل 0^4 و 0^2 هستند، لذا ساخت معادله $m+n=2$ و یا $m+n=4$ می‌تواند صرفاً به 3 و 5 حالت به ترتیب انجام شود که با حذف پاسخ‌های مشابه داریم:

$$SP(L_1)=\{ab, aa, bb, aaaa, aaab, aabb, abbb, bbbb\}$$

برای $SM(L_1)$ پاسخ‌ها شامل تعداد حالت‌های بسیار بیشتری هستند که در اینجا شکل کلی رشته‌ها را ارائه می‌نماییم:

$$SM(L_1)=\{a^i b^j \mid (i-j=2)\ OR (i-j=4)\}$$

مثال ۱۷: اگر A مجموعه پیشوند و B مجموعه پسوند و C مجموعه زیر رشته‌های یک رشته مفروض باشند، کدام رابطه زیر نادرست است؟

۱) $A\subseteq C$　　۲) $A\cup B\subseteq C$

۳) دربرخی رشته‌ها A, B, C مساویند　　۴) هیچ‌کدام

☑ **پاسخ:** گزینه «۴» اجتماع مجموعه پیشوندها و پسوندها لزوماً مجموعه زیر رشته‌ها نیست. چراکه در برخی رشته‌ها مثل abc ، زیر رشته 'b' نه پسوند است و نه پیشوند. با این حال همه پیشوندها و همه پسوندها خودشان جزو زیررشته‌ها هستند. لذا گزینه (۲) درست است. درباره گزینه ۳، اگر $w = aaa$ فرض شود، $A=B=C$ خواهد بود. در ضمن همواره داریم:

$$\begin{cases} B\subseteq C \\ A\subseteq C \\ A\cup B\subseteq C \end{cases}$$

مثال ۱۸: مجموعه پیشوند زبان $L=\{ab, baa\}$ که $pre(L)$ می‌نامیم را مشخص کنید. همچنین $L\setminus Pre(L)$ و $L/Pre(L)$ و $Pre(L)\setminus L$ و $Pre(L)/L$.

☑ **پاسخ:** خیلی راحت برای هر رشته از ابتدا شروع و پیشوندهای با طول $0,1,...$ را خارج کنید. اکنون برای L و $pre(L)$ موارد خواسته شده را به شکل زیر خواهیم داشت:

$$pre(L)=\{\lambda, a, ab, b, ba, baa\}$$
$$L\setminus Pre(L)=\phi$$
$$L/Pre(L)=\{ab, baa, a, \lambda, ba\}$$
$$pre(L)/L=\{\lambda\}$$
$$pre(L)\setminus L=\{\lambda, a, b, ba\}$$

مثال ۱۹: فرض کنید ϕ مجموعه تهی و λ مجموعه رشته تهی است. کدام گزینه صحیح می‌باشد؟

۱) $\phi^*-\phi^*=\phi$　　۲) $\lambda^*\phi^*=\phi$　　۳) $\lambda^*-\phi=\phi$　　۴) $\phi^*=\phi$

☑ **پاسخ:** گزینه «۱» زیرا $\{\lambda\}=\phi^*=\lambda^*$. از طرفی $\phi^*=\{\lambda\}$ و $\phi^*\neq\phi$ است ، لذا گزینه ۴ غلط می‌باشد . همین طور $\lambda^*\phi^*=\{\lambda\}$ است، لذا گزینه ۲ نیز نادرست است.

مدرسان شریف

فصل سوم

«ماشین تورینگ و زبان‌های حساس به متن»

مقدمه

در دو فصل قبل مفاهیم اولیه اتوماتا همچون زبان‌های منظم (باقاعده) و ارتباط آن با ماشین‌های متناهی (DFA , NFA) و زبان‌های مستقل از متن و ارتباط‌شان با پذیرنده‌های پشته‌ای (PDA) مورد بحث قرار گرفت. همان‌طور که بیان شد زبان‌های منظم، زیر مجموعه‌ای از زبان‌های مستقل از متن می‌باشند؛ لذا قدرت اتوماتای پشته‌ای (PDA) از توانایی‌های ماشین متناهی (DFA) بیشتر است. با این وجود، هر چند زبان‌های مستقل از متن مبنای زبان‌های برنامه‌سازی هستند ولی محدودیت‌هایی دارند.

همان‌طور که می‌دانید، ماشین متناهی معین (DFA)، رشته ورودی خود را از چپ به راست خوانده و بر اساس وضعیت فعلی و ورودی خوانده شده، وضعیت بعدی را کنترل و مشخص می‌کند. ماشین پشته‌ای (PDA) دارای یک حافظه‌ی پشته‌ای اضافه‌ای است و ورودی را از چپ به راست می‌خواند. وضعیت بعدی ماشین بر حسب حالت فعلی، ورودی خوانده شده و آنچه از پشته خارج می‌شود مشخص می‌گردد. کنترل خواندن یا نوشتن در پشته با دستورهای Push و Pop انجام می‌شود. با مرور ویژگی‌های اصلی یک DFA (که در فصل اول بیان شد)، مشخص می‌شود این نوع پذیرنده نمی‌تواند اطلاعات و حالت‌های خود را به خاطر بیاورد؛ برعکس DFA، یک PDA با استفاده از حافظه‌ی پشته‌ای می‌تواند بین حالت‌های مختلف وضعیت خود تمایز قایل شود.

دو ماشین DFA و PDA نمی‌توانند زبان‌های معرفی شده توسط یک گرامر حساس به متن یا بدون محدودیت را تولید کنند. از این رو ما به توسعه‌ی ماشین‌ها و گرامرها و نیز معرفی ماشین‌های قدرتمند نیازمند هستیم. یک مثال ابتدایی برای این نوع زبان‌ها $L = \{a^n b^n c^n : n \geq 0\}$ روی الفبای $\Sigma = \{a,b,c\}$ است. هر انتخاب دیگری به شکل $L_{new} = \{a^n b^{f(n)} c^{g(n)} : n \geq 0\}$ که f و g توابعی چندجمله‌ای غیر ثابت باشد، نیز این‌گونه خواهد بود. زمینه ساخت این زبان‌ها در شکل‌های محاسباتی پیشرفته مانند صف‌های اولویت ایجاد می‌گردد، از همین‌رو یک ماشین پشته‌ای نمی‌تواند آن‌ها را تشخیص دهد.

توسعه‌ی بعدی نسل ماشین‌های پذیرنده، در جهت امکان تمایز و تشخیص حالت‌های بیشتر و پیچیده‌تری است؛ به‌طوری که توانایی پذیرنده در پردازش نوار نیز شکل‌های مختلفی پیدا کند. امکان خواندن، نوشتن، حرکت به راست، حرکت به چپ و ثابت ماندن نوارخوان در کنار تک نواره یا چند نواره بودن، حالت‌های متنوع مورد نظر برای پذیرنده‌های پیشرفته جدید خواهد بود.

توسعه ماشین‌ها از پشته‌ای به نوع قوی‌تر، موضوع این فصل است. در همین رابطه پذیرنده‌ی یک زبان حساس به متن یا بدون محدودیت به دقت مورد بررسی قرار می‌گیرد. پاسخی که در ادامه برای توسعه ماشین‌های پذیرنده برای زبان‌های حساس به متن و بدون محدودیت ارائه خواهد شد، صرفاً یک کلاس ساده از پذیرنده‌ها با یک قدرت محدود و مشخص نخواهد بود. چرا که این پاسخ براساس رفتار عمومی مسائل محاسباتی و با روش دقیق ریاضی انجام می‌شود و کلاسی ماکزیمال از زبان‌های محاسبه‌پذیر را در بر خواهد داشت.

طبق دسته‌بندی چامسکی، گرامرهای حساس به متن (Context Sensitive) به گرامرهای نوع یک، موسوم هستند، این گرامرها فاقد قانون λ بوده و به‌صورت زیر تعریف می‌شوند:

$$\begin{cases} G = (\sum, V, R, S). \\ R = \left\{ u \to v \,\middle|\, u \in (V \cup E)^+ \,,\, v \in (V \cup E)^+ \,, \text{Length}(u) \leq \text{Length}(v) \right\} \end{cases}$$

گرامرهای بدون محدودیت یا گرامرهای نوع صفر نیز به‌فرم مقابل تعریف می‌شوند:

$$\begin{cases} G = (\sum, V, R, S). \\ R = \{u \to v \mid u \in (V \cup E)^+ \,,\, v \in (V \cup E)^*\} \end{cases}$$

گرامرهای حساس به متن را گرامرهای یکنواخت یا غیر انقباضی (Non Contracting) می‌نامند. این گرامرها زیر مجموعه گرامرهای بدون محدودیت هستند. همانند انواع دیگر زبان‌ها، زبان‌های حساس به متن و زبان‌های بدون محدودیت توسط ماشین تورینگ که یک مفهوم کلی از ماشین‌های محاسباتی است،

پذیرفته می‌شوند. برخلاف گرامر حساس (وابسته) به متن، در سمت راست یک گرامر بدون محدودیت می‌تواند λ قرار داشته باشد. با این حال بدون در نظر گرفتن رشته λ رابطه زیر میان کلاس‌های زبان‌های رشته‌ای برقرار است.

R_L : مجموعه کلیه زبان‌های منظم (باقاعده) باشد؛　　CF_L : مجموعه زبان‌های مستقل از متن باشد؛

CS_L : مجموعه زبان‌های وابسته به متن باشد؛　　T_L : مجموعه زبان‌های بدون محدودیت باشد؛

همواره خواهیم داشت: $R_L \subseteq CF_L \subseteq CS_L \subseteq T_L$

یعنی «گرامر نوع صفر $\supseteq$ گرامر نوع اول $\supseteq$ گرامر نوع دوم $\supseteq$ گرامر نوع سوم» است. به عبارتی هر زبان نوع i یک زبان نوع $i-1$ خواهد بود. همه زبان‌های بدون محدودیت توسط ماشین تورینگ پذیرفته می‌شوند. همچنین ماشین تورینگ قادر به تولید زبان‌های تعریف شده توسط یک گرامر حساس به متن است.

مثال ۱: گرامری بنویسید که زبان $L = \{a^n b^n c^n \mid n > 0\}$ را تولیدکند؟

☑ **پاسخ:** یکی از پاسخ‌ها برای این سوال به شکل مقابل است.

$$G_1: \begin{cases} S \to aSbc \mid abc. \\ cb \to bc. \end{cases}$$

پاسخ دیگر می‌تواند این‌گونه ارائه شود:

$$G_2: \begin{cases} S \to aB \\ B \to Cc \\ C \to Sb \\ cb \to bc \end{cases}$$

ماشین تورینگ

مفهوم ماشین تورینگ از آنجا شکل گرفت که ماشین‌های متناهی از مخزن (انباره) موقتی برای ذخیره اطلاعات برخوردار نبودند. در اتوماتای پشته‌ای برای نگهداری نمادها از یک یا چند پشته استفاده می‌شد. مجهز کردن اتوماتا با محل ذخیره‌سازی انعطاف‌پذیرتر برای پذیرش رشته‌های یک خانواده جدید از زبان‌ها ما را به مفهوم تورینگ و زبان‌های حساس به متن خواهد رساند.

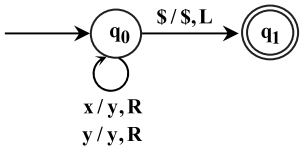
می‌توان ماشین تورینگ را کامپیوتری ساده در نظر گرفت که واحد پردازشی آن حافظه محدودی دارد و محل حافظه جانبی ذخیره نشان‌ها (نوار) ظرفیتی بدون محدودیت دارد. دستورالعمل‌های این کامپیوتر محدود است؛ و ماشین با تشخیص یک نشانه روی نوار تصمیم می‌گیرد در مرحله بعد، چه کاری انجام دهد. تابع گذر δ در عملکرد این کامپیوتر نقش مهمی دارد.

ماشین تورینگ از حالت اولیه (q_0) و اطلاعات موجود روی نوار شروع می‌شود. سپس گام‌های کنترل شده توسط تابع δ را دنبال می‌کند. ممکن است در طول این فرآیند، محتوای هر سلول از نوار تغییر کرده یا چند بار خوانده شود؛ در پایان، تورینگ در حالات نهایی متوقف می‌شود.

می‌توان برای نمایش ماشین تورینگ از یک گراف انتقال استفاده کرد. یال‌های گراف با سه مؤلفه به صورت مثالی $a/m,R$ نشان داده می‌شود؛ در این‌جا a نشانه فعلی روی نوار، m نشانه‌ی جایگزین a و R بیانگر جهت حرکت هد خواندن / نوشتن نوار به سمت راست (Right) است.

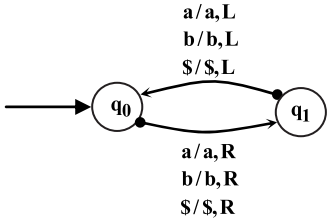
برای مثال، ماشین تورینگ با مشخصات زیر مفروض است:

$$Q = \{q_0, q_1\},\ \sum = \{x, y\},\ \Gamma = \{x, y, \$\},\ F = \{q_1\} \begin{cases} \delta(q_0, x) = (q_0, y, R) \\ \delta(q_0, y) = (q_0, y, R) \\ \delta(q_0, \$) = (q_1, \$, L) \end{cases}$$



گراف بیانگر تورینگ داده شده در ذیل آمده است، مثلاً در این گراف، اگر حرف x زیر هد قرار گرفته باشد، طبق قانون $\delta(q_0, x) = (q_0, y, R)$، مقدار y جایگزین x شده و هد یک خانه به راست حرکت می‌کند.

در نمونه بعدی، ماشین تورینگی نمایش داده شده که هیچ‌گاه متوقف نمی‌شود و مانند حلقه بی‌نهایت در برنامه‌نویسی است. اگر ورودی نوار به صورت $(ab)^n$ که $n > 0$ باشد و هد روی حرف a قرار گرفته باشد: طبق ماشین زیر، در حالت q_0 با دیدن a، به حالت q_1 رفته و هد به راست حرکت می‌کند. اکنون نوک هد روی b واقع شده است و این نشانه نیز خوانده شده و مطابق دستور $b/b,L$، حرف b تغییر نکرده و هد یک خانه به چپ حرکت می‌کند. در اینجا باز هم نوک هد a را دیده و به حالت q_1 رفته و a نیز تغییر نمی‌کند. این دنباله از حرکت‌ها بارها اجرا می‌شود.



شکل ۱- ماشین تورینگ بدون توقف

توضیح: در هر ماشین تورینگ موارد زیر برقرار است:

الف ـ ماشین تورینگ قطعی است؛ یعنی تابع گذر δ حداکثر یک حرکت را برحسب مقدارش (به چپ یا راست) تعریف می‌کند.

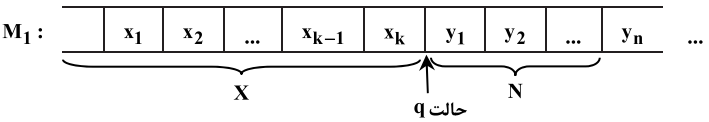
ب ـ ماشین تورینگ حاوی نواری نامحدود از طرفین بوده و امکان حرکت به چپ یا راست را به صورت نامحدود ممکن می‌کند.

ج ـ در ابتدای حرکت هد، نوار محتوای مشخصی دارد که می‌توان برخی قسمت‌های آن را به عنوان ورودی فرض نمود. هرگاه ماشین متوقف شود، بخشی از محتوای نوار یا همه‌ی آن می‌تواند به عنوان خروجی در نظر گرفته شود؛ لذا این ماشین فایل ورودی یا دستگاه خروجی خاصی ندارد.

د- بسیاری از استدلال‌های این فصل تنها برای زبان‌هایی صادق است که تهی نیستند؛ چرا که یک ماشین تورینگ رشته‌های تهی را نمی‌پذیرد.

❖ **تعریف ۱:** توصیف لحظه‌ای از ماشین تورینگ: شامل اطلاعات رشته‌هایی است که در چپ و راست هد خواندن ـ نوشتن قرار دارد.

برای مثال توصیف لحظه‌ای $x_1 x_2 \dots x_k \; q \; y_1 y_2 \dots y_n$ یا XqN برای ماشین M_1 شامل این اطلاعات است.



در این نمای کلی، نوک هد نوار روی خانه‌ای از نوار است که محتوایش بلافاصله پس از q قرار می‌گیرد.

❖ **تعریف ۲:** گذر از یک پیکربندی به پیکربندی دیگر، با استفاده از تابع δ در تورینگ با علامت $\vdash_M$ نشان داده می‌شود. برای مثال به ازای تابع $\delta(q_1, a) = (q_2, n, R)$، اگر حالت فعلی q_1 بوده و هد روی نماد a از رشته ورودی نوار، یعنی $bac\$$، باشد، با دیدن a، به جای آن باید n در رشته فوق قرار گرفته و هد یک خانه به راست حرکت کند، این موضوع را به صورت $b\,q_1\,a\,c\,\$ \vdash_M b\,n\,q_2\,c\,\$$ نشان می‌دهیم.

❖ **تعریف ۳:** در ماشین تورینگ M، به ازای $\forall a_i \in T$ و $q_1 \in Q$، رشته‌ی $a_1 a_2 \cdots a_{k-1} q_1 a_k a_{k+1} \cdots a_n$ یک توصیف لحظه‌ای از M است.

حرکت $a_1 a_2 \cdots a_{k-1} q_1 a_k \cdots a_n \vdash a_1 \cdots a_{k-1} b q_2 a_{k+1} \cdots a_n$ در شرایطی امکان‌پذیر است که داشته باشیم:

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

همین‌طور حرکت $a_1 a_2 \cdots a_{k-1} q_1 a_k \cdots a_n \vdash a_1 a_2 \cdots q_2 a_{k-1} b a_{k+1} \cdots a_n$ تنها زمانی امکان‌پذیر است که داشته باشیم:

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

تورینگ M با شروع از پیکربندی $x_1 q_i x_2$ توقف‌پذیر است؛ در شرایطی که داشته باشیم: $x_1 q_i x_2 \vdash^* y_1 q_j a y_2$ به‌طوری که برای q_j و a هیچ قانونی مثل $\delta(q_j, a)$ تعریف نشده باشد. به توالی (Sequence) از پیکربندی‌ها که به حالت توقف منتهی می‌شود، محاسبه یا Computation می‌گویند.

مثال ۲: مفهوم عبارت $\infty \vdash^* aqb$ چیست؟

☑ **پاسخ:** منظور از عبارت مورد نظر این است که اگر تورینگ با پیکربندی اولیه aqb شروع به کار کند، در حلقه نامتناهی خواهد افتاد؛ این تعریف برای معرفی ماشین تورینگ توقف‌ناپذیر به‌کار می رود.

بنابر تز چرچ، هر مسأله قابل حل می‌تواند توسط ماشین تورینگ پذیرفته شود. از این رو ماشین تورینگ، پاسخ کاملی برای ساخت زبان محاسبه‌پذیر است. در ادامه ماشین تورینگ استاندارد را تعریف و انواع مختلف آن را بررسی می‌کنیم.

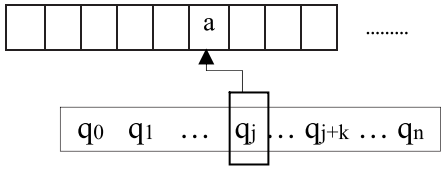
❖ **تعریف ۴: ماشین تورینگ استاندارد**

ماشین تورینگ استاندارد عبارت است از پنج‌تایی $M = (Q, \Sigma, P, q_0, \delta)$ که:

Q یک مجموعه ناتهی متناهی از وضعیت‌ها است. Σ الفبای زبان است؛ و P الفبای نوار، که همواره نماد # را شامل می‌شود و $\Sigma \subseteq P - \{\#\}$ است.

q_0 وضعیت شروع ماشین است. تابع جزئی δ که محاسبات ماشین را کنترل می‌کند، به این صورت است: $\delta : Q \times P \to Q \times P \times \{L, R\}$

شکل عمومی یک ماشین تورینگ این‌گونه است:



بنابر شکل صفحه قبل ماشین در وضعیت q_j بوده و سر نوار خوان روی حرف a قرار دارد.

ماشین‌های تورینگ براساس این‌که پذیرنده (یا تشخیص‌پذیر) زبان باشند یا تصمیم‌پذیر زبان، به دو نوع مختلف تقسیم خواهند شد. یک ماشین‌تورینگ پذیرنده، عبارت است از شش‌تایی $M = (Q, \Sigma, P, q_0, \delta, F)$ که با اضافه کردن پارامتر جدید F به عنوان وضعیت‌های نهایی، به تعریف تورینگ استاندارد ساخته می‌شود. ماشین $M(L)$ برای هر رشته ورودی w، که عضو زبان L باشد، پس از پردازش رشته به یک وضعیت نهایی در F می‌رود.

اگر $F = \{q_{accept}, q_{reject}\}$ باشد، یک ماشین تورینگ تصمیم‌پذیر است. در ماشین‌های تصمیم‌پذیر وضعیت نهایی یا q_{accept} یا q_{reject} است بدین ترتیب اگر رشته $w \in L$ باشد و $M(L)$ یک ماشین تورینگ تصمیم‌پذیر زبان L، باشد ماشین روی w به وضعیت نهایی q_{accept} می‌رود؛ اگر w عضو L نباشد، ماشین به q_{reject} می‌رود. به ماشین تورینگی که نوار آن از دو طرف محدود شده باشد، ماشین کراندار خطی (LBA) می‌گویند.

توضیح: ویژگی‌های ماشین تورینگ

مهم‌ترین ویژگی ماشین تورینگ عبارت است از:

۱ـ تعداد حرکات ماشین تورینگ نامحدود است و نوار ورودی تعداد نامتناهی خانه دارد.

۲ـ ماشین تورینگ، معین است، یعنی به ازای هر پیکربندی حداکثر (و نه دقیقاً) یک حرکت تعریف می‌شود.

۳ـ در تورینگ استاندارد، ورودی ماشین بخشی از نوار است و پس از توقف ماشین، بخشی از نوار خروجی در نظر گرفته می‌شود.

۴ـ واحد کنترل ماشین بر اساس حالت جاری و ورودی خوانده شده‌ی فعلی، نوع حرکت ماشین را مشخص کرده و وضعیت بعدی را می‌سازد.

۵ـ سر نوارخوان ماشین، قابلیت حرکت به چپ و راست دارد.

۶ـ موقعیت‌های خالی حافظه با # یا □ و B، برچسب خورده می‌شود.

توضیح: زبان‌های تصمیم‌پذیر

اگر برای زبان، یک تورینگ تصمیم‌پذیر وجود داشته باشد آن را تصمیم‌پذیر می‌نامند. کلاس زبان‌های تصمیم‌پذیر از الگوریتم عضویت برخوردار هستند. یعنی برای هر $w \in \Sigma^*$ که عضو زبان L باشد ماشین در وضعیت q_{accept} و برای هر w دیگر در وضعیت q_{reject} متوقف شود. مجموعه زبان‌های تصمیم‌پذیر را زبان‌های بازگشتی نامیده و با RE_L نمایش می‌دهیم. از آنجا که کلاس زبان‌های تصمیم‌پذیر شامل تمام زبان‌ها نیست، ماشین‌های تورینگ عمومی‌تر از ماشین‌های تورینگ تصمیم‌پذیرند. ماشین تورینگ محدودیتی در تولید زبان نداشته و هر زبان محاسبه‌پذیری را تولید می‌کند.

ماشین تورینگ با ماشین‌های قبلی متفاوت است. نخست این‌که ماشین تورینگ هم خواندن و هم نوشتن را روی نوار انجام می‌دهد؛ نوارخوان هم به چپ و هم به راست حرکت می‌کند؛ و از همه مهم‌تر طول نوار و تعداد حرکات آن، نامتناهی در نظر گرفته می‌شود. تفاوت ماشین تورینگ با PDA در این است که اولا تورینگ توانایی نوشتن و خواندن روی نوار را دارد ولی PDA فقط روی پشته می‌تواند Push یا Pop انجام دهد، ثانیاً نوارخوان می‌تواند در دو جهت حرکت کند و سوم اینکه در PDA فقط طول پشته نامتناهی است ولی در تورینگ علاوه بر آن طول نوار و تعداد حرکات ماشین نیز نامتناهی است.

توضیح: ساختار ماشین تورینگ

الف) علامت خانه‌های بی‌ارزش نوار یکی از نشانه‌های #، □ یا B است. به عبارت دیگر هر خانه شامل این علامت‌ها یک محل خالی را مشخص می‌کند. برای مثال ممکن است سمت چپ نوار با قسمت‌های خالی یا # پر شود.

ب) تورینگ یک ماشین غیرقطعی، است زیرا تابع δ، $\delta: Q \times P \to Q \times P \times \{L, R\}$، یک تابع جزئی است و ممکن است برای خیلی حالت‌ها تعریف نشده باشد که این باعث توقف ماشین می‌شود. ممکن است یک تورینگ پذیرنده در وضعیت غیرپایانی متوقف شود. در این حالت، رشته ورودی برای زبان مورد نظر پذیرفته نخواهد شد. توقف در حالت تعریف نشده با توقف در وضعیت نهایی فرق می‌کند. شرط پذیرش رشته ورودی، توقف ماشین در حالت نهایی است.

ج) با اجرای دستور $\delta(q_i, a) = [q_j, b, R]$، هد با حرکت روی نوار، با مشاهده a از حالت q_i به حالت q_j رفته و b را در محل a روی نوار می‌نویسد، روی نوار، یک واحد به سمت راست حرکت می‌کند. دیاگرام حالت این دستور چنین است:

اگر بخشی از تورینگ، به‌صورت $(q) \xrightarrow{a/b,L} (s)$ باشد و نوار ورودی به‌صورت | a | b | a | a | # | باشد؛ پس از گذار به وضعیت s، نوار به صورت | # | a | b | a | a | درمی‌آید؛ لذا در هر حرکت سه عمل انجام می‌شود: ۱ـ تغییر حالت ماشین، ۲ـ نوشتن یک نشانه روی خانه جاری نوارخوان و ۳ـ حرکت نوارخوان به سمت چپ یا راست.

د) تورینگ پذیرنده یکی از انواع ماشین‌های تورینگ است که زبان‌های بدون محدودیت را می‌پذیرد و قادر است از وضعیت ابتدایی سمت چپ به وضعیت نهایی سمت راست برسد. درمدل این ماشین، F حالات نهایی را بیان می‌کند.

نکته ۱: اگر گرامر یک زبان شامل n قانون باشد، می‌توان برای پذیرش رشته‌های زبان از یک تورینگ $n+2$ نواره استفاده کرد که نماد شروع در صفر و جمله ورودی در نوار $n+1$، ذخیره می‌شود. در حالت پذیرش، نوار صفر و نوار $n+1$ محتوای یکسانی دارند.

نکته ۲: ماشین تورینگ بدون قابلیت تغییر محتوای نوار و با حرکت به راست، تنها زبان‌های منظم (R_L) را می‌پذیرد. اگر نوشتن و خواندن نوار فقط با دستورهای Push و Pop انجام شود، ماشین تورینگ زبان‌های مستقل از متن (CF_L) را تولید خواهد کرد. اگر نوار از دو طرف محدود شود، ماشین تورینگ فقط زبان‌های حساس به متن (CS_L) را خواهد پذیرفت. اگر تورینگ برای هر ورودی توقف کند، ماشین، زبان‌های بازگشتی را (RE_L) خواهد ساخت. تورینگ استاندارد زبان‌های بازگشتی برشمردنی (T_L) را می‌سازد. البته هنوز زبان‌هایی وجود دارند که هیچ ماشین تورینگی ندارند.

نکته ۳: q_0 نمی‌تواند، در ماشین تورینگ حالت نهایی باشد.

نکته ۴: برای هر ماشین تورینگ، تورینگ معادل استاندارد آن با کمتر یا مساوی ۶ حالت وجود دارد.

مثال ۳: اگر $M=(Q,\Sigma,\mu,q_0,\delta,F)$ ماشین تورینگ پذیرنده با توصیف روبه‌رو باشد $Q=\{q_0,q_1\},\Sigma=\{a,b\},\mu=\{a,b,\#\},\ F=\{q_1\}$ و تابع δ به صورت زیر تعریف شود؛ توصیف بلافصل این ماشین چگونه است؟

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

توصیف بلافصل این ماشین تورینگ از وضعیت محاسبات به شکل زیر خواهد بود.

$$a_1\dots a_{k-1}q_0a_k\dots a_n \vdash a_1\dots a_{k-1}bq_0\,a_{k+1}\dots a_n \vdash {}^*a_1\dots a_{k-1}bb\dots bq_0$$

در حرکت بعد، ماشین با دیدن q_0 و $\#$ متوقف می‌شود. به بیان ریاضی، می‌توان مجموعه عبارت‌های پذیرفته شده توسط ماشین تورینگ M را به صورت زیر بیان کرد:

$$L(M)=\{w\in\Sigma^*;q_0w \vdash {}^*X_1q_fX_2,X_1X_2\in\mu^*,q_f\in f\}$$

پس عبارت‌هایی که ماشین تورینگ M برای آن‌ها متوقف نمی‌شود، عضو زبان $L(M)$ نیست.

مثال ۴: برای زبان $L=\{a^mb^*,\ m\le 3\}$ یک ماشین تورینگ طراحی کنید.

فرض کنید $\Sigma=\{a,b\}$ و $Q=\{q_0,q_1,q_2,q_3\}$ که $F=\{q_3\}$ است. حال برای δ داریم:

$$\delta(q_0,a)=(q_1,\ a,R) \qquad \delta(q_0,a)=(q_2,b,R)$$
$$\delta(q_0,b)=(q_2,b,R) \qquad \delta(q_1,a)=(q_2,a,R)$$
$$\delta(q_1,b)=(q_2,b,R) \qquad \delta(q_2,a)=\delta(q_2,b,R)$$
$$\delta(q_2,b)=(q_2,b,R) \qquad \delta(q_2,\#)=(q_3,\#,L)$$

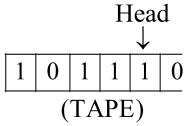
این ماشین تورینگ می‌تواند هر عبارت به شکل ab^*، a^2b^*، a^3b^* و یا b^* را تولید کند.

❖ تعریف ۵: پیکربندی تورینگ

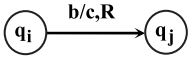
ماشین تورینگ $M=(Q,\Sigma,P,\delta,q_0,\{q_{Accept},q_{Reject}\})$ دارای پیکربندی (Configuration) به‌فرم رشته uqv است که $u,v\in P^*$ و $q\in Q$ به طوری که :

الف) M در حالت q است. ب) نوار شامل رشته‌ی uv می‌باشد. ج) نوک نوارخوان روی حفره‌ای که شامل اولین عنصر در v است.

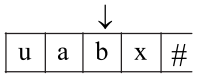
مثلاً $1011q_210$ به این معنا است که q_2 حالت جاری بوده و نوار به شکل زیر است:



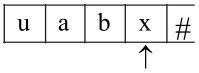
نتیجه‌ی پیکربندی C_1، پیکربندی C_2 است در شرایطی که ماشین تورینگ بتواند به‌طور قانونی با یک گام از C_1 به C_2 برود. برای مثال $\delta(q_i,b)=(q_j,c,R)$، پیکربندی uaq_ibx، پیکربندی $uacq_jx$ را حاصل می‌کند.



وضعیت نوار قبل از عملیات



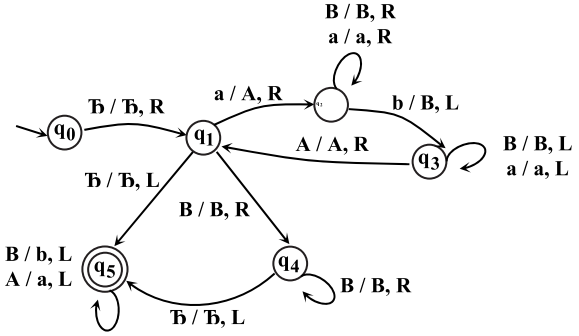
وضعیت نوار بعد از عملیات



پیکربندی شروع (Starting Configuration) روی رشته‌ی ورودی w به صورت q_0w و پیکربندی پذیرش(Accepting Configuration) به‌فرم xq_Ay است که q_A حالت پذیرش و $x,y\in P^*$ هستند. پیکربندی عدم‌پذیرش (Rejecting Configuration) به‌صورت xq_Ry است که q_R حالت عدم پذیرش و $x,y\in P^*$ می‌باشند. به پیکربندی‌های پذیرش و عدم پذیرش، پیکربندی‌های توقف (Halting) گفته می‌شود.

مثال ۵: ماشین تورینگی طراحی کنید که زبان $L=a^nb^n\ (n\ge 0)$ را بپذیرد.

☑ **پاسخ:** با خواندن هر a از چپ نوار ، یک A روی نوار نوشته و به راست می‌رویم تا به b برسیم. به‌جای b عنصر B روی نوار نوشته و به چپ برمی‌گردیم تا a بعدی خوانده شود. به این ترتیب همه‌ی b ها به B و a ها به A تبدیل می‌شوند. اگر w رشته‌ای به طول صفر باشد از q_1 به q_5 می‌رویم.



مثال ۶: (به‌طور گام به گام نشان دهید) اگر M ماشین تورینگ پذیرنده زبان $A = \{s\#s \mid s \in \{0,1\}^*\}$ باشد، رشته‌ی (01101#01101) را چگونه می‌پذیرد؟

☑ **پاسخ:** الف) هِد از چپ نوار خوانده و به‌جای عنصر خوانده شده، X را روی نوار ذخیره می‌کند.

↓										
X	1	1	0	1	#	1	1	0	1	#

ب) اگر عنصر بعدی فضای خالی (قبل از #) باشد، Reject یا عدم پذیرش رخ داده است.

ج) تا به # برسیم به راست حرکت می‌کنیم. اگر عنصر بعد از # با نماد جاری خوانده شده یکسان نبود Reject شده در غیر این‌صورت نماد جاری x جایگزین می‌شود.

						↓				
X	1	1	0	1	#	x	1	1	0	1

د) تا به x برسیم به چپ حرکت می‌کنیم. یک فضا به جلو رفته و آن را با X جایگزین می‌کنیم.

X	x	1	0	1	#	x	1	1	0	1		#

هـ) به جلو حرکت کرده، عملیات مراحل قبل زیگ زاگی ادامه می‌یابد تا به حالت زیر برسیم:

	↑									↓	
X	X	X	X	X	#	X	X	X	X	X	#

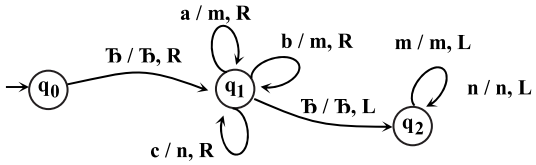
و) سرانجام، پس از اینکه آخرین نماد در چپ # به حرف X تغییر یافت، سایر نمادهای واقع در سمت راست # چک می‌شود، اگر در این مرحله به # (فضای خالی) رسیدیم حالت پذیرش و اگر به صفر یا یک برخورد کردیم، حالت عدم پذیرش رخ داده است.

چون حالت نهایی نوار به صورت زیر درمی‌آید، لذا رشته ورودی 11101#11101 توسط ماشین تورینگ پذیرفته شده است.

X	X	X	X	X	#	X	X	X	X	X↓
										#

مثال ۷: با توجه به $a \to m$ و $b \to m$ و $c \to n$ برای $L = \{(a \cup b \cup c)^*\}$ ماشین M را طوری طراحی کنید که $L' = \{(m \cup n)^*\}$ باشد.

☑ **پاسخ:** می‌دانیم:



$$P = \{a, b, c, m, n, t_0\} \cup \{Ƀ\}$$

چون در اینجا ترجمه یا تبدیل زبان مطرح است لذا حالت نهایی لزومی ندارد. با دیدن a و b و c به ترتیب m و m و n روی نوار جایگزین می‌کنیم. اگر در بین کار تبدیل، عنصری غیر از a و b و c دیده شود ماشین متوقف شده و خطا رخ می‌دهد؛ وقتی به t_0 یا انتهای رشته رسیدیم، به چپ (عقب) برگشته و عناصر تغییر یافته را تا ابتدای رشته بدون تغییر می‌گذاریم.

مثال ۸: برای زبان $A = \{0^{2^n} \mid n \geq 0\}$ ماشین تورینگ طراحی کنید.

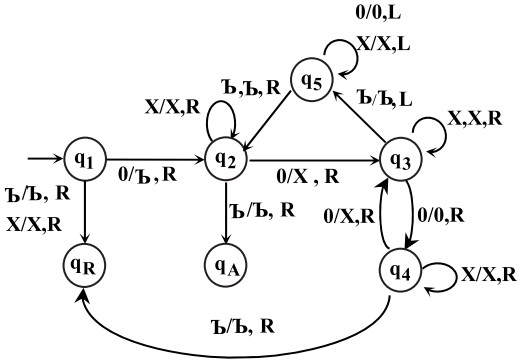
☑ **پاسخ:** در این TM، داریم:

$$Q = \{q_1, q_2, q_3, q_4, q_5, q_{Accept}, q_{Reject}\}$$

$$\Sigma = \{0\}, P = \{0, x, Ƀ\}$$

$$\delta : Q \times P \to Q \times P \times \{L, R\} \quad (\text{مثلاً } \delta(q_4, 0) = (q_3, x, R))$$

در شکل زیر q_R همان q_{Reject} (عدم پذیرش) و q_A همان q_{Accept} حالت پذیرش می‌باشد.



مثال ۹: پیکربندی لازم جهت پذیرش یا عدم پذیرش رشته **0000** روی ماشین تورینگ زبان $L = \{0^{2^n} \mid n \geq 0\}$ را مرحله به مرحله بنویسید.

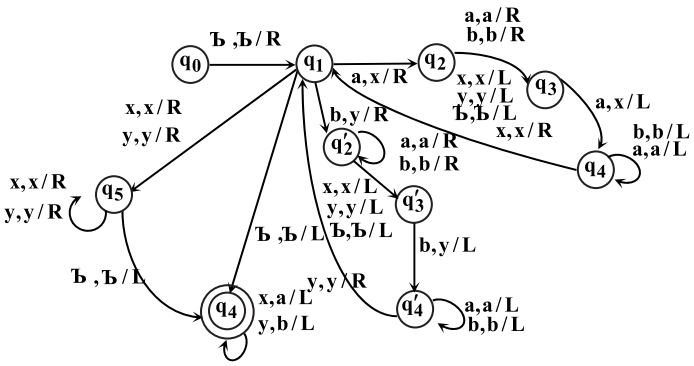
☑ **پاسخ:**

$$q_1 0000 \Rightarrow Ъq_2 000 \Rightarrow Ъxq_3 00 \Rightarrow Ъx0q_4 0 \Rightarrow Ъx0xq_3 Ъ \Rightarrow Ъx0q_5 x \Rightarrow$$
$$Ъxq_5 0x \Rightarrow Ъq_5 x0x \Rightarrow q_5 Ъx0x \Rightarrow Ъq_2 x0x \Rightarrow Ъxq_2 0x \Rightarrow Ъxxq_3 x$$
$$Ъxxxq_3 Ъ \Rightarrow Ъxxxq_5 x \Rightarrow Ъxxq_5 xx \Rightarrow Ъxq_5 xxx \Rightarrow q_5 Ъxxx \Rightarrow Ъq_2 xxx$$
$$Ъxq_2 xx \Rightarrow Ъxxq_2 x \Rightarrow Ъxxxq_2 Ъ \Rightarrow Ъxxx Ъq_{Accept} Ъ$$

مثال ۱۰: ماشین تورینگ مناسبی طراحی کنید که رشته‌های متعلق به زبان L را شناسایی کند. همچنین در ادامه تعلق رشته **#aba#** را به کمک آن چک کنید.

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

☑ **پاسخ:** برای رسم ماشین تورینگ M، از ابتدای رشته خوانده و هر عنصر مشابه را از انتهای رشته علامت می‌زنیم. اگر عناصر به درستی علامت خوردند، محاسبه موفقیت‌آمیز بوده است؛ در ضمن رشته به طول صفر نیز به شکل | # | # | قابل پذیرش است (گذر از q_1 به q_4).



برای تست تعلق رشته aba مراحل پردازش رشته را دنبال می‌کنیم.

$$Ъ\,aba\,Ъ \Rightarrow Ъa\,ba\,Ъ \Rightarrow Ъx\,b\,aЪ \Rightarrow Ъxb\,aЪ \Rightarrow Ъxba Ъ \Rightarrow$$
$$\uparrow q_0 \quad \uparrow q_1 \quad \uparrow q_2 \quad \uparrow q_2 \quad \uparrow q_2$$

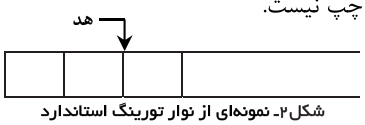
$$Ъxb\,aЪ \Rightarrow Ъx\,b\,xЪ \Rightarrow Ъx\,bxЪ \Rightarrow Ъx\,b\,bxЪ \Rightarrow Ъ\,xy\,xЪ \Rightarrow bx\,y\,xb \Rightarrow \text{توقف}$$
$$\uparrow q_3 \quad \uparrow q_4 \quad \uparrow q_4 \quad \uparrow q_1 \quad \uparrow q'_2 \quad \uparrow q'_3$$

چون q'_3 حالت نهایی نیست، لذا رشته $Ъ\,aba\,Ъ$ به زبان L تعلق ندارد.

انواع ماشین تورینگ

ماشین‌های تورینگ با نوار نیمه متناهی:

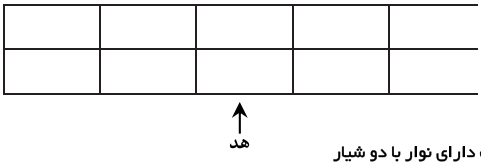
بسیاری از محققان، مدل کلی ارائه شده در ابتدای فصل را به عنوان تورینگ استاندارد در نظر نمی‌گیرند. این محققان، در نظری بالعکس، ماشینی را به عنوان ماشین تورینگ استاندارد می‌پذیرند که نوار آن از یک طرف نامحدود است، یعنی مدلی همانند شکل زیر که نوارش در چپ مسدود شده است و زمانی که هد به انتهای چپ نوار برسد مجاز به حرکت به چپ نیست.



شکل۲- نمونه‌ای از نوار تورینگ استاندارد

شبیه‌سازی ماشین‌های تورینگ استاندارد با ماشین تورینگ نوار نیمه متناهی:

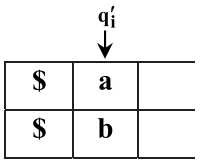
برای نشان دادن این‌که مدل های مختلف ماشین تورینگ، عملکرد یکسانی دارند نشان می دهیم هر کدام توسط دیگری شبیه‌سازی می‌شود. ابتدا مطابق شکل ۳ ماشین تورینگ شبیه‌ساز M_1 را دارای نواری با دو شیار فرض بگیرید. شیار بالایی، اطلاعات سمت راست یک نقطه مرجع در نوار M_2 را درج می‌کند، برای نمونه می‌توان، مکان قرار گرفتن R/W در ابتدای محاسبات را به عنوان نقطه‌ی مرجع فرض کرد. شیار پایینی حاوی محتوای سمت چپ نوار در جهت عکس است.



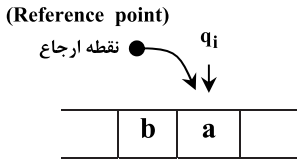
شکل ۳- تورینگ دارای نوار با دو شیار

ماشین M_1 طوری برنامه‌ریزی شده که وقتی هد ماشین M سمت راست نقطه‌ی مرجع قرار دارد از محتوای شیار بالا استفاده کرده و در غیر این صورت از محتوای شیار پایین استفاده کند. برای این منظور، باید مجموعه حالات M_1 را به دو مجموعه‌ی Q_V و Q_L تقسیم کرد. اولی برای استفاده از شیار بالایی و دیگری برای استفاده از شیار پایینی نوار است. انتهای چپ نوار توسط کاراکتر خاصی مثل $ مشخص شده و بدین ترتیب امکان حرکت از یک شیار به شیار دیگر فراهم خواهد شد.

برای مثال در شکل۴ـ (الف) و ۴ـ (ب) ماشینی که باید شبیه‌سازی شود و ماشین شبیه‌سازی شده نشان داده شده‌اند. قانون $\delta(q_i, a) = (q_j, c, L)$ یکی از قوانینی است که باید شبیه‌سازی شود.



شکل۴ـ (الف) ماشین شبیه‌سازی شده



شکل ۴ـ (ب) ماشین شبیه‌سازی کننده

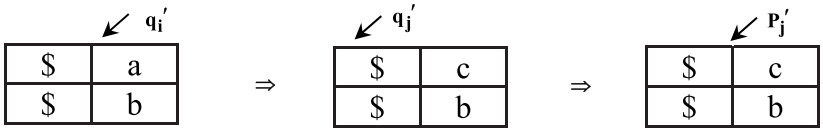
لازم است ماشین شبیه‌سازی شده ابتدا حرکت مقابل را اجرا نماید که در آن $q'_i \in Q_V$ است.

$$\delta'(q'_i, (a,b)) = (q'_j, (c,b), L)$$

از آنجا که $q'_i \in Q_V$ است، در این نقطه تنها اطلاعات شیار بالایی را در نظر می‌گیرد. حال، ماشین شبیه‌سازی شده در حالت $q'_j \in Q_V$، نشانه $(\$,\$)$ را دیده و از تابع گذر مقابل استفاده می‌کند:

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

این گذر، ماشین را به پیکربندی شکل ۵ می‌برد. اکنون ماشین در حالتی متعلق به Q_L قرار دارد و روی شیار پایینی کار می‌کند.

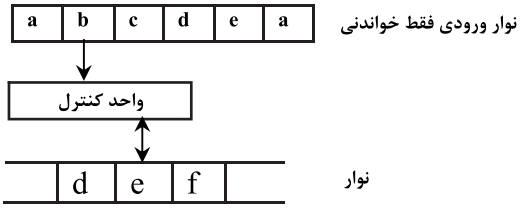


شکل ۵ یک توالی از پیکربندی‌ها در شبیه‌سازی $\delta(q_i, a) = (q_j, c, L)$

ماشین تورینگ off – line (برون خطی):

در تورینگ برون خطی حرکت توسط حالت درونی و کاراکتر فعلی خوانده شده از فایل ورودی و کاراکتر خوانده شده توسط هد مدیریت می‌گردد. نمای کلی این ماشین در شکل ۶ آمده است.

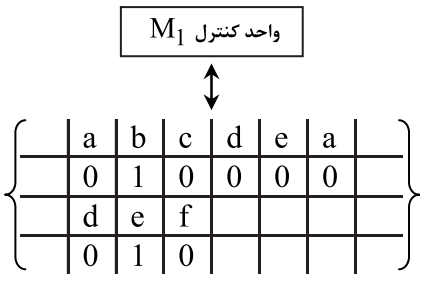
M :



شکل۶ـ شمای کلی ماشین تورینگ M برون خطی (off line)

ماشین M می‌تواند توسط ماشین تورینگ استاندارد M_1 شبیه‌سازی شود. M_1 می‌تواند محاسبه‌های ماشین تورینگ برون خطی (M) را توسط یک نواری ۴ شیاره شبیه‌سازی کند. اولین شیار این ماشین شامل ورودی است؛ شیار دوم مکان کاراکتر خوانده شده از ورودی است، شیار سوم نوار ماشین M است؛ شیار چهارم هم مکان قرار گرفتن هد در ماشین برون خطی M را نشان می‌دهد.

تورینگ استاندارد M_1 که عملیات ماشین برون خطی M شکل قبل را شبیه‌سازی می‌کند:



شکل۷ـ ماشین تورینگ استاندارد شبیه‌ساز M_1 برون خطی

شبیه‌سازی هر حرکت از تورینگ برون خطی M به چند حرکت از تورینگ استاندارد M_1 نیاز دارد. با شروع از یک موقعیت استاندارد (مثلاً انتهای سمت چپ) و اطلاعات مربوط که با یک علامت پایان خاص مشخص شده، تورینگ M_1 در شیار دوم موقعیت اطلاعات خوانده شده از فایل ورودی M را جستجو می‌کند. با قرار دادن واحد کنترل در حالتی که برای این منظور در نظر گرفته شده است، ارزشی که در خانه متناظر با آن در شیار اول وجود دارد، به خاطر سپرده می‌شود.

سپس شیار چهارم جستجو شده تا مکان قرار گرفتن هد M مشخص شود. با به خاطر آوردن ورودی و کاراکتری که در شیار سوم قرار دارد، نحوه‌ی کار M درک می‌شود. این اطلاعات، مجدداً به کمک حالت‌های درونی توسط M_1 به خاطر آورده می‌شود. در مرحله بعدی، تمام چهار شیار نوار M_1 تغییر می‌کند تا حرکت M منعکس شود؛ در نهایت هد خواندن ـ نوشتن ماشین M_1 به موقعیت استاندارد برمی‌گردد تا حرکت بعدی شبیه‌سازی شود.

مثال ۱۱: ماشین تورینگی را در نظر بگیرید که می‌تواند در هر حرکت، نشانه نوار یا مکان قرارگیری هد خواندن ـ نوشتن را تغییر دهد، ولی قادر به انجام هم‌زمان هر دوی این کارها نیست. مطلوب است:

الف ـ ارائه یک تعریف صوری (Formal) برای این ماشین.

ب ـ نشان دهید کلاس این ماشین‌ها با کلاس ماشین‌های تورینگ استاندارد، معادل (هم‌ارز) می‌باشد.

☑ **پاسخ:** (الف) تابع گذر این ماشین به صورت $\delta: Q\times\Gamma \to Q\times\Gamma\{L,R,S\}$ است که در آن به ازای هر تابع (R یا L) $\delta(q_i,a)=(q_j,b,L$ در صورتی که $a=b$ برابر خود تابع خواهد بود.

(ب) به ازای $a\neq b$ برای شبیه‌سازی $\delta(q_i,a)=(q_j,b,L)$ در تورینگ استاندارد، توابع جدید $\delta(q_i,a)=(q_{jL},b,S)$ و $\delta(q_{jL},b)=(q_j,b,L)$ را به ازای $\forall a\in\Gamma$ تعریف کرده و مانند قضیه اثبات شده در این فصل، ادامه می‌دهیم.

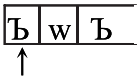
مثال ۱۲: ماشین تورینگی را در نظر بگیرید که روال متفاوتی دارد. به این شکل که انتقال در صورتی رخ می‌دهد که کاراکتر فعلی نوار از مجموعه‌ای معین نباشد. برای نمونه حرکت مقابل $\delta(q_i,\{a,b\})=(q_j,c,R)$

تنها در صورتی امکان‌پذیر است که کاراکتر فعلی نوار a یا b نباشد. به صورت صوری این موضوع را بیان کنید.

☑ **پاسخ:** به ازای هر $m\in\Gamma-\{a,b\}$، مقدار $\delta(q_i,m)=(q_j,c,R)$ به جای $\delta(q_i,\{a,b\})=(q_j,c,R)$ آورده می‌شود.

❖ تعریف ۶: ماشین تورینگ مبدل

مدل ریاضی این ماشین به شکل $M=(Q,\sum,P,\delta,q_0)$ است و برای پذیرش رشته w به شکل

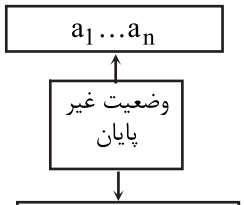


حالت نهایی ماشین به صورت

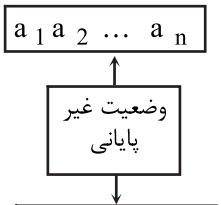
است یعنی اگر در خاتمه نوار خواندن مقابل خانه صفر قرار گیرد محاسبه موفق خواهد بود.

مثال ۱۳: برای پذیرش زبان مقابل یک ماشین تورینگ مبدل طراحی کنید. $L=\{(x,x^R)\mid x\in\{a,b\}^* \text{ and } x=x^R\}$

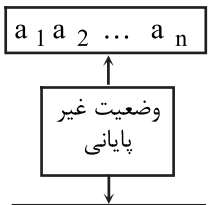
بعد از خواندن کل ورودی



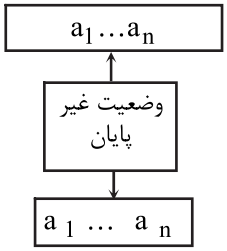
بعد از خواندن یک حرف



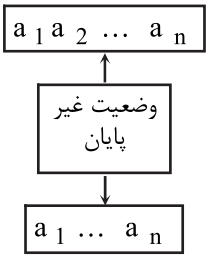
در آغاز محاسبه



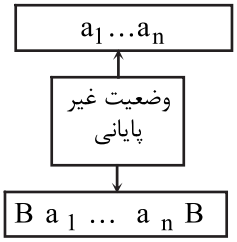
وقتی به آخرین نماد که به طور هم‌زمان خوانده شده



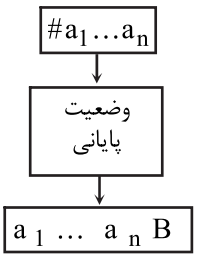
در آغاز خواندن هم‌زمان نوارها



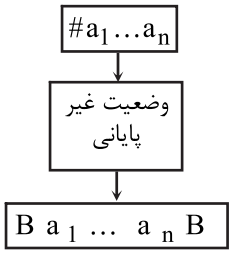
پس از خواندن عکس عبارت



پایان محاسبه



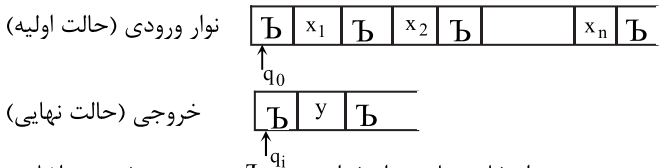
بعد از کامل کردن خواندن نوار



❖ **تعریف ۷: ماشین تورینگ محاسبه‌گر**

اگر $M=(Q,\Sigma,P,q_0,\delta,\{h\})$ یک ماشین تورینگ باشد که $h\in Q$ و $h\neq q_0$ و $0,1\in\Sigma$ است، در این‌صورت تابع $f:N^k\to N$ که $k\geq 1$ توسط ماشین M محاسبه‌پذیر است، اگر $M(w_1;\cdots;w_k)|=f(|w_1|,\cdots,|w_k|)$ باشد که در اینجا $|M(\cdots)|$ بخشی از نوار خروجی است به تابع چندتایی که توسط ماشین تورینگ M محاسبه می‌شود محاسبه‌پذیر گویند. به طور کلی یک تابع f را روی دامنه D محاسبه‌پذیر تورینگ یا قابل محاسبه می‌گوییم اگر $M=(Q,\Sigma,\mu,q_0,\delta,F)$ یک ماشین تورینگ باشد که برای هر $w\in D$ ماشین به یک وضعیت نهایی برود.

این نوع ماشین می‌تواند توابعی با پارامترهای مختلف را محاسبه کند به این صورت که پارامترهای ورودی از چپ به راست روی نوار واقع می‌شوند و هر پارامتر توسط یک # از پارامتر بعدی جدا می‌شود. به طور مثال برای تابع $y=f(x_1,x_2,...x_n)$ داریم:

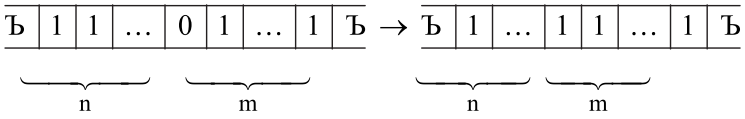


ماشین تورینگ محاسبه‌گر، خروجی را روی نوار ذخیره می‌کند؛ پس ورودی از بین می‌رود. پس از خاتمه باید، نوار خواندن، به Ъ سمت چپ خروجی اشاره کند. این ماشین، ابتدا به فرآیندی جهت نمایش اعداد طبیعی نیاز دارد. نمونه‌ای از سیستم کدگذاری اعداد طبیعی در زیر آمده است.

عدد	0	1	2	3	4	…	n
نحوه نمایش	1	11	111	1111	11111	…	111111…1 (n+1)

مثال ۱۴: ماشین محاسبه‌گر Add(n,m) را روی الفبای $\sum=\{1\}$ طراحی کنید.

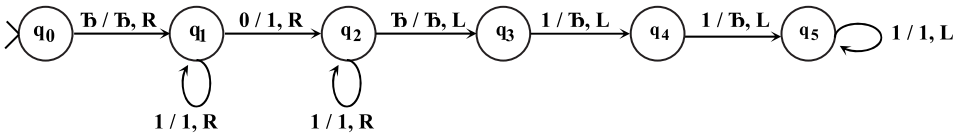
☑ **پاسخ:** پارامترهای تابع add با 0 از هم جدا شده‌اند:



تابع δ این ماشین به شکل زیر است:

$1:\delta(q_0,1)=(q_0,1,R)$
$2:\delta(q_0,0)=(q_1,1,R)$
$3:\delta(q_1,1)=(q_1,1,R)$
$4:\delta(q_1,\#)=(q_2,\#,L)$
$5:\delta(q_2,1)=(q_f,\#,L)$

پس از خواندن 1های مربوط به n، با دیدن 0 بین m و n روی نوار 1 نوشته شده سپس به راست می‌رویم. بالاخره پس از m تا 1 دیگر به Ъ می‌رسیم. از این به بعد با دیدنЪ و دوتا 1، Ъ نوشته و به چپ برمی‌گردیم تا به ابتدای نوار برسیم.



حال ماشین فوق را روی مثال $x=2$ و $y=3$ اجرا می‌کنیم:

$\#q_0110111\# \vdash_1 \#1q_010111\# \vdash_1 \#11q_00111\# \vdash_2$
$\#111q_1111\# \vdash_3 \#1111q_111\# \vdash_3 \#11111q_11\# \vdash_3$
$\#111111q_1\# \vdash_4 \#11111q_21\# \vdash_5 \#11111q_f\#$

❖ **تعریف۸:** اگر ماشین تورینگ M وجود داشته باشد تابع f با قلمرو D، محاسبه‌گر یا محاسبه‌پذیر به وسیله تورینگ گفته می‌شود، به طوری که به ازای تمام مقادیر $w\in D$ داشته باشیم: $q_0\,w\overset{*}{\underset{M}{\vdash}}q_f\,f(w),q_f\in F$

ماشین‌های تورینگ دارای گزینه‌ی توقف (Stop Stay Option)

بنا به تعریف ماشین تورینگ استاندارد، هد خواندن ـ نوشتن باید به راست یا چپ حرکت کند.

گاهی اوقات انتخاب سومی هم مورد نیاز است تا بر اساس آن، هد خواندن ـ نوشتن پس از بازنویسی یک خانه از نوار، در جای خود باقی بماند، لذا می‌توان یک ماشین تورینگ که دارای گزینه توقف باشد تعریف کرد.

زبان کلاس NP

❖ **تعریف ۶:** ماشین تورینگ نامعین $M=(Q,\Sigma,P,\delta,q_0)$ چند جمله‌ای محدود گفته می‌شود، در شرایطی که اگر یک چند جمله‌ای $P(n)$ وجود داشته باشد، به‌طوری که برای هر نمونه ورودی S، هیچ رشته e وجود نداشته باشد که:

$$\#q_0s\# \overset{P(n)+1}{\underset{M}{\vdash\!\!\!\!-\!\!\!\!-}} s'q_ie\#$$

به‌عبارت دقیق‌تر، تمام محاسبات این ماشین ـ برای هر نوع ورودی ـ تعداد مراحل محاسباتی که بیش از چندجمله‌ای باشد ندارد. این مجموعه زبان‌ها، کلاس NP را تشکیل می‌دهند. مراحل محاسباتی تورینگ نامعین پذیرنده هر زبان در کلاس NP بیش از $P(n)$ نخواهد بود.

در حقیقت NP به کلاسی از زبان‌ها گفته می‌شود که دارای اثبات‌کننده‌های صحت (Verifier) چندجمله‌ای هستند. از آن‌جا که این کلاس شامل مسایل واقعی و جالب زیادی می‌باشد، حائز اهمیت است. البته می‌توان کلاس NP را برحسب ماشین‌های تورینگ چندجمله‌ای غیرقطعی (نامعین) تعریف نمود.

مساله‌های(HAMPATH) و فروشنده دوره‌گرد (TSP) از مسایل مهم کلاس NP است. ماشین تورینگ غیرقطعی N_1 برای مساله HAMPATH در ذیل آمده است:

ورودی: $<G,s,t>$ که G گراف جهت‌دار حاوی گره‌های s و t می‌باشد.

۱ـ لیستی از m شماره $P_1,P_2,...,P_m$ را می‌نویسیم (m تعداد گره‌های G است). هر عدد لیست به‌طور غیرقطعی بین ۱ تا m انتخاب می‌شود.

۲ـ تکرار لیست را چک می‌کنیم. اگر چنین چیزی یافت شد Reject رخ داده است.

۳ـ $P_m=t$ و $P_1=s$، اگر هر کدام از این‌ها وجود نداشت Reject می‌شود.

۴ـ هر $1\le i\le m-1$، را چک می‌کنیم تا دریابیم (P_i,P_{i+1}) لبه‌ای از G هست یا خیر. اگر بود Accept و در غیر این‌صورت Reject می‌شود.

پیچیدگی ماشین N_1 عبارت است از:

مرحله‌ی ۱، زمان $O(m)$ صرف می‌کند.

مراحل ۲ و ۳، چک‌های ساده است که زمان چندجمله‌ای $O(m)$ صرف می‌کنند.

مرحله‌ی ۴، در زمان $O(m^3)$ اجرا می‌گردد.

$$T_{N_1}(m)=O(m^3)$$

قضیه ۸: اگر زبانی به‌وسیله ماشین تورینگ چندجمله‌ای غیرقطعی تصمیم‌گیری شود یک زبان در کلاس NP است.

اثبات: برای زبان A و ماشین تورینگ غیرقطعی چندجمله‌ای N دو مرحله اثبات داریم:

الف) $A\in NP$ لذا N برای A تصمیم می‌گیرد

فرض کنید V ثابت‌کننده چندجمله‌ای برای A باشد. به عبارت دیگر V ماشین تورینگی است که در زمان n^k اجرا می‌شود (n طول رشته ورودی w است).

طی مراحل زیر از V برای ساخت ماشین N استفاده می‌شود (ورودی: رشته w به طول n):

۱ـ انتخاب غیرقطعی (حدسی) رشته c به طول حداکثر n^k؛

۲ـ اجرای V روی ورودی (w,c)؛

۳ـ اگر V را پذیرفت، "Accept" در غیر این‌صورت "Reject" می‌شود.

ب) A توسط ماشین N تصمیم‌گیری می‌شود در نتیجه $A\in NP$.

برای ساخت اثبات کننده‌ی چندجمله‌ای V، طبق روال زیر از N استفاده می‌شود:

۱ـ شبیه‌سازی N روی ورودی w، بررسی علامت c برای توصیف انتخاب‌های غیرقطعی انجام شده در هر گام.

۲ـ اگر چنین شاخه‌ای از محاسبه N پذیرفته شود "Accept" در غیر این‌صورت "Reject" رخ می‌دهد.

❖ **تعریف ۷: تابع NTIME:**

$$NTIME(t(n))=\{L\mid L \text{ زبانی است تصمیم‌پذیر که به‌وسیله‌ی ماشین تورینگ غیرقطعی با رتبه } O(t(n)) \text{ است.}\}$$

قضیه ۹: درباره‌ی کلاس NP داریم: $NP=\bigcup_{K\ge 0} NTIME(n^k)$

❖ **تعریف ۸:** اگر چند جمله‌ای مانند $P(n)$ موجود باشد، ماشین تورینگ $M=(Q,\Sigma,P,\delta,q_0)$ را محدود توانی می‌گویند. به‌طوری‌که تمام رشته‌های ورودی s، رشته ورودی e را نداشته باشد به‌طوری‌که:

$$\#q_0s\# \overset{2^{P(n)+1}}{\underset{M}{\vdash\!\!\!\!-\!\!\!\!-}} \#q_fe\#$$

یعنی ماشین پس از محاسبات توانی متوقف می‌شود. مجموعه زبان‌های محدود توانی را EXP می‌نامیم.

قضیه ۱۰: اگر $L\in NP$ آنگاه $L\in EXP$ است.

قضیه ۱۱: کلاس‌های مختلف معرفی شده به صورت زیر با هم در ارتباط هستند:

$$P \subseteq NP \subseteq EXP$$

بحث در مورد کلاس‌های P و NP وابسته به زمان اجرای ماشین است. می‌توان با روش‌هایی مانند قطری‌سازی نشان داد زبان‌های قضیه ۱ در P نیستند و این با توسعه زمان اجرای محاسبات تایید می‌شود. در ادامه، با معرفی کاهش‌پذیری تورینگ‌ها سعی خواهیم کرد کلاس‌های پیچیدگی را به زبان‌های تصمیم‌ناپذیر مانند بازگشتی برشمردنی تعمیم می‌دهیم.

برخی روابط بین دو کلاس پیچیدگی DTIME و NTIME عبارتست از:

الف) $DTIME(T(n)) \subseteq NTIME(T(n))$

ب) $T_1(n) = O(T_2(n) \Rightarrow DTIME(T_1(n)) \subseteq DTIME(T_2(n))$

قضیه ۱۲: به ازای مقدار صحیح $k > 0$، داریم: $DTIME(n^k) \subset DTIME(n^{k+1})$

با استناد به قضیه‌ی فوق ادعا می‌شود زبان‌هایی در $DTIME(n^3)$ قرار دارند که عضو $DTIME(n^2)$ نیستند.

❖ **تعریف ۹:** کلیک (Clique) در گراف عبارت است از زیرگرافی که هر دو گره آن با یک لبه به هم متصل است. K-Clique کلیکی به اندازه‌ی k است.

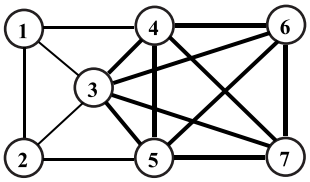
مساله CLIQUE به‌صورت زیر تعریف می‌شود:

$CLIQUE = \{< G, k > |$ G یک گراف غیر جهت‌دار حاوی یک k-clique می‌باشد $\}$

همان‌طور که در شکل، برای گراف G می‌بینیم 5-clique وجود داشته و 6 - clique وجود ندارد.

$< G,5 > \in CLIQUE$

$< G,6 > \notin CLIQUE$



شکل ۱. گراف G و مساله clique

❖ **تعریف ۱۰:** پوشش گره (Vertex Cover): مجموعه گره‌هایی است که تمام لبه را می‌پوشانند، مثل $\{3,4,5\}$ در G.

قضیه ۱۳: ثابت کنید $CLIQUE \in NP$

اثبات: برای اثبات از ثابت‌کننده V روی ورودی $< G,k >$ و متغیر c استفاده شده و مراحل ذیل را طی می‌کند:

مرحله‌ی ۱ـ تست این است که c مجموعه‌ای از k گره مختلف در G باشد.

مرحله‌ی ۲ـ تست این‌که G شامل تمام لبه‌هایی است که گره‌های موجود در c را به‌هم وصل می‌کند.

مرحله‌ی ۳ـ اگر هر دو تست موفق بود "Accept" و در غیر این‌صورت "Reject" را خواهیم داشت.

تحلیل الگوریتم: اگر گراف G حاوی m گره باشد، $\binom{m}{k}$ زیر مجموعه خواهیم داشت:

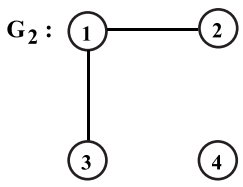
مرحله‌ی ۱ـ رتبه‌ی زمانی $O(k) \times O(m) = O(km)$ صرف می‌کند.

مرحله‌ی ۲ـ مرتبه‌ی اجرایی $O(k^2) \times O(m^2) = O(k^2 m^2)$ هزینه دارد.

تعریف مساله IS (Independent Set): تعریف رسمی از مجموعه مستقل در گراف G چنین است:

$IS = \{< G,k > |$ G گرافی شامل یک مجموعه مستقل از k گره می‌باشد$\}$

در گراف G_2 مقابل $\{2,3,4\}, \{1,4\}$ و $\{1\}$ مجموعه‌های مجزا (IS) و $\{1,2\}, \{1,3\}, \{4\}$ کلیک هستند.



قضیه ۱۴: اگر IS ماشین تورینگی با رتبه‌ی زمانی چندجمله‌ای داشته باشد، CLIQUE نیز همین‌گونه خواهد بود.

نتیجه ۲: ۱ـ اگر گراف G دارای یک IS با سایز k باشد، آن‌گاه G یک کلیک با سایز k نیز خواهد داشت. ۲ـ اگر G فاقد IS به طول k باشد، CLIQUE به طول k نیز نخواهد داشت.

تعریف مساله SUBSET-SUM:

مجموعه‌ای از k عدد $x_i (1 \le i \le k)$ و متغیر هدف t را درنظر بگیرید. مساله این است که آیا زیرمجموعه‌ای از k عدد فوق‌الذکر وجود دارد که مجموع عناصرش t شود؟

$$SUBSET - SUM = \{< s,t > | s = \{x_1,...,x_k\}, \{y_1,...,y_L\} \subseteq \{x_1,...,x_k\} | \sum_{i=1}^{L} y_i = t\}$$

قضیه ۱۵: ثابت کنید $SUBSET - SUM \in NP$

اثبات: متغیر c و $< s,t >$ ورودی اثبات‌کننده V فرض شده‌اند. مراحل عملیات، چنین است:

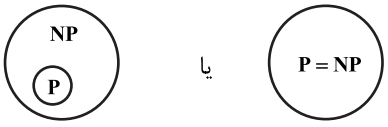
مرحله ۱ ـ تست این‌که c مجموعه‌ای از اعداد است که جمع آن‌ها برابر t می‌گردد.

مرحله ۲ ـ تست این‌که هر عدد در c به S تعلق دارد.

مرحله ۳ ـ اگر مراحل فوق بدون مشکل اجرا شدند Accept و در غیر این‌صورت Reject شده است.

تحلیل الگوریتم: وقتی $|S| = m$ و $|c| \le m$ باشد، این الگوریتم $O(m^2)$ زمان می‌برد.

سوال مهم: آیا P = NP است یا $P \neq NP$؟



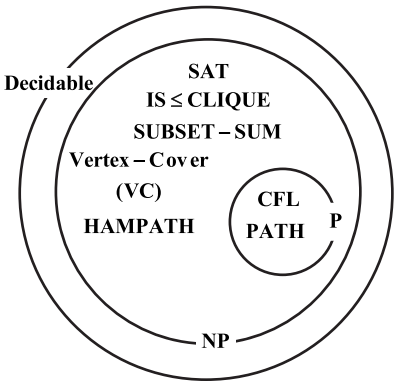
زبان‌هایی که در کلاس P قرار دارند از تصمیم‌گیرنده‌های با زمان چند جمله‌ای برخوردار هستند؛ در حالی که زبان‌های کلاس NP دارای ثابت‌کننده‌ای در زمان چند جمله‌ای خواهند بود. پاسخ به سوال $P = NP$ یکی از رازهای حل نشده در علوم کامپیوتر به‌شمار می‌رود به طوری که بسیاری از محققان معتقدند $P \neq NP$ است و برخی نیز خلاف آن را پذیرفته‌اند.

آنچه مهم است اینکه اگر $P \neq NP$ باشد آنگاه خواهیم داشت:

الف ـ زبان‌های کلاس P رام شدنی (Tractable) است، یعنی در زمان چند جمله‌ای قابل حل می‌باشد.

ب ـ زبان‌های مجموعه‌ی NP-P رام نشدنی بوده و فقط در زمان نمایی حل می‌شوند.

دید کلی درباره زبان‌های کلاس NP و P همراه با برخی نمونه‌های مطرح شده در این فصل در شکل زیر آمده است:



شکل ۲. دیاگرام کلی مسائل NP و P

البته تا کنون ثابت نشده زبانی در کلاس NP وجود دارد که به‌طور قطع در P نیست.

هر چند در فصل سوم کاهش‌پذیری مورد بررسی قرار گرفت اما در ادامه بحث مبانی و مفهوم نگاشت کاهش‌پذیر زمان چند جمله‌ای که تقریباً معادل تعریف فصل سوم خواهد بود ارائه می‌شود.

توابع محاسبه‌پذیر زمان چندجمله‌ای و کاهش‌پذیری:

اگر ماشین تورینگ M وجود داشته باشد تابع $f : \Sigma_1^* \to \Sigma_2^*$ را محاسبه‌پذیر زمان چند جمله‌ای (Polynomial-Time Computable) می‌گویند، به طوری‌که:

الف ـ با ورودی $w \in \Sigma_1^*$ آغاز شود؛

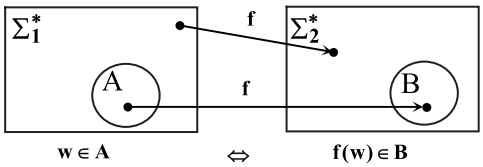
ب ـ فقط با $f(w) \in \Sigma_2^*$ روی نوار متوقف شود؛

ج ـ در زمان چندجمله‌ای اجرا شود.

تعریف نگاشت کاهش‌پذیر با زمان چند جمله‌ای:

زبان A روی الفبای Σ_1 و زبان B روی Σ_2 مفروض است. A یک نگاشت کاهش‌پذیر با زمان چند جمله‌ای به B بوده و به فرم $A \le_m B$ نوشته می‌شود اگر تابع محاسبه‌پذیری با زمان چند جمله‌ای $(f : \Sigma_1^* \to \Sigma_2^*)$ موجود باشد به طوری‌که به ازای هر $w \in \Sigma_1^*$ داشته باشیم:

$w \in A \Leftrightarrow f(w) \in B$



شکل ۳. کاهش‌پذیری در زمان چندجمله‌ای از زبان A به B

قضیه ۱۶: اگر $A \le_P B$ و $B \in P$ باشد آن‌گاه $A \in P$ خواهد بود.

قضیه ۱۷: ثابت کنید که $3SAT \le_P CLIQUE$ قابل انجام است.

از قضیه فوق می‌توان مدعی شد اگر ϕ قابل ارضاء (Satisfiable) باشد، آن‌گاه گراف G دارای k-clique است.

همچنین می‌توان ادعا کرد: نگاشت $\phi \to G$ در زمان چندجمله‌ای قابل محاسبه است. بدین ترتیب که اگر ϕ تابع 3CNF دارای k گزاره و m متغیر باشد گراف G حاوی $3k$ گره و تعداد لبه کمتر از $\binom{3k}{2} = O(k^2)$ ساخته می‌شود.

$$\text{تعداد لبه‌های گراف } G < \binom{3k}{2} = \frac{3k(3k-1)}{2} = O(k^2)$$

اندازه گراف G براساس اندازه تابع ϕ، 3CNF به‌صورت چندجمله‌ای خواهد بود.

قضیه ۱۸: اگر R_1 کاهش چند جمله‌ای از زبان L_1 به L_2 باشد و R_2 هم کاهش چند جمله‌ای از L_2 به L_3؛ آن‌گاه $R_1 o R_2$ کاهش چند جمله‌ای از L_1 به L_3 خواهد بود.

❖ **تعریف ۱۱: زبان $L \subseteq \Sigma^*$ زبان NP – complete است اگر**

الف: $L \in NP$ باشد.

ب: برای هر زبان $L' \in NP$ یک کاهش چند جمله‌ای از L' به L وجود داشته باشد. با حذف شرط الف، زبان L در کلاس NP-hard خواهد بود.

قضیه ۱۹: فرض کنید L_1 زبان $NP-complete$ باشد، آن‌گاه اگر $L_1 \in P$ باشد $P = NP$ خواهد بود.

قضیه ۲۰: هر یک از زبان‌های زیر $NP-complete$ است.

الف ـ $L_{SAT} \in NP-complete$

ب ـ $L_{MAXSAT} \in NP-complete$

ج ـ $L_{3SAT} \in NP-complete$

د ـ $L_{Exact\,Cover} \in NP-complete$

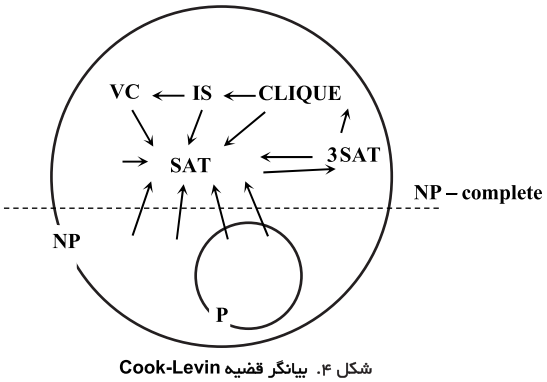
هـ ـ $L_{Hamiltonian\,Path} \in NP-complete$

تمرین: ثابت کنید مساله $Two-CLIQUE = \{< G,K > \mid$ گراف غیرجهت‌دار G دو کلیک مجزا به طول k دارد $\}$ یک مساله در کلاس NP-Complete است.

قضیه ۲۱: اگر B در کلاس NP-Complete بوده و $B \leq_P C$ باشد $(C \in NP)$، آن‌گاه $C \in NP-complete$ خواهد بود.

طبق قضیه قبل، با داشتن مساله NP-Complete به کمک کاهش‌پذیری در زمان چندجمله‌ای می‌توان سایر زبان‌های کلاس NP-Complete را شناسایی کرد. نکته مهم شناسایی مساله NP-Complete می‌باشد که نخست توسط Cook-Levin مطرح شد و به‌عنوان قضیه Cook-Levin شناخته می‌شود. بدین ترتیب مساله SAT در کلاس NP-Complete است. همچنین می‌توان نشان داد برای هر زبان $A \in NP$ خواهیم داشت $A \leq_P SAT$.

به عبارتی هر مساله‌ای در کلاس NP به مساله SAT قابل کاهش است. اگر کاهش‌پذیری مساله A به مساله B با علامت $A \to B$ نشان داده شود قضیه کوک، طبق شکل زیر خواهد بود:



شکل ۴. بیانگر قضیه Cook-Levin

مثال مسئله تطابق پست

فرض کنید S_1 و S_2 دو دنباله از رشته‌ها روی الفبای Σ باشند؛ به این‌صورت:

$$S_1 = s_1, s_2, ..., s_n$$
$$S_2 = s'_1, s'_2, ...s'_n$$

در تطابق پست بررسی می‌کنیم آیا جایگشتی وجود دارد که S_1 و S_2 را بر هم منطبق کند یا خیر. به عبارت دیگر، آیا دنباله‌ای از اعداد طبیعی مانند $i, j, ..., k$ وجود دارد که بتوان طبق آن، با کنار هم گذاشتن رشته‌های هر دنباله این دو بر هم منطبق شوند؛ یعنی: $s_i, s_j ....s_k = s'_i.s'_j ...s'_k$

دقت کنید s_1, s_2 با $s_1 s_2$ فرق می‌کند. در اینجا $s_i s_j$ رشته از الحاق s_i و s_j ساخته شده است.

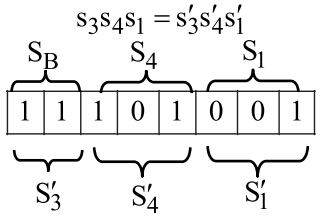
اگر بخواهیم برای این مساله که آیا دنباله‌ای مانند $i,j,....k$ وجود دارد که تطابق پست برقرار شود، یک الگوریتم یا ماشین تورینگ معرفی کنیم ـ که مساله را برای هر دو دنباله مانند s_1 و s_2 بررسی و امکان یا عدم امکان آن بررسی کند ـ کار چندان مشکلی نخواهیم داشت؛ برای مثال ابتدا، دنباله‌های s_1 و s_2 را بر اساس این‌که عناصر آنها با چه نشانه‌ای شروع شده‌اند، دسته‌بندی کنیم. چرا که s_i و s'_i باید از کاراکتر اول مثل هم باشند. به پاسخی که درنهایت به‌دست می‌آید $PC-solution$ می‌گویند. در ادامه با یک مثال نشان می‌دهیم $PC-solution$ چه می‌باشد؟

مثال ۵: فرض کنید $\Sigma=\{0,1\}$ و داشته باشیم:

$S_1: s_1=001 \quad s_2=0011 \quad s_3=11 \quad s_4=101$

$S_2: s'_1=01 \quad s'_2=111 \quad s'_3=111 \quad s'_4=010$

اگر دنباله $i=3,j=4,k=1$ در این‌صورت:



حال ما یک PC-solution را در اختیار خواهیم داشت. با این وجود مسأله تطابق پست یک مسأله تصمیم‌ناپذیر بوده و از اهمیت زیادی نیز برخوردار است.

سؤالات برگزیده فصل چهارم

مثال ۶: اگر $A \le_m B$ بوده و B یک زبان منظم (Regular) باشد، آیا زبان A نیز منظم خواهد بود؟

☑ **پاسخ:** خیر، زبان نامنظم $A=\{0^n1^n \mid n\ge 0\}$ مجموعه $B=\{1\}$ و تابع f را در نظر بگیرید:

$$f(x)=\begin{cases}1 & ,x=0^n1^n\\ 11 & ,else\end{cases}$$

در این‌صورت اولاً $x\in A \Leftrightarrow f(x)\in B$ بوده و ثانیا f یک تابع محاسبه‌پذیر است؛ لذا $A \le_m B$. با این وجود B زبان منظمی است ولی A زبان منظمی نیست.

مثال ۷: فرض کنید $REG_{TM} \le B$ باشد. آیا B تصمیم‌پذیر است؟

توجه شود:

$REG_{TM}=\{< M > \mid$ ماشین تورینگ M که زبان آن منظم است$\}$

☑ **پاسخ:** از آنجا که مساله REG_{TM} تصمیم‌ناپذیر و قابل کاهش به B است، یعنی $REG_{TM} \le_m B$، لذا B نیز تصمیم‌ناپذیر خواهد بود.

مثال ۸ : با فرض $\{D$ یک DFA و P یک PDA بوده و $DFA_{PDA}=\{D,P> \mid L(D)\subseteq L(P)\}$ نشان دهید $ALL_{PDA} \le_m DFA_{PDA}$ است یا خیر؟ و همچنین آیا DFA_{PDA} تصمیم‌پذیر است؟

☑ **پاسخ:** فرض کنید ماشین پشته‌ای $< P >$ نمونه‌ای از ALL_{PDA} است و $f(< P >)=(D,f)$ که D یک DFA می‌باشد که تمام رشته‌ها را می‌پذیرد. آن‌گاه خواهیم داشت:

$$< P >\in ALL_{PDA} \Leftrightarrow L(P)=\Sigma^* \Leftrightarrow L(D)\subseteq L(P) \Leftrightarrow f(< p >)\in DFA_{PDA}$$

چون ALL_{PDA} تصمیم‌پذیر نیست لذا DFA_{PDA} نیز تصمیم‌پذیر نمی‌باشد.

مثال ۹: آیا $L=\{< M > \mid$ که $L(M)$ دارای حداقل ۱۲ عضو می‌باشد$\}$ تصمیم‌پذیر است؟

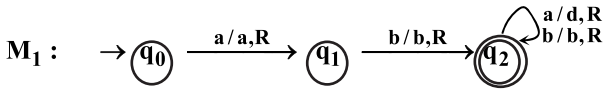
☑ **پاسخ:** خیر. طبق قضیه Rice، L تصمیم‌پذیر نیست. فرض کنید M_0 تورینگ پذیرنده همه‌چیز است و M_1 ماشین تورینگ رد کننده همه چیز. آن‌گاه خواهیم داشت: $< M_1 >\in L, < M_0 >\notin L$ همچنین اگر داشته باشیم $L(M_1)=L(M_2)$ آنگاه به نتایج زیر خواهیم رسید.

$$< M_1 >\in L \Leftrightarrow L(M_1) \text{ حداقل ۱۲ عضو داشته} \Leftrightarrow L(M_2) \text{ حداقل ۱۲ عضو داشته} \Leftrightarrow < M_2 >\in L$$

لذا قضیه Rice روی L قابل اجرا است.

مثال ۱۰: ماشین تورینگی طراحی کنید که روی الفبای $\{a,b\}$ زبان $L=\{ab(a+b)^*\}$ را بپذیرد.

☑ **پاسخ:** این ماشین برای پذیرش (نیاز) به خواندن کل رشته ورودی ندارد، زیرا می‌دانیم، پس از خواندن ab، هر فرمی از الفبای ورودی می‌تواند در رشته بیاید. پس ماشین تورینگ مد نظر چنین خواهد شد:



مثال ۱۱: الگوریتمی ارائه کنید که تورینگ سه نواره تابع $f(x)=x^2$ را محاسبه کند.

☑ **پاسخ:** ورودی الگوریتم: عدد x واقع بر نوار اول

مرحله ۱ـ x را روی نوار دوم و سوم کپی کنید.

مرحله ۲ـ محتوای نوار دوم را به انتهای محتوای نوار اول اضافه کنید؛

مرحله ۳ـ یک 1 را از رشته نوار سوم بردارید:

مرحله ۴ـ اگر هنوز بیش از کاراکتر 1 روی نوار سوم وجود دارد به مرحله ۲ بر می‌گردیم.

مثال ۱۲: ثابت کنید زبان L، زبان بازگشتی شمارش‌پذیر (RE) است اگر و فقط اگر بتوان آن‌را به فرم ذیل بیان شود:

$L=\{x|\exists y :< x,y >\in R\}$ زبانی تصمیم‌پذیر است و R

☑ **پاسخ:** ۱) فرض: زبان L با مشخصات مساله داده شده است.

حکم: از آنجا که ورودی x داده شده، به‌سادگی می‌توان تمام yها را به ترتیب الفبایی شمرد و برای هر کدام تصمیم گرفت آیا $(x,y)\in R$ می‌باشد یا $(x,y)\notin R$. اگر پاسخ، yes باشد x پذیرفته شده و در غیر این صورت کار ادامه می‌یابد، پس واضح است زبان پذیرفته شده دقیقاً همان L است.

۲) فرض: زبان L بازگشتی شمارش‌پذیر (RE)

حکم: تعیین شمارنده E برای L

راه حل: E ماشینی است که یک رشته (نامتناهی) را روی نوارش می‌نویسد، با این تضمین که مجموعه‌ی این رشته‌ها دقیقاً L است. حال R را به فرم زیر تعریف می‌کنیم:

$$R=\{(x,y)|\ x\}$$ یک خروجی در اولین y گام از اجرای E می‌باشد.

واضح است که R تصمیم‌پذیر می‌باشد، چون می‌توان ماشین E را جهت تصمیم‌گیری در خصوص $(x,y)\in R$ برای y گام شبیه‌سازی کرد. پس L دقیقاً مجموعه X هایی است که برای آن‌ها y هایی وجود دارد؛ به‌طوری‌که $(x,y)\in R$. به عبارتی رشته X در زبان L است اگر (به کمک شمارنده E) به‌عنوان خروجی مطرح شود.

مثال ۱۳: کدام گزینه درباره زبان‌های L_1 و L_2 صحیح است؟ (G یک گرامر مستقل از متن و E یک عبارت باقاعده است)

۱) هر دو تصمیم‌پذیر هستند. $L_1=\{(G,E)|L(G)\subseteq L(E)\}$ ۲) هر دو تصمیم ناپذیرند. $L_2=\{(G,E)|L(E)\subseteq L(G)\}$

۳) L_1 تصمیم‌ناپذیر و L_2 تصمیم‌پذیر است. ۴) L_1 تصمیم‌ناپذیر و L_2 تصمیم‌ناپذیر است.

☑ **پاسخ:** گزینه «۳» می‌توان این مسئله را به E_{CFG} (تُهی بودن گرامرهای مستقل از متن) کاهش داد چرا که؛ این مسئله تصمیم‌پذیر است. می‌توان برای E مفروض، DFA طراحی کرد که زبان A را شناسایی کند $(\overline{L(E)})$؛ همین‌طور می‌توان NPDA طراحی کرد که زبان $L(G)\cap A$ را شناسایی کند. حال می‌توان بررسی کرد که آیا B خالی است یا خیر؟ از طرفی، عملیات مکمل و اشتراک، محاسبه‌پذیر (computable) هستند؛ در نتیجه B تهی است اگر و فقط اگر $L(G)\subseteq L(E)$ و لذا L_1 تصمیم‌پذیر می‌باشد. درباره زبان L_2 می‌توان ALL_{CFG} را به L_2 کاهش داد. مجموعه $E=\Sigma^*$ با نمونه G از ALL_{CFG} مفروض است. می‌توان زوج (E,G) را تولید کرد. اگر $G=\Sigma^*$ آن‌گاه $L(E)\subseteq L(G)$ خواهد شد. اگر $G\neq\Sigma^*$ باشد آن‌گاه $L(E)\not\subseteq L(G)$ بنابراین مساله ALL_{CFG} به L_2 کاهش یافت. طبق قضایای متن درس می‌دانیم ALL_{CFG} تصمیم‌پذیر است، لذا L_2 نیز تصمیم‌ناپذیر می‌باشد.

❖ **تعریف ۱۲:** دو زبان مُجزای L_1 و L_2 را تفکیک‌پذیر بازگشتی (Recursively Separable) می‌گوییم. اگر یک زبان تصمیم‌پذیر (D) وجود داشته باشد به‌طوری‌که $L_2\subseteq D$ و $L_1\cap D=\varnothing$. L_1 و L_2 را تفکیک‌ناپذیر بازگشتی می‌گوییم اگر چنین زبان تصمیم‌پذیری (D) موجود نباشد. می‌توان ادعا کرد زبان تصمیم‌ناپذیر و مکملش، تفکیک‌ناپذیر بازگشتی هستند.

مثال ۱۴: زبان‌های L_1 و L_2 مقابل مفروض هستند:

$L_1 = \{< M > \mid$ **توقف کرده و ورودی** $< M >$ **را می‌پذیرد** $M\}$

$L_2 = \{< M > \mid$ **توقف کرده و ورودی** $< M >$ **را نمی‌پذیرد** $M\}$

ثابت کنید L_1 و L_2 تفکیک‌ناپذیر بازگشتی می‌باشند.

☑ **پاسخ:** فرض کنید زبان D تصمیم‌پذیر است به‌طوری‌که $L_2 \subseteq CD$ و $L_1 \cap D = \phi$ باشد (ماشین تورینگ متناظر M_D). در این‌جا از برهان خلف استفاده می‌کنیم.

فرض می‌شود $M_D(< M_D >)$ ورودی را می‌پذیرد. باید توجه داشت $< M_D >$ در زبان D است. لذا طبق تعریف L_1، $< M_D >$ در زبان L_1 می‌باشد، که مخالف فرض $L_1 \cap D = \phi$ خواهد بود.

حال فرض کنید $M_D(< M_D >)$ ورودی را نپذیرد (حالت Reject) یعنی $< M_D >$ در زبان D نیست، پس طبق تعریف L_2، $< M_D >$ در زبان L_2 بوده که مخالف $L_2 \subseteq D$ می‌باشد، لذا $L_2 \subseteq D$ می‌باشد، پس L_1 و L_2 تفکیک‌ناپذیر بازگشتی هستند.

مثال ۱۵: زبان L مفروض است. زبان MAJ_L به فرم زیر تعریف می‌شود:

$$MAJ_L = \{\#x_1\#x_2\#...\#x_k \mid (k \geq 0) AND (|\{i \mid x_i \in L\}| > \frac{k}{2})\}$$

ثابت کنید MAJ_L زبان بازگشتی شمارا (RE) است اگر فقط L بازگشتی شمارا (RE) باشد. توجه شود در این‌جا x_iها رشته‌هایی روی الفبای L بوده و # در الفبای L نیست.

☑ **پاسخ:** فرض کنید M یک شناسایی کننده (Recognizer) برای L باشد. و روی $\#x_1\#x_2\#...\#x_k\#$ $(K \geq 0)$ داده شده است. می‌توان M را به‌طور موازی، روی هر x_i شبیه‌سازی کرد و بیش از $\frac{k}{2}$ شبیه‌سازی‌ها با پذیرش همراه باشد. به ازای $M, j = 1,2,3,...$، روی هر x_i، برای j گام شبیه‌سازی می‌کنیم؛ اگر بیش از $\frac{k}{2}$ این شبیه‌سازی‌ها متوقف یا و پذیرش شوند، آنگاه ما توقف کرده و پذیرفته شده است.

واضح است تا زمان j اگر اکثریت x_i در L باشند، ماکزیمم تعداد گام‌های مورد نیاز، برای پذیرش این‌که x_iها در L هستند، خواهیم پذیرفت. در غیر این‌صورت در بین این شبیه‌سازی‌ها هرگز بیش از $\frac{k}{2}$ پذیرش برای هر j میسر نبوده و پذیرش نخواهیم داشت.

نکته ۲: همواره داریم: تمام زبان‌ها $Decidable \subset RE \subset$

همچنین برخی از مهمترین مسایل تصمیم ناپذیر عبارتند از:

آیا $L = \varnothing$ می‌باشد؟ (L زبان حساس به متن است)

برای زبان L مستقل از متن یا حساس به متن آیا $L = \Sigma^*$ می‌باشد یا خیر؟

به ازای هر دو زبان L_1 و L_2 مستقل از متن یا وابسته (حساس) به متن:

الف ـ آیا $L_1 = L_2$ است؟

ب ـ آیا $L_1 \subseteq L_2$ است؟

ج ـ آیا $L_1 \cap L_2 = \varphi$ است؟

به ازای L مستقل از متن باشد آیا $\overline{L}$ نیز مستقل از متن است؟

به ازای L_1 و L_2 مستقل از متن، زبان $L_1 \cap L_2$ نیز مستقل از متن می‌باشد یا خیر؟ آیا $L = L^2$ است یا خیر؟

آیا n متناهی وجود دارد؛ به‌طوری که $L^n = L^{n+1}$؟

اگر L_1 و L_2، CF_L باشد آیا L_1/L_2 متناهی است؟ عضویت در L_1/L_2

اگر L یک زبان مبهم باشد آیا L یک زبان مستقل از متن است؟

مثال ۱۶: کدام عبارت صحیح است؟

الف) مجموعه زبان‌های قابل تصمیم‌گیری (Decidable) یا تصمیم‌پذیر، تحت عمل مکمل بسته هستند.

ب) ابر کامپیوترها (Super Computers) می‌توانند بیشتر از ماشین‌های تورینگ روی زبان‌های تصمیم بگیرند.

ج) مجموعه $\{M$ روی رشته w توقف نمی‌کند $\mid \langle M, w \rangle\} = A$ به‌طور بازگشتی شمارا (RE) است.

د) اگر زبان L به‌طور بازگشتی شمارا (RE) باشد آن‌گاه L تصمیم‌پذیر است.

۱) الف و د　　۲) ب و د　　۳) فقط ج　　۴) فقط الف

☑ **پاسخ:** گزینه «۴» در خصوص الف، باید گفت: کافی است جای حالت پذیرش و حالت رد جابه‌جا شود، در ماشین تورینگی که برای تصمیم‌پذیری درباره زبان L طراحی می‌شود تا ماشین تورینگ زبان $\overline{L}$ را بپذیرد. در خصوص عبارت ب نیز طبق تئوری Church توان هر دو ماشین یکسان است. در خصوص عبارت (ج) چون $\{M$ روی ورودی w متوقف می‌شود $|\langle M,w\rangle\} = HALT_{TM}$ شناسایی پذیر (Recognizable) بوده ولی تصمیم‌پذیر (Decidable) نیست؛ لذا مکمل آن نباید شناسایی‌پذیر یا تشخیص‌پذیر باشد.

در خصوص عبارت د، مثال نقض A_{TM} وجود دارد که به‌طور بازگشتی شمارا است (RE) ولی تصمیم‌پذیر نمی‌باشد.

✍ **مثال ۱۷: آیا مجموعه زبان‌های به‌طور بازگشتی شمارا (RE) قابل شمارش هستند؟ چرا؟**

☑ **پاسخ:** بله؛ باید ماشین تورینگی وجود داشته باشد که زبان را شناسایی کند و از آن‌جا که مجموعه ماشین‌های تورینگ قابل شمارش هستند لذا مجموعه زبان‌های RE نیز قابل شمارش یا Countable می‌باشند.

✍ **مثال ۱۸: آیا مساله مقابل تصمیم‌پذیر است؟ «اگر M یک DFA دلخواه باشد آیا $L(M)$ نامتناهی است؟»**

☑ **پاسخ:** تصمیم‌پذیر است. یک DFA دلخواه، ماشین متناهی است و می‌توان که با بررسی کدینگ ماشین و مشاهده حلقه‌هایی که زبان را نامتناهی می‌نمایند به نامتناهی بودن زبان $L(M)$ دست یافت. لذا این مساله تصمیم‌پذیر است.

تمرین: با توجه به مساله «SUBSET – SUM» مطرح شده در فصل، مساله SUB-SET SUM2 بدین ترتیب تعریف می‌شود که: عدد دلخواه t لیست $(b_1,...,b_m)$ در مبنای 2 نوشته که در آن $b_i < b_j$ است، زمانی که اگر $i < j$ باشد؛ به صورت $2^{b_1} + 2^{b_2} + ... + 2^{b_m}$ نوشته می‌شود. مثلاً 6 به صورت (10 و 1) یعنی $6 = 2^1 + 2^2$ نوشته می‌شود: ثابت کنید SUB-SET SUM2 در کلاس NP-Complete است؟

✍ **مثال ۱۹: ثابت کنید برای هر زبان L متعلق به کلاس NP، یک ثابت c وجود دارد؛ به طوری که زبان L در رتبه زمانی $2^{O(n^c)}$ تصمیم‌پذیر خواهد بود.**

☑ **پاسخ:** فرض $L \in NP$ است. طبق Cook – Levin یک کاهش از L به مساله 3SAT وجود دارد که در زمان $O(n^c)$ انجام خواهد شد. ماشین تورینگ زیر را در نظر بگیرید:

به ازای ورودی x، ابتدا x به یک نمونه ϕ از 3SAT کاهش می‌یابد؛ تصمیم‌گیری روی ϕ با زمان $2^{O(n)}$ صورت می‌پذیرد. از طرفی زمان کاهش‌پذیری $O(n^c)$، سایز ϕ حداکثر $O(n^c)$ و در نتیجه زمان اجرای تصمیم‌گیرنده برای L عبارت است از $O(n^c) + 2^{O(n^c)} = 2^{O(n^c)}$. برای اثبات تصمیم‌گیری درباره مساله 3SAT، فرمول 3CNF دارای k متغیر را تحت عنوان ϕ در نظر بگیرید. در کل 2^k تخصیص به ϕ وجود دارد که صدق پذیری آن‌ها موجب Accept یا Reject می‌گردد. لذا زمان اجرای این الگوریتم $O(n \times 2^k) = 2^{O(n)}$ خواهد بود؛ که n همان طول ϕ است.