بسم الله الرحمن الرحیم

مقدمه‌ای بر نظریه زبانها و ماشینها

پیتر لینز

دکتر مهدی صادق‌زاده

فهرست مطالب

۱ مقدمه‌ای بر نظریه محاسبات

رشته کامپیوتر یک زمینه کاربردی است. کسانی که در این رشته کار می‌کنند اغلب مسائل علمی سودمند را بر مسائل نظری ترجیح می‌دهند. این امر واقعاً در مورد دانشجویان رشته کامپیوتر که علاقه‌مند به کار بر روی مسائل مشکل در دنیای واقعی هستند صدق می‌کند. راه حل‌های خوب نظری فقط زمانی مورد علاقه دانشجویان این رشته قرار می‌گیرند که در پیدا کردن راه حل خوب کمک کنند. این نگرش درست است، زیرا بدون کاربرد، علاقه مندان کمی به کامپیوتر وجود خواهند داشت. اما با این جهت گیری کاربردی، یک سوال مطرح می‌شود "چرا به مطالعه تئوری می‌پردازیم؟"

اولین پاسخ این است که نظریه، مفاهیم و اصولی را مطرح می‌کند که در درک ماهیت عمومی این رشته به ما کمک می‌کند. عرصه رشته کامپیوتر شامل دامنه وسیعی از موضوعات خاص از طراحی ماشین تا برنامه سازی است. کاربرد کامپیوتر در دنیای واقعی وابسته به جزئیات زیادی است که برای کاربرد موفقیت آمیز باید آموخته شود. این مورد، رشته کامپیوتر را به یک زمینه بسیار وسیع و متنوع تبدیل کرده است. اما بر خلاف این گوناگونی، مفاهیم و اصول مشترک وجود دارند. برای مطالعه در این خصوص، ما مدلهای مجردی از کامپیوتر و محاسبه را ساختیم. این مدلها خواص مشترکی در سخت افزارها و نرم افزارها و مسائل مهمی که در ساختارهای پیچیده‌ای که در هنگام کار با آنها برخورد می‌کنیم، دارا می‌باشند. همین گونه با این مدل‌ها به سادگی و بی‌درنگ با دنیای واقعی ارتباط برقرار می‌کنند، با مطالعه آنها بینشی که به ما می‌دهند، باعث ایجاد پیشرفت واضحی در این زمینه می‌شود. این روش مختص رشته کامپیوتر نبوده و در سایر رشته‌های علمی نیز مدل سازی مرسوم بوده و سودمندی یک رشته اغلب به یک ساختار ساده بستگی دارد. اکنون برتری با قوانین و نظریه‌هاست.

دومین پیشنهاد و پاسخ روشن این است که ایده‌هایی که بیان و مطرح می‌کنیم کاربردهای مهمی دارند. عرصه‌های طراحی دیجیتال، برنامه نویسی و کامپایلرها مثال‌های ساده ولی بسیار مختلف هستند. اصولی که ما در این جا مطالعه میکنیم مثل نخی بیشتر مفاهیم کامپیوتر از سیستم عامل تا شناسایی الگوها را به هم پیوند می‌زنند.

سومین پاسخی که ما امیدواریم خواننده را راضی کند، این است که این موضوع سرگرم کننده و مفرح می‌باشد. این موضوع مثل مسائلی که در هنگام خواب، انسان را به خواب می‌برند به پرس و جو می‌پردازد. این مسئله حل کردن به جای خود بسیار ضروری است.

در این کتاب ما به مدلهایی نظر داریم که شکل درونی کامپیوترها را نشان می‌دهند و کاربردشان را به نمایش می‌گذارند. برای مدل کزدن سخت افزار یک کامپیوتر ما تصویری از یک **ماشین** (جمع آن، **ماشین‌ها**) را معرفی می‌کنیم.

یک ماشین، ساختاری است که تمام ویژگی‌های یک کامپیوتر دیجیتال را داراست. ماشین ورودی را دریافت می‌کند، خروجی تولید می‌کند، ممکن است انواع مختلفی از ذخیره سازی را داشته باشد، و می‌تواند در مورد انتقال و تبدیل ورودی به خروجی تصمیم گیری نماید. یک **زبان صوری** انتزاعی است از خصیصه‌های عمومی یک زبان برنامه سازی. یک زبان صوری، مجموعه‌ای است از نشاندن علائم و بعضی قوانین پیکربندی که بوسیله آن‌ها علائم ترکیب شده و به صورت موجودیت‌هایی به نام جمله در می‌آیند. یک زبان صوری، مجموعه تمام رشته‌هایی است که قوانین شکل دهی را تایید می‌کنند. گرچه بسیاری از زبان‌های صوری که ما در اینجا مطالعه می‌کنیم ساده تر از زبان‌های برنامه سازی هستند، اما دارای ویژگی‌های یکسانی هستند. ما می‌توانیم مطالب زیادی را در مورد زبان‌های برنامه سازی با یادگیری زبان‌های صوری فرا بگیریم. سرانجام ، ما می‌توانیم با بدست آوردن تعریف دقیقی از اصطلاح **الگوریتم** و یادگیری انواع مختلفی از مسائل که مناسب برای حل توسط وسایل مکانیزه هستند (یا نیستند)، مفهوم محاسبه مکانیزه را شکل دهی و پیکربندی کنیم. با پیشرفت مطالعه ما تمامی اتصالات میان این جداسازی‌ها با مشخص کردن‌ها و بررسی نتایج حاصل از آن‌ها را نشان خواهیم داد.

در فصل اول، ما این پهنه عریض روشن کار را شاهد هستیم. در بخش ۱-۱ ما بازنگری خواهیم داشت بر نظریه‌های مهم محاسبات که نیازمند آنها خواهیم بود. مادامیکه شهود گرایی راهنمای ما در جستجوی کردن ایده‌ها باشد سرانجام ما به مباحثات سخت کشیده خواهد شد. این شامل مقداری تکنیک محاسبات خواهد بود اگر چه این نیاز بسیار نخواهد بود. خواننده به یک فهم خوب و معقولانه از اصطلاحات علمی و نتایج ابتدایی از نظریه مجموعه‌ها، توابع و روابط احتیاج خواهد داشت. درخت‌ها و نمودارها به طور مکرر به کار خواهند رفت و چون احتیاج کمی به تعریف یک برچسب وجود داشت، گراف طراحی شد. شاید بزرگترین و شدیدترین نیاز، توانایی در استفاده و یافتن مدارک و مستندات و فهمیدن این که چه برهان محاسباتی مناسب است می‌باشد و ما فرض میکنیم که خواننده این ذهنیت لازم را داراست. بخش ۱-۱ شامل نگرش دوباره‌ای بر بعضی از نتایج و قوانین مورد استفاده و جاگذاری علائم مرسوم در بحث‌های وابسته می‌باشد.

در بخش ۱-۲ ما ابتدا نگاهی خواهیم داشت بر مفهوم اصلی زبان‌های گرامری و آتاماتا. این مفاهیم به شکل واضح در سرتاسر کتاب وجود دارند. در بخش ۱-۳ ما استفاده ساده‌ای از این نظرات عمومی برای نتیجه گیری نشان خواهیم داد که این مفهوم استفاده بسیار گسترده‌ای در رشته کامپیوتر دارد.

بحث موجود در این دو بخش به شکل محسوسی دقیق و موشکافانه خواهد بود. سپس، ازاین ظرائف استفاده خواهیم کرد. اما برای مواقعی، نتیجه، بدست آوردن تصویر واضح و روشنی از مفاهیمی است که ارائه خواهیم داد.

۱-۱ مقدمات ریاضی و علامت گذاری مجموعه‌ها

یک **مجموعه**، دسته‌ای از اعضاء است، که به جز عضویت دارای هیچ ساختاری نمی‌باشند. برای نشان دادن این که x یک عضو از مجموعه S است، می‌نویسیم $x \in S$. برای نشان دادن این که x عضو مجموعه S نیست، می‌نویسیم $x \notin S$. یک مجموعه با محصور کردن اعضایش در نشان ابروهای پیچیده (آکولاد) نشان داده می‌شود. برای مثال، مجموعه اعداد صحیح ۰و۱و۲ این گونه نمایش داده می‌شود:

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

نقطه چین زمانی استفاده می‌شود که معنای آن واضح و روشن باشد. مثلا $\{a,b,\ldots,z\}$ برای همه حروف الفبای کوچک انگلیسی، $\{2,4,6,\ldots\}$ نشان دهنده همه اعداد صحیح زوج می‌باشد. در زمانی که توضیح احتیاج داریم، ما از اشارات نزدیک‌تر و صریح‌تری استفاده می‌کنیم. دراینصورت می‌نویسیم :

$$S = \{i : i > 0, i \quad is \quad even\} \qquad (۱-۱)$$

برای مثال اخیر، این گونه می‌خوانیم: " S مجموعه‌ای است از همه i هایی که i بزرگتر از صفر بوده و i زوج است،" در ضمن روشن است که i یک عدد صحیح است.

عملیات معمول بر روی مجموعه‌ها **اجتماع** $(\bigcup)$، **اشتراک** $(\bigcap)$، و **تفاضل** $(-)$ است، که بصورت زیر تعریف می‌شوند :

$$S_1 \bigcup S_2 = \{x : x \in S_1 \quad or \quad x \in S_2\},$$

$$S_1 \bigcap S_2 = \{x : x \in S_1 \quad and \quad x \in S_2\},$$

$$S_1 - S_2 = \{x : x \in S_1 \quad and \quad x \notin S_2\}.$$

عمل اساسی دیگر **متمم** است. متمم مجموعه S با $\overline{S}$ نمایش داده می‌شود. $\overline{S}$ شامل تمامی عناصری است که در S نیست. برای یافتن این مفهوم، نیازمند شناخت **مجموعه جهانی** U شامل تمامی عناصر ممکن هستیم. اگر U مشخص باشد، آنگاه :

$$\overline{S} = \{x : x \in U, \quad x \notin S\}$$

مجموعه‌ای که هیچ عضوی ندارد، یا نامیده می‌شود و به شکل $\varnothing$ نمایش داده می‌شود. از تعریف مجموعه روشن است که :

$$S \bigcup \varnothing = S - \varnothing = S,$$
$$S \bigcap \varnothing = \varnothing,$$
$$\overline{\varnothing} = U,$$
$$\overline{\overline{S}} = S.$$

قوانین مفید زیر، به **قوانین دمورگان** مشهورند:

$$\overline{S_1 \bigcup S_2} = \overline{S_1} \bigcap \overline{S_2}, \qquad (۱-۲)$$

$$\overline{S_1 \bigcap S_2} = \overline{S_1} \bigcup \overline{S_2}, \qquad (۱-۳)$$

که در موارد زیادی استفاده می‌شوند.

یک مجموعه S_1 **زیر مجموعه** S گفته می‌شود، اگر تمامی عناصر S_1 همگی در S موجود باشند. ما این گونه می‌نویسیم :

$$S_1 \subseteq S.$$

اگر $S_1 \subseteq S$ اما S دارای عنصری بود که در S_1 وجود نداشت، آن گاه گوییم S_1 یک **زیر مجموعه محض** از S است و می‌نویسیم :

$$S_1 \subset S.$$

اگر S_1 و S_2 دارای هیچ عضو مشترکی نباشند، بطوریکه $S_1 \bigcap S_2 = \varnothing$، آنها را دو **مجموعه جدا ازهم** گویند.

یک مجموعه را متناهی گویند اگر شامل تعداد محدودی از عناصر باشد، در غیراینصورت نامتناهی نامیده می‌شود. اندازه یک مجموعه متناهی، تعداد اعضای موجود در آن می‌باشد و بصورت $|S|$ نمایش داده می‌شود.

یک مجموعه دارای تعداد زیادی زیرمجموعه است. مجموعه همه زیر مجموعه‌های مجموعه S، **مجموعه توانی** S نامیده می‌شود و با 2^S نمایش داده می‌شود.

مثال ۱-۱ : اگر مجموعه S ، $\{a,b,c\}$ باشد، آنگاه مجموعه توانی آن اینگونه است :

$$2^S = \{\varnothing, \{a\}, \{b\}, \{c\}, \{a,b\}, \{a,c\}, \{b,c\}, \{a,b,c\}\}.$$

در اینجا $|S| = 3$ و $|2^S| = 8$. این از یک نتیجه کلی حاصل می‌شود، اگر S مجموعه‌ای متناهی باشد، دراینصورت : $|2^S| = 2^{|S|}$

در بسیاری از مثال‌ها، عناصر یک مجموعه دنباله‌ای از عناصر چیده شده و مرتب شده مجموعه‌های دیگر هستند. به این گونه مجموعه‌ها، **حاصلضرب دکارتی** گویند. برای حاصلضرب دکارتی دو مجموعه، که خود مجموعه‌ای است اززوجهای مرتب، می‌نویسیم :

$$S = S_1 \times S_2 = \{(x,y) : x \in S_1, y \in S_2\}$$

مثال ۱-۲: فرض میکنیم $S_1 = \{2,4\}$ و $S_2 = \{2,3,5,6\}$ آنگاه :

$$S_1 \times S_2 = \{(2,2),(2,3),(2,5),(2,6),(4,2),(4,3),(4,5),(4,6)\}.$$

توجه کنید که ترتیب قرار گرفتن اعضای هر زوج که نوشته می‌شوند، مهم است، زوج $(2,4)$ در $S_1 \times S_2$ وجود ندارد.

علامت گذاری برای تعمیم حاصلضرب دکارتی به بیش از دو مجموعه در حالت کلی زیر است:

$$S_1 \times S_2 \times \cdots \times S_n = \{(x_1, x_2, \ldots, x_n) : x_i \in S_i\}.$$

توابع و روابط

یک **تابع**، قانون یا ضابطه‌ای است که عناصر یک مجموعه را بصورت یکتا به عناصر مجموعه دیگری مربوط می‌سازد. اگر f نشان دهنده یک تابع باشد، مجموعه اول را **دامنه** و مجموعه دوم را **برد** f گویند. می‌نویسیم :

$$f : S_1 \to S_2$$

که بیانگر این است که دامنه f یک زیر مجموعه S_1 و برد آن یک زیر مجموعه از S_2 می‌باشد. اگر دامنه تابع f همه S_1 باشد، گوییم f یک **تابع کلی** روی S_1 است، درغیر اینصورت f را **تابع جزئی** گویند.

در بسیاری از کاربردها، دامنه و برد توابع، مجموعه اعداد صحیح مثبت است. بعلاوه، ما اغلب علاقمند به بررسی رفتار این توابع به ازای مقادیر خیلی بزرگ آرگومانهایشان هستیم. در چنین مواردی، درکی از نرخ رشد ضروری است، و مرتبه مرسوم از علامت دامنه قابل استفاده است. فرض کنید $f(n)$ و $g(n)$ توابعی باشند که دامنه آنها زیر مجموعه‌ای از اعداد صحیح مثبت است. اگر یک ثابت مثبت c وجود داشته باشد طوریکه برای همه n ها داشته باشیم :

$$f(n) \le cg(n),$$

گوییم f از **مرتبه حداکثر** g است، و می‌نویسیم :

$$f(n) = O(g(n)).$$

اگر

$$f(n) \ge cg(n),$$

دراینصورت f از **مرتبه حداقل** g است، و از نماد زیر استفاده میکنیم:

$$f(n) = \Omega(g(n)).$$

درنهایت، اگر ثوابت c_1 و c_2 وجود داشته باشند طوریکه

$$c_1|g(n)| \le |f(n)| \le c_2|g(n)|,$$

f و g از **مرتبه دامنه یکسانی** هستند، و می‌نویسیم:

$$f(n) = \Theta(g(n)).$$

در این نماد مرتبه دامنه، ما از ضرایب ثابت و جملات از مرتبه پایین تر صرفنظر می‌کنیم، که با افزایش n قابل چشم پوشی می‌باشند.

مثال ۱-۳-۱: فرض کنید

$$f(n) = 2n^2 + 3n,$$
$$g(n) = n^3,$$
$$h(n) = 10n^2 + 100.$$

در اینصورت

$$f(n) = O(g(n)),$$
$$g(n) = \Omega(h(n)),$$
$$f(n) = \Theta(h(n)).$$

در نماد مرتبه دامنه، نشانه = نباید به معنای تساوی تفسیر شود و عبارات مرتبه دامنه، نباید شبیه عبارات معمولی نگریسته شوند. دستکاری‌هایی مانند

$$O(n) + O(n) = 2O(n)$$

قابل قبول نیستند، و می توانند به نتایج نادرست منجر شوند. هنوز، اگر بدرستی استفاده شود، آرگومان‌های مرتبه دامنه می‌توانند موثر باشند، چنانچه در فصول بعدی در تحلیل الگوریتم‌ها خواهیم دید.

یک تابع را می‌توان بصورت مجموعه ای از زوج‌ها نمایش داد :

$$\{(x_1, y_1), (x_2, y_2), \cdots\},$$

که x_i یک عضو در دامنه تابع و y_i مقدار متناظر با آن در برد آن می‌باشد. برای آنکه یک مجموعه تابع باشد، هر x می‌تواند حداکثر یکبار به عنوان اولین مولفه یک زوج ظاهر شود. اگر این گونه نباشد، مجموعه یک **رابطه** نامیده می‌شود. رابطه بسیار کلی تر از تابع می‌باشد. در یک تابع، هر عضو دامنه فقط دارای یک عنصر متناظر در برد می‌باشد، در رابطه ممکن است چندین عنصر متناظر در برد وجود داشته باشد. یک نوع رابطه، رابطه **هم ارزی** می‌باشد که مشتق از مفهوم تساوی (برابری) است. برای نمایش این که زوج (x, y) رابطه هم ارزی دارند، می‌نویسیم :

$$x \equiv y.$$

رابطه هم ارزی با $\equiv$ نمایش داده می‌شود، اگر سه شرط زیر برقرار باشد :

قانون بازتابی

برای همه xها، $x \equiv x$

قانون تقارنی

اگر $x \equiv y$، آنگاه $y \equiv x$

قانون تعدی

اگر $x \equiv y$ و $y \equiv z$، آنگاه $x \equiv z$

مثال ۱-۴ : رابطه روی مجموعه اعداد غیر منفی صحیح که بصورت زیر تعریف می‌شود را در نظر بگیرید:

$$x \equiv y$$

اگر و فقط اگر $x \bmod 3 = y \bmod 3$

در اینصورت $5 \equiv 2$، $0 \equiv 12$ و $36 \equiv 0$. واضح است که این یک رابطه هم ارزی است وسه قانون بازتابی، تقارنی، و تعدی را ارضا می‌نماید.

گراف‌ها و درخت‌ها

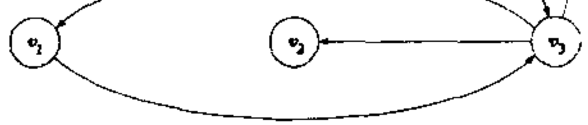
یک گراف، ساختاری است دارای دو مجموعه متناهی، مجموعه $V = \{v_1, v_2, \ldots, v_n\}$ از **رئوس** و مجموعه $E = \{e_1, e_2, \ldots, e_m\}$ از **یال‌ها**. هر یال زوجی از رئوس V است، برای نمونه:

$$e_i = (v_j, v_k)$$

یالی از v_j به v_k است. گوییم که یال e_i، یک یال خروجی از v_j و یال ورودی به v_k است. چنین ساختاری را گراف جهت‌دار گوییم، زیرا ما به هر یال جهتی را (از v_j به v_k) نسبت داده‌ایم. گراف‌ها می‌توانند دارای برچسب باشند. برچسب می‌تواند نام و یا اطلاعات دیگری در بخش‌های دیگر گراف باشد. هر دوی یالها و رئوس می‌توانند دارای برچسب باشند.

گراف‌ها برای سهولت با نمودار نشان داده می‌شوند. به این صورت که رئوس بصورت دایره و یال‌ها بصورت پیکان‌هایی رئوس را به هم متصل می‌کنند. گراف با رئوس $\{v_1, v_2, v_3\}$ و یال‌های $\{(v_1, v_3), (v_3, v_1), (v_3, v_2), (v_3, v_3)\}$ در شکل ۱-۱ نشان داده شده است.

به دنباله‌ای از یال‌های $(v_i, v_j), (v_j, v_k), \ldots, (v_m, v_n)$ **راهی** از i به n گویند. طول یک راه مساوی با تمام یال‌هایی است که از راس مبدا خارج و به راس **مقصد** می‌رسند. راهی که هیچ یالی در آن تکرار نشود، یک **مسیر** نامیده می‌شود. مسیر را **ساده** گویند هرگاه هیچ راسی در آن تکرار نشود. یک راه از v_i به خودش، بدون هیچ یال تکراری، یک **چرخه** با **پایه** v_i نامیده می‌شود. در یک چرخه با پایه

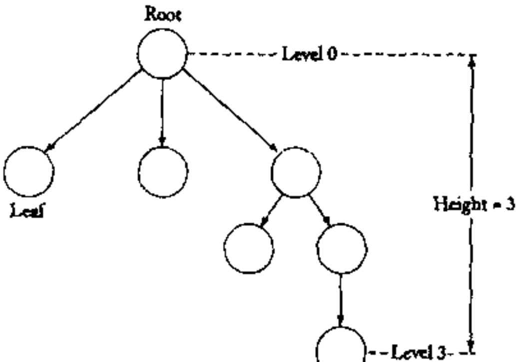


شکل ۱-۱

v_i اگر بجز پایه هیچ راسی تکرار نشود آنگاه چرخه، ساده نامیده می‌شود. در شکل ۱-۱، $(v_1, v_3), (v_3, v_2)$ مسیر ساده‌ای از v_1 به v_2 است. دنباله یال‌های $(v_1, v_3), (v_3, v_3), (v_3, v_1)$ یک چرخه غیر ساده است. اگر یال‌های گراف دارای برچسب باشند، آنگاه می‌توان در مورد برچسب یک راه صحبت کرد. این برچسب شامل یک رشته از برچسب یال‌هایی است که در طول مسیر پیمایش می‌شوند. سرانجام، یالی از یک راس به خودش را یک **طوقه** گویند. در شکل ۱-۱ طوقه‌ای روی راس v_3 وجود دارد.

در چند فرصت، به الگوریتمی برای یافتن همه مسیرهای ساده بین دو راس مفروض (یا همه چرخه‌های ساده با پایه یک راس) مراجعه خواهیم کرد. اگر خودمان را چندان درگیر کارآیی نکنیم، می‌توانیم به راحتی از روشی که گفته خواهد شد استفاده کنیم. با شروع از یک راس مفروض مثل v_i، همه یال‌های خروجی ازاین نقطه را لیست می‌کنیم مانند $(v_i, v_k), (v_i, v_l), \ldots$. ما همه مسیرهایی با طول یک که از v_i آغاز می‌شوند را داریم. برای تمامی رئوس $v_k, v_l, \ldots$ که به آنها رسیده‌ایم، تمامی یال‌های خروجی را مادامیکه در مسیر به هیچ یک از رئوسی که قبلاً در مسیر استفاده شده‌اند نرسیده باشیم، لیست می‌کنیم. بعد از انجام این عمل ما تمامی مسیرهای ساده‌ای را که با مبدا v_i هستند، خواهیم داشت. این عمل را تا جایی که امکان داشته باشد، ادامه می‌دهیم. از آنجایی که فقط تعداد محدودی از رئوس وجود دارد، تمامی مسیرهای ساده شروع شده از v_i را لیست کرده‌ایم. ازبین مسیرهای ساده بدست آمده، آنهایی را انتخاب می‌کنیم که با راس مطلوب خاتمه یافته‌اند.

درخت‌ها نوع خاصی از گراف هستند. درخت، گراف جهت داری است که چرخه ندارد، و یک گره خاص دارد که به آن **ریشه** گویند، طوریکه فقط یک مسیر از ریشه به سایر رئوس وجود دارد. روشن است که هیچ گونه یالی به ریشه وارد نمی‌شود، و رئوسی هستند که هیچ یالی از آنها خارج نمی‌شود. این رئوس **برگ‌های** درخت نامیده می‌شوند. اگر یک یال از v_i به v_j وجود داشته باشد، آنگاه v_i را **والد** v_j نامند، و v_j **فرزند** v_i خواهد بود. **سطح** هر راس، برابر با تعداد یال‌هایی است که در مسیر موجود از ریشه تا آن راس وجود دارند. **ارتفاع درخت**، بزرگترین شماره سطح همه رئوس است. این اصطلاحات همگی در شکل ۱-۲ مشخص شده‌اند.



شکل ۱-۲

اکنون ما می‌خواهیم ارتباط دقیقی میان گره‌ها و سطوح مختلف برقرار کنیم. در مراحل دیگر، ما در مورد **درخت‌های مرتب**، بحث خواهیم کرد.

مطالب زیادی در مورد گراف‌ها و درخت‌ها در کتاب‌هایی با موضوعات ریاضیات گسسته می‌توان یافت.

روش‌های اثبات

نیاز مهمی که برای خواندن مطالب این متن وجود دارد، استفاده و پیروی از مدارک و مستندات است. در اثبات‌های ریاضی ما بسیاری از قوانین مرسوم کاهشی و بسیاری از قوانین ساده که بصورت قدم به قدم می‌باشند را به کار می‌بریم. دو روش اثبات بسیار معمول وجود دارد که به شرح مختصر آنها می‌پردازیم. آنها **اثبات بوسیله استقرا** و **اثبات بوسیله برهان خلف** می‌باشند.

استقرا، روشی است که در آن درستی تعدادی از جملات، از درستی چند نمونه خاص استنتاج می‌شود. فرض کنید که ما زنجیره‌ای از جملات $P_1, P_2, \ldots$ را داریم که می‌خواهیم ثابت کنیم آنها درست می‌باشند. بعلاوه فرض کنید که شرایط زیر برقرار است :

۱- برای برخی $k \geq 1$، ما می‌دانیم که $P_1, P_2, \ldots, P_k$ درست هستند.

۲- مسئله طوری است که برای هر $n \geq k$، درستی $P_1, P_2, \ldots, P_n$ درستی P_{n+1} را ایجاب می‌نماید.

ما می‌توانیم از استقرا برای نمایش اینکه هر جمله در زنجیره درست است، استفاده نماییم.

در یک اثبات بوسیله استقرا، ما بصورت زیر بیان می‌کنیم: از شرط ۱ می‌دانیم که k جمله درست است. سپس شرط ۲ به ما می‌گوید که P_{k+1} نیز باید درست باشد. ولی اکنون ما می‌دانیم که $k+1$ جمله اول درست هستند، بنابراین می‌توانیم شرط ۲ را بکار ببریم تا ادعا کنیم که P_{k+2} باید درست باشد، و مانند آن. ما صریحاً به این آرگومان ادامه نمی‌دهیم،زیرا الگو واضح است. زنجیره استدلال می‌تواند به هر جمله‌ای توسعه یابد. بنابراین، هر جمله درست است.

جملات شروع $P_1, P_2, \ldots, P_k$ **پایه** استقرا نامیده می‌شوند. گامی که P_n را با P_{n+1} مرتبط می‌سازد، **گام استقرا** نامیده می شود. گام استقرا عموماً بوسیله **فرض استقرا** یعنی فرض درست بودن $P_1, P_2, \ldots, P_n$، ساده‌تر می‌شود. در یک استدلال استقرائی صوری، ما هر سه بخش را صریحاً نشان می‌دهیم.

مثال ۱-۵: یک درخت دودویی، درختی است که هیچ والدی نمی‌تواند بیش از دو فرزند داشته باشد. ثابت کنید که یک درخت دودویی با ارتفاع n دارای حداکثر 2^n برگ است.

اثبات : اگر ما بیشترین تعداد برگ‌های یک درخت دودویی با ارتفاع n را بوسیله $l(n)$ تعریف کنیم، می‌خواهیم نشان دهیم که :

$$l(n) \leq 2^n.$$

پایه : روشن است که $l(0) = 1 = 2^0$ زیرا درخت با ارتفاع ۰ دارای گره‌ای بـه جـز ریـشه نیـست، یعنـی حداکثر یک برگ دارد.

فرض استقرا : برای n ، $i = 0,1,\ldots,n$ ، $l(i) \le 2^i$

گام استقرا : برای بدست آوردن یک درخت دودویی با ارتفاع $n+1$ از درختـی دودویـی بـا ارتفـاع n، می‌توانیم حداکثر دو برگ را به هر برگ در درخت قبلی اضافه نماییم، بنابراین

$$l(n+1) = 2l(n)$$

حالا با استفاده از فرض استقرا خواهیم داشت :

$$l(n+1) \le 2 \times 2^n = 2^{n+1}$$

بنابراین، ادعای ما برای $n+1$ نیز درست است. ازآنجایی که n هر عددی می‌تواند باشد، این جمله بایـد برای همه n ها درست باشد. ■

در اینجا ما نشانه ■ معرفی می‌کنیم که در این کتاب بمنظور نمایش انتهای اثبات استفاده شده است.

مثال ۱-۶ : نشان دهید که

$$S_n = \sum_{i=0}^{n} i = \frac{n(n+1)}{2}. \qquad (۱-۴)$$

ابتدا میدانیم که :

$$S_{n+1} = S_n + n + 1.$$

می‌دانیم که فرض استقرا (۱-۴) برای هر S_n برقرار است، در اینصورت :

$$S_{n+1} = \frac{n(n+1)}{2} + n + 1$$
$$= \frac{(n+2)(n+1)}{2}.$$

بنابراین (۱-۴) برای S_{n+1} نیز صحیح است و ما مرحله استقرا را گذرانـدیم. از آنجـایی کـه (۱-۴) بـرای $n = 1$ درست است، ما یک پایه و رابطه (۱-۴) اثبات شده برای همه n ها بوسیله استقرا داریم.

در این مثال آخر، شناسایی پایه، فرض استقرا، و مرحله استقرا کمتر صوری شده‌اند، ولی آنها وجود دارند و الزامی نیز هستند. برای اینکه نتیجه بحث بسیار صوری نباشد، ما عموماً سبک مثـال دوم را تـرجیح می‌دهیم. به هر حال، اگر شما در ادامه یا ساخت یک اثبات مشکلی داشتید، برای صراحت بیشتر به عقب به مثال ۱-۵ برگردید.

استدلال بوسیله استقرا می‌تواند مشکل باشد، و بـه توجـه داشـتن بـه ارتبـاط نزدیکـی بـین اسـتقرا و بازگشت پذیری در برنامه نویسی کمک می‌نماید. برای مثال، تعریف بازگشتی از یک تابع $f(n)$، که n

هر عدد صحیح مثبت است، اغلب دارای دو بخش است. یک بخش شامل تعریف $f(n+1)$ بر حسب $f(n), f(n-1), \ldots, f(1)$ می‌باشد که متناظر با مرحله استقرا است. بخش دوم "فرار" از بازگشتی است، که بوسیله تعریف $f(1), f(2), \ldots, f(k)$ به صورت غیر بازگشتی انجام می‌پذیرد، ومتناظر با پایه استقرا است. مانند استقراء، بازگشت پذیری نیز به ما اجازه نتیجه گیری درباره همه نمونه‌های مسئله را با تعدادی مقادیر آغازین و استفاده از طبیعت بازگشتی مسئله می‌دهد.

اثبات بوسیله برهان خلف، روش قدرتمند دیگری است و هنگامی بکار می‌رود که هر چیز دیگری با شکست روبرو می‌شود. فرض کنید ما می‌خواهیم ثابت کنیم که جمله P درست است. برای لحظه‌ای فرض می‌کنیم که P نادرست باشد و می‌بینیم که این فرض به کجا منجر میشود. اگر ما به نتیجه‌ای برسیم که می‌دانیم نادرست است، ما می‌توانیم فرض آغازی را زیر سوال ببریم و نتیجه بگیریم که P باید درست باشد. در ادامه مثال کلاسیک و مربوطه می‌آید.

مثال ۱-۷: یک عدد گویا عددی است که می‌تواند بصورت حاصل تقسیم دو عدد صحیح m و n نوشته شود طوریکه m و n عامل مشترکی ندارند. عدد حقیقی که گویا نباشد، گنگ نامیده می‌شود. نشان دهید که $\sqrt{2}$ گویا نیست.

در همه اثبات‌ها بوسیله تناقض، ما نقیض آنچه را که می خواهیم نشان دهیم، فرض می‌کنیم. در اینجا فرض می‌کنیم که $\sqrt{2}$ یک عدد گویا باشد، طوریکه می‌توانیم آن را بصورت زیر بنویسیم :

$$\sqrt{2} = \frac{n}{m}, \tag{۱-۵}$$

جاییکه m و n اعداد صحیح بدون عامل مشترک باشند. با مرتب کردن دوباره (۱-۵) داریم :

$$2m^2 = n^2.$$

بنابراین n^2 باید زوج باشد. این امر، زوج بودن n را ایجاب می‌کند، بنابراین می‌توانیم بنویسیم $n = 2k$ یا

$$2m^2 = 4k^2,$$

و

$$m^2 = 2k^2.$$

بنابراین m زوج است. اما این فرض اولیه ما که m و n عامل مشترکی ندارند، نقض می‌کند. بنابراین، m و n در (۱-۵) وجود ندارند و $\sqrt{2}$ یک عدد گویا نیست.

در این مثال از برهان خلف استفاده کردیم. با ساخت یک فرض به یک تناقضی از فرض یا حقایق معلوم می‌رسیم. اگر همه مراحل در آرگومان ما از نظر منطقی درست باشند، ما باید نتیجه بگیریم که فرض اولیه ما نادرست بوده است.

☐ تمرین‌ها

۱- با استفاده از استقرا بر روی اندازه S نشان دهید که اگر S مجموعه متناهی باشد آنگاه $\left|2^S\right| = 2^{|S|}$.

۲- نشان دهید که اگر S_1 و S_2 مجموعه‌های متناهی باشند بطوریکه $n = |S_1|$ و $m = |S_2|$ باشد، آنگاه

$$|S_1 \cup S_2| \le n + m.$$

۳- اگر S_1 و S_2 مجموعه‌های متناهی باشند، نشان دهید که $|S_1 \times S_2| = |S_1||S_2|$.

۴- رابطه بین دو مجموعه بصورت $S_1 \equiv S_2$ اگر و فقط اگر $|S_1| = |S_2|$ تعریف شده است. نشان دهید که این رابطه، یک رابطه هم ارزی است.

۵- قوانین دمورگان، معادلات (۱-۲)و (۱-۳) را ثابت کنید.

۶- گاهی اوقات مجبور می‌شویم که از علائم اجتماع و اشتراک مانند علامت جمع Σ استفاده کنیم. تعریف می‌کنیم

$$\bigcup_{p \in \{i,j,k,\dots\}} S_p = S_i \cup S_j \cup S_k \dots$$

که با نماد اشتراک چندین مجموعه مشابه است. با این نماد، قوانین کلی دمورگان بصورت زیر نوشته می‌شوند :

$$\overline{\bigcup_{p \in P} S_p} = \bigcap_{p \in P} \overline{S}_p$$

و

$$\overline{\bigcap_{p \in P} S_p} = \bigcup_{p \in P} \overline{S}_p.$$

ثابت کنید که این تساوی‌ها هنگامی که P یک مجموعه متناهی باشد، برقرار است.

۷- نشان دهید که

$$S_1 \cup S_2 = \overline{\overline{S}_1 \cap \overline{S}_2}.$$

۸- نشان دهید که $S_1 = S_2$ اگر و فقط اگر

$$(S_1 \cap \overline{S}_2) \cup (\overline{S}_1 \cap S_2) = \varnothing.$$

۹- نشان دهید که

$$S_1 \cup S_2 - (S_1 \cap \overline{S}_2) = S_2.$$

۱۰- نشان دهید که

$$S_1 \times (S_2 \cup S_3) = (S_1 \times S_2) \cup (S_1 \times S_3).$$

۱۱- نشان دهید که اگر $S_1 \subseteq S_2$، آنگاه $\overline{S}_2 \subseteq \overline{S}_1$.

۱۲- شرایطی لازم و کافی روی S_1 و S_2 بیابید طوری که

$$S_1 = (S_1 \cup S_2) - S_2 .$$

۱۳- نشان دهید که اگر $f(n) = O(g(n))$ و $g(n) = O(f(n))$، در اینصورت $f(n) = \Theta(g(n))$.

۱۴- نشان دهید که $2^n = O(3^n)$ ولی $2^n \neq \Theta(3^n)$.

۱۵- نشان دهید که مرتبه دامنه زیر برقرار است.

الف) $n^2 + 5\log n = O(n^2)$

ب) $3^n = O(n!)$

ج) $n! = O(n^n)$

۱۶- ثابت کنید که اگر $f(n) = O(g(n))$ و $g(n) = O(h(n))$، در اینصورت $f(n) = O(h(n))$.

۱۷- نشان دهید که اگر $f(n) = O(n^2)$ و $g(n) = O(n^3)$، در اینصورت

$$f(n) + g(n) = O(n^3)$$

و

$$f(n)g(n) = O(n^6).$$

۱۸- در تمرین ۱۷، آیا $g(n)/f(n) = O(n)$ درست است؟

۱۹- فرض کنید که $f(n) = 2n^2 + n$ و $g(n) = O(n^2)$. چه چیزی در جمله زیر نادرست است؟

$$f(n) = O(n^2) + O(n).$$

طوریکه

$$f(n) - g(n) = O(n^2) + O(n) - O(n^2).$$

بنابراین

$$f(n) - g(n) = O(n).$$

۲۰- تصویری از یک گراف با رئوس $\{v_1, v_2, v_3\}$ و یال‌های $\{(v_1, v_1), (v_1, v_2), (v_2, v_3), (v_2, v_1), (v_3, v_1)\}$ رسم کنید. همه چرخه‌ها با پایه v_1 را بشمارید.

۲۱- فرض کنید $G = (V, E)$ هر گراف دلخواهی باشد. ادعای زیر را ثابت کنید : اگر هر راهی بین $v_i \in V$ و $v_j \in V$ وجود داشته باشد، در اینصورت باید مسیری با طول نابیشتر از $|V| - 1$ بین این دو راس موجود باشد.

۲۲- گراف‌هایی را در نظر بگیرید که حداکثر یک یال بین هر دو راس موجود باشد. نشان دهید که تحت این شرط، یک گراف با n راس، دارای حداکثر n^2 یال خواهد بود.

۲۳- نشان دهید که

$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}.$$

۲۴- نشان دهید که

$$\sum_{i=1}^{n} \frac{1}{i^2} \le 2 - \frac{1}{n}.$$

۲۵- ثابت کنید که برای همه $n \ge 4$ نامعادله $2^n \prec n!$ برقرار است.

۲۶- نشان دهید که $\sqrt{8}$ یک عدد گویا نیست.

۲۷- نشان دهید که $2 - \sqrt{2}$ گویا نیست.

۲۸- جملات زیر را اثبات و یا عدم اثبات نمایید.

الف) مجموع یک عدد گویا و یک عدد غیر گویا، غیر گویا است.

ب) مجموع دو عدد مثبت غیر گویا، باید غیر گویا باشد.

ج) حاصلضرب یک عدد گویا و یک عدد غیر گویا، باید غیر گویا باشد.

۲۹- نشان دهید که هر عدد صحیح مثبت می تواند بصورت حاصلضرب اعداد اول بیان شود.

۳۰- ثابت کنید که مجموعه همه اعداد اول نامتناهی است. *

۳۱- یک زوج اول شامل دو عدد اول است که اختلاف آنها دو می‌باشد. زوجهای اول زیادی وجود دارند، مانند ۱۱ و ۱۳، ۱۷و۱۹، و غیره. سه تایی‌های اول، سه عدد n، $n+2$، و $n+4$ می‌باشند که همگی اول هستند. نشان دهید که تنها سه تایی‌های اول $(1,3,5)$ و $(3,5,7)$ می‌باشند.

۱-۲ سه مفهوم اساسی

سه ایده اصلی، موضوعات اصلی این کتاب هستند : **زبانها**، **گرامرها**، و **ماشین‌ها**. در دوره مورد مطالعه ما، نتایج زیادی درباره این مفاهیم و درباره ارتباطشان با یکدیگر مورد بررسی قرار می‌گیرند. ابتدا، باید معنای این واژه‌ها را درک نماییم.

زبان‌ها

ما همگی با زبانهای ملی، مانند انگلیسی و فرانسه آشناییم. هنوز هم اغلب ما احتمالاً در یافتن معنای دقیق لغت "زبان" مشکل داریم. فرهنگ لغات یک واژه را غیر صوری تعریف می‌کنند، چنانچه یک سیستم برای بیان ایده‌های مشخص، حقایق، یا مفاهیم شامل مجموعه‌ای از نمادها و قوانین برای دستکاری آنها مناسب است. اگر چه این یک ایده ظاهری در مورد آنچه که یک زبان هست به ما می‌دهد، به عنوان تعریفی جهت مطالعه زبان‌های صوری کافی نیست. ما به یک تعریف دقیق برای این واژه نیاز داریم.

ما با یک مجموعه متناهی غیرتهی Σ ازنمادها به نام **الفبا** آغاز می کنیم. از نمادهای مجزا، **رشته‌ها** را می‌سازیم، که که دنباله‌های متناهی از نمادهای الفبا می‌باشند. برای مثال، اگر الفبا $\Sigma = \{a,b\}$ باشد،

آنگاه $abab$ و $aaabbba$ رشته‌هایی روی Σ هستند. به جز موارد استثنائی، ما از حروف کوچک $a,b,c,\dots$ برای نمایش عناصر Σ و حروف $u,v,w,\dots$ برای نامگذاری رشته‌ها استفاده می‌نماییم. برای مثال، می‌نویسیم :

$$w = abaaa$$

تا نشان دهیم که رشته‌ای با نام w دارای مقدار خاص $abaaa$ می‌باشد.

اتصال دو رشته w و v رشته‌ای است که بوسیله الحاق نمادهای v به انتهای سمت راست w حاصل می‌شود، یعنی اگر

$$w = a_1 a_2 \cdots a_n$$

و

$$v = b_1 b_2 \cdots b_m,$$

آنگاه اتصال دو رشته w و v، بوسیله wv تعیین می‌شود و عبارت است از :

$$wv = a_1 a_2 \cdots a_n b_1 b_2 \cdots b_m.$$

معکوس یک رشته بوسیله نوشتن نمادها به ترتیب معکوس حاصل می‌شود، اگر w رشته نمایش داده شده در بالا باشد، آنگاه معکوس آن w^R عبارت است از :

$$w^R = a_n \cdots a_2 a_1.$$

طول رشته w، که بوسیله $|w|$ نمایش داده می‌شود، تعداد نمادها در رشته می‌باشد. ما غالبا نیاز به مراجعه به **رشته تهی** داریم، که رشته‌ای بدون هیچ نمادی است. آن را بوسیله λ نمایش می‌دهیم. روابط ساده زیر را داریم :

$$|\lambda| = 0,$$
$$\lambda w = w\lambda = w,$$

که برای هر w صادق است.

هر رشته‌ای از بخشی از حروف متوالی رشته w را **زیررشته** w گویند. اگر

$$w = vu,$$

آنگاه زیررشته‌های v و u را بترتیب **پیشوند و پسوند** w نامند. برای مثال، اگر $w = abbab$ آنگاه $\{\lambda, a, ab, abb, abba, abbab\}$ مجموعه همه پیشوندهای w است، درحالیکه bab, ab, b برخی از پسوندهای آن می‌باشند.

خصوصیات ساده رشته‌ها، مانند طول آنهاخیلی ظاهری است و احتمالاً به کمی دقت نیاز دارد. برای مثال، اگر u و v رشته باشند، آنگاه طول اتصال آنها برابر مجموع طول‌های تک تک آنها می‌باشد، یعنی :

$$|uv| = |u| + |v|. \tag{۱-۶}$$

اگرچه این ارتباط، واضح است، ولی برای درستی و اثبات آن مفید است. روش‌ها برای انجام آن در حالات پیچیده‌تر اهمیت پیدا می‌کنند.

مثال ۱-۸ : نشان دهید (۱-۶) برای هر u و v برقرار است. برای اثبات این، ما ابتدا به تعریف طول یک رشته نیاز داریم. ما چنین تعریفی را بصورت بازگشتی می‌سازیم :

$$|a| = 1,$$
$$|wa| = |w| + 1,$$

برای هر $a \in \Sigma$ و w هر رشته‌ای روی Σ است. این تعریف یک جمله صوری از فهم شهودی ما درباره طول یک رشته است : طول یک نماد تنها یک است، و طول هر رشته‌ای با اضافه کردن یک نماد به آن، یک واحد افزایش می‌یابد. با این تعریف صوری، ما آماده هستیم تا (۱-۶) را بوسیله استقرا ثابت کنیم.

از روی تعریف، (۱-۶) برای هر u با هر طول و هر v با طول ۱، برقرار است، بنابراین ما یک پایه داریم. به عنوان فرض استقرا، ما می‌دانیم که (۱-۶) برای هر u با هر طولی و هر v با طول $1,2,\ldots,n$ برقرار است. اکنون هر v با طول $n+1$ را بگیرید و آن را بصورت $v = wa$ بنویسید. آنگاه،

$$|v| = |w| + 1,$$
$$|uv| = |uwa| = |uw| + 1.$$

ولی از روی فرض استقرا (که قابل کاربرد است زیرا w دارای طول n است)، داریم :

$$|uw| = |u| + |w|,$$

طوریکه

$$|uv| = |u| + |w| + 1 = |u| + |v|.$$

بنابراین، (۱-۶) برای هر u و هر v با طول حداکثر $n+1$ برقرار است، و مرحله استقرا و آرگومان را تکمیل می‌کند. ■

اگر w یک رشته باشد، آنگاه w^n نشاندهنده رشته‌ای است که بوسیله تکرار w به تعداد n بار حاصل می‌شود. برای یک مورد خاص، ما تعریف می‌کنیم :

$$w^0 = \lambda,$$

برای هر w.

اگر Σ یک الفبا باشد، آنگاه از Σ^* برای نمایش مجموعه رشته‌هایی که از اتصال صفر یا بیشتر از نمادهای Σ حاصل می‌شود، استفاده می‌شود. مجموعه Σ^* همیشه شامل λ است. برای خارج کردن رشته تهی تعریف می‌کنیم :

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

درحالیکه Σ بفرض محدود است، Σ^* و Σ^+ همیشه نامتناهی هستند، زیرا هیچ محدودیتی روی طول رشته‌ها در این مجموعه‌ها وجود ندارد.

یک زبان بصورت خیلی عمومی به عنوان زیر مجموعه‌ای از Σ^* تعریف می‌شود. یک رشته در یک زبان L را یک **جمله** در L گویند. این تعریف کاملاً وسیع است، هر مجموعه‌ای از رشته‌های روی

یک الفبای Σ می‌تواند یک زبان در نظر گرفته شود. بعداً ما روشهایی را مطالعه می‌کنیم که توسط آنها زبانهای خاص می‌توانند تعریف و توصیف شوند، این امر ما را قادر می‌سازد تا ساختاری برای این مفهوم وسیع ارائه دهیم. برای لحظه‌ای، تفکر، ما به تعدادی مثال خاص نظر می‌کنیم.

مثال ۱-۹: فرض کنید $\Sigma = \{a,b\}$. آنگاه

$$\Sigma^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}.$$

مجموعه

$$\{a, aa, aab\}$$

یک زبان روی Σ است. بدلیل اینکه آن زبان تعداد محدودی جمله دارد، آن را یک زبان متناهی نامیم. مجموعه

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

نیز یک زبان روی Σ است. رشته‌های $aabb$ و $aaaabbbb$ در زبان L هستند، ولی رشته abb در L نیست. این زبان نامتناهی است. اکثر زبان‌های مورد علاقه نامتناهی می‌باشند.

از آنجایی که زبان‌ها مجموعه هستند، اجتماع، اشتراک، و تفاضل دو زبان بلافاصله قابل تعریف است. مکمل یک زبان با توجه به Σ^* تعریف می‌شود، یعنی مکمل زبان L عبارت است از :

$$\overline{L} = \Sigma^* - L.$$

معکوس زبان، مجموعه همه رشته‌های معکوس شده است، یعنی

$$L^R = \{w^R : w \in L\}.$$

اتصال دو زبان L_1 و L_2، مجموعه همه رشته‌هایی است که بوسیله اتصال هر عنصری از L_1 به هر عنصری از L_2 حاصل می‌شود، خصوصاً،

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

ما L^n را به عنوان زبان L که n بار به خودش متصل شده است، تعریف می‌کنیم، با موارد خاص

$$L^0 = \{\lambda\}$$

و

$$L^1 = L$$

برای هر زبان L.

درنهایت، ما **بستار ستاره‌ای** یک زبان را بصورت

$$L^* = L^0 \cup L^1 \cup L^2 \ldots$$

و **بستار مثبت** را بصورت

$$L^+ = L^1 \cup L^2 \ldots.$$

تعریف می‌کنیم.

مثال ۱-۱۰ : اگر

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

آنگاه

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

توجه کنید که n و m در بالا غیر وابسته‌اند، رشته $aabbaaabbb$ در L^2 می‌باشد.

معکوس L به سادگی بوسیله مجموعه زیر توصیف می‌شود.

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

ولی توصیف $\overline{L}$ یا L^* بدین روش به میزان قابل توجهی مشکل‌تر است. تلاش اندکی شما را به محدودیت نماد مجموعه برای مشخص کردن زبان‌های پیچیده واقف می‌گرداند.

گرامرها

برای مطالعه زبان‌ها از نظر ریاضی، ما به مکانیزمی جهت توصیف آنها نیازمندیم. زبان هر روزه غیر دقیق و مبهم است، بنابراین غالباً توصیفات غیر صوری در انگلیسی، غیر کافی می‌باشند. نماد مجموعه استفاده شده در مثال‌های ۱-۹ و ۱-۱۰ مناسب تر است، ولی محدود است. در ادامه، ما درباره چندین مکانیسم تعریف زبان می‌آموزیم که در شرایط مختلف، مفید هستند. در اینجا، ما یک مکانیزم رایج و قوی، یعنی نماد **گرامر** را معرفی می‌نماییم.

یک گرامر برای زبان انگلیسی به ما می‌گوید که آیا یک جمله خاص خوش ساخت هست یا خیر. یک قانون نمونه در گرامر انگلیسی عبارت است از: "یک جمله می‌تواند شامل یک عبارت اسمی باشد که بوسیله یک گزاره دنبال می‌شود." بصورت دقیق‌تر ما می‌توانیم بنویسیم :

$$\langle sentence \rangle \rightarrow \langle noun\_phrase \rangle \ \langle predicate \rangle,$$

با تفسیر آشکار. این امر، البته برای بحث درباره جملات واقعی کافی نیست. اکنون، ما باید تعاریفی برای ساختارهای جدیداً معرفی شده $\langle noun\_phrase \rangle$ و $\langle predicate \rangle$ مهیا نماییم. این تعاریف را بصورت زیر خواهیم داشت :

$$\langle noun\_phrase \rangle \rightarrow \langle article \rangle \ \langle noun \rangle,$$

$$\langle predicate \rangle \rightarrow \langle verb \rangle,$$

و اگر ما لغات واقعی "a" و "the" را با $\langle article \rangle$، "boy" و "dog" را با $\langle noun \rangle$، و "runs" و "walks" را با $\langle verb \rangle$ ارتباط دهیم، آنگاه گرامر به ما می‌گوید که جملات "a boy runs" و "the "dog walks" شکل درستی دارند. اگر ما یک گرامر کامل بدهیم، آنگاه در تئوری، هر جمله درست می‌تواند بدین روش تشریح شود.

این مثال تعریف یک مفهوم عمومی بر حسب مفاهیم ساده را روشن کرد. ما با مفهوم سطح بالا آغاز می‌کنیم، که در اینجا $\langle sentence\rangle$ است، و متناوباً آن را کاهش می‌دهیم تا به بلوک‌های ساختمانی غیر قابل کاهش در زبان برسیم. تعمیم این ایده‌ها، ما را به گرامرهای صوری رهنمون می‌سازد.

تعریف ۱-۱ : یک گرامر G بصورت چهارتایی زیر تعریف می‌شود :

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

جاییکه V یک مجموعه متناهی از اشیاء به نام **متغیرها** است،

T یک مجموعه متناهی ازاشیاء به نام **نمادهای پایانی** است،

$S \in V$ یک نماد خاص به نام متغیر **شروع** است،

P یک مجموعه متناهی از **قوانین تولید** است.

فرض می‌شود که مجموعه های V و T غیر تهی و مجزا هستند.

قوانین تولید، قلب یک گرامر می‌اشند، آنها مشخص می‌کنند که یک گرامر چگونه یک رشته را به رشته دیگر تبدیل می‌کند، و ازاین طریق، زبان وابسته به گرامر را تعریف می‌کنند. در بحث ما، ما فرض می‌کنیم که همه قوانین تولید به شکل

$$x \to y,$$

باشند، جاییکه x عنصری از $(V \cup T)^+$ و y عنصری در $(V \cup T)^*$ باشد. قوانین تولید به روش زیر به کار می روند : رشته w را به شکل زیر در نظر بگیرید :

$$w = uxv,$$

ما می‌گوییم قانون تولید $x \to y$ روی این رشته قابل کاربرد است، و ما می‌توانیم از آن برای جایگزینی x توسط y استفاده نماییم، بدین ترتیب رشته جدیدی حاصل می‌شود :

$$z = uyv.$$

این امر بصورت زیر نوشته می‌شود:

$$w \Rightarrow z.$$

ما می‌گوییم w، z را **مشتق** می‌کند، یا z از w مشتق شده است. رشته‌های متوالی با بکارگیری قوانین تولید از گرامر با ترتیب دلخواه مشتق می‌شوند. یک قانون تولید می‌تواند هر جایی که قابل کاربرد باشد، استفاده شود، و می‌تواند هر جایی که مطلوب باشد، بکار رود. اگر

$$w_1 \Rightarrow w_2 \Rightarrow \cdots \Rightarrow w_n,$$

گوییم رشته w_1، w_n را مشتق می‌کند، و می‌نویسیم :

www.prozhe.com

$$w_1 \stackrel{*}{\Rightarrow} w_n.$$

* نشان می‌دهد که یک تعداد نامشخصی از مراحل (شامل صفر) می‌توانند برای اشتقاق w_n از w_1 استفاده شود. بنابراین

$$w \stackrel{*}{\Rightarrow} w$$

نیز یک نمونه است.

با کاربرد قوانین تولید در ترتیب متفاوت، یک گرامر داده شده می‌تواند بطور طبیعی رشته‌های زیادی را تولید نماید. مجموعه همه چنین رشته‌های پایانی، زبان تعریف شده و یا تولید شده بوسیله گرامر می‌باشد.

تعریف ۱-۲ : فرض کنید $G = (V, T, S, P)$ یک گرامر باشد. آنگاه مجموعه

$$L(G) = \{w \in T^* : S \stackrel{*}{\Rightarrow} w\}$$

زبان تولید شده بوسیله G است.

اگر $w \in L(G)$ ، آنگاه دنباله

$$S \Rightarrow w_1 \Rightarrow w_2 \Rightarrow \cdots \Rightarrow w_n \Rightarrow w$$

اشتقاقی از جمله w است. رشته‌های $S, w_1, w_2, \ldots, w_n$ ، که شامل متغیرها به همراه پایانه‌ها می‌باشند، **شکل‌های جمله‌ای** از اشتقاق نامیده می‌شوند.

مثال ۱-۱۱ : گرامر زیر را در نظر بگیرید :

$$G = (\{S\}, \{a, b\}, S, P),$$

که P بصورت زیر داده شده است

$$S \to aSb,$$
$$S \to \lambda.$$

آنگاه

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb,$$

بنابراین ما می‌توانیم بنویسیم

$$S \stackrel{*}{\Rightarrow} aabb.$$

رشته $aabb$ جمله‌ای در زبان تولید شده بوسیله G است، در حالیکه $aaSbb$ یک شکل جمله‌ای است.

یک گرامر G کاملاً زبان $L(G)$ را تعریف می‌کند، ولی ممکن است بدست آوردن یک توصیف بسیار صریح از زبان از طریق گرامر، ساده نباشد. در اینجا، جواب نسبتاً واضح است. سخت نیست که ببینیم

$$L(G) = \{a^n b^n : n \geq 0\},$$

و اثبات آن، ساده میباشد. اگر ما توجه کنیم که قانون $S \to aSb$ بازگشتی است، یک اثبات بوسیله استقرا، خودش را پیشنهاد می‌کند. ما ابتدا نشان می‌دهیم که همه شکل‌های جمله‌ای باید به شکل زیر باشند

$$w_i = a^i S b^i. \tag{۱-۷}$$

فرض کنید که (۱-۷) برای هر شکل جمله‌ای w_i با طول $2i+1$ یا کمتر برقرار باشد. برای بدست آوردن شکل جمله‌ای دیگر (که یک جمله نیست)، ما می‌توانیم فقط قانون تولید $S \to aSb$ رابکار ببریم. از این کاربرد داریم

$$a^i S b^i \Rightarrow a^{i+1} S b^{i+1},$$

طوریکه هر شکل جمله‌ای به طول $2i+3$ نیز به شکل (۱-۷) است. ازآنجایی که (۱-۷) برای $i=1$ آشکارا درست است، بوسیله استقرا برای هر i نیز برقرار است. درنهایت، برای بدست آوردن یک جمله، ما باید قانون تولید $S \to \lambda$ را بکار ببریم، و می‌بینیم که

$$S \overset{*}{\Rightarrow} a^n S b^n \Rightarrow a^n b^n$$

همه اشتقاق‌های ممکن را ارائه می‌دهد. بنابراین، G می‌تواند فقط رشته‌هایی به شکل $a^n b^n$ را مشتق نماید.

ما همچنین باید نشان دهیم که همه رشته‌ها از این شکل قابل اشتقاق هستند. این کار آسان است. ما به سادگی قانون $S \to aSb$ را به تعداد دفعات مورد نیاز به کار می‌بریم، و بوسیله کاربرد قانون $S \to \lambda$ آن را دنبال می‌نماییم.

مثال ۱-۱۲: گرامری پیدا کنید که زبان زیر را تولید کند.

$$L = \{a^n b^{n+1} : n \geq 0\}.$$

ایده پشت مثال قبلی می‌تواند در این مورد توسعه یابد. همه آنچیزی که نیاز داریم تولید یک b اضافی است. این کار می‌تواند بوسیله قانون تولید $S \to Ab$ انجام شود، در حالیکه دیگر قوانین تولید باید طوری انتخاب شوند که A بتواند زبان مثال قبل را مشتق کند. با استدلال به این سبک، ما می‌توانیم گرامر $G = (\{S, A\}, \{a, b\}, S, P)$ را با قوانین تولید زیر بدست آوریم

$$S \to Ab,$$
$$A \to aAb,$$
$$A \to \lambda.$$

تعدادی جمله خاص را مشتق کنید تا مطمئن شوید که گرامر بدست آمده، کار می‌کند.

مثال‌های بالا نسبتاً آسان هستند، آرگومان‌های سخت ممکن است سخت بنظر برسند. ولی اغلب پیدا کردن یک گرامر برای زبانی که به یک روش غیر صوری توصیف می‌شود، یا دادن یک مشخصه ذاتی از زبان تعریف شده توسط یک گرامر آسان نیست. برای نشان دادن اینکه یک زبان داده شده واقعاً

بوسیله یک گرامر خاص G تولید می‌شود، ما باید بتوانیم نشان دهیم که الف) هر $w \in L$ می‌تواند از S با استفاده از G حاصل شود، و ب) هر رشته تولید شده توسط گرامر در L می‌باشد.

مثال ۱ ۱۳ : فرض کنید $\Sigma = \{a,b\}$، و $n_a(w)$ و $n_b(w)$ بترتیب نشاندهنده تعداد a ها و b ها در رشته w باشند. گرامر G با قوانین تولید

$$S \to SS,$$
$$S \to \lambda,$$
$$S \to aSb,$$
$$S \to bSa,$$

زبان زیر را تولید می‌کنند

$$L = \{w : n_a(w) = n_b(w)\}.$$

این ادعا خیلی واضح نیست، و ما نیاز به فراهم نمودن آرگومانهای اطمینان داریم.

ابتدا، واضح است که هر شکل جمله‌ای از G به تعداد مساوی a و b دارد، زیرا تنها قوانینی که یک a را تولید می‌کنند، به نام‌های $S \to aSb$ و $S \to bSa$ بطور همزمان یک b را نیز تولید می‌نمایند. بنابراین هر عنصر از $L(G)$ در L نیز هست. مشکل‌تر است که ببینیم هر رشته‌ای در L می‌تواند با G تولید شود.

فرض کنید از نگرش روی مسئله آغاز کنیم، و فرض کنید $w \in L$ می‌تواند شکل‌های مختلفی داشته باشد. فرض کنید w با یک a آغاز و به یک b ختم می‌شود. آنگاه آن دارای شکل زیر است

$$w = aw_1b,$$

جاییکه w_1 نیز در L است. ما می‌توانیم این مورد را مشتقی بدانیم که از قاعده زیر آغاز شده است

$$S \Rightarrow aSb,$$

اگر S واقعاً هر رشته‌ای در L را مشتق نماید. آرگومان مشابهی می‌تواند انجام شود اگر w با یک b آغاز و به یک a ختم شود. ولی این شامل همه موارد نیست، زیرا یک رشته در L می‌تواند با نماد یکسانی آغاز شده و نیز خاتمه یابد. اگر بخواهیم رشته‌ای ازاین نوع بنویسیم، گوییم $aabbba$، می‌بینیم که این رشته می‌تواند به عنوان اتصال دو رشته کوتاه‌تر $aabb$ و ba در نظر گرفته شود که هر دو نیز در L می‌باشند. آیا این در کل درست است؟ برای نمایش این که این مطلب واقعاً درست است، ما می‌توانیم از آرگومان زیر استفاده کنیم : فرض کنید که ازانتهای سمت چپ رشته آغاز می‌کنیم، و برای هر a، ۱+ و برای هر b، ۱- را می‌شماریم. اگر یک رشته w با a آغاز شده و خاتمه یابد، آنگاه شمارش پس ازسمت چپ ترین نماد ۱+ و بلافاصله قبل از سمت راست ترین نماد ۱- خواهد بود. بنابراین، شمارش در جایی در وسط رشته از صفر می‌گذرد، و نشاندهنده این مطلب است که رشته باید به شکل زیر باشد

$$w = w_1w_2,$$

جاییکه w_1 و w_2 در L هستند. این مورد می‌تواند بوسیله قانون تولید $S \to SS$ حاصل شود.

از آنجایی که ما آرگومان حدسی را دیدیم، آماده هستیم تا ادامه دهیم. مجدداً از استقرا استفاده می‌کنیم. فرض کنید که هر $w \in L$ با $|w| \le 2n$ می‌تواند از G مشتق شود. هر $w \in L$ با طول $2n+2$ را بگیرید. اگر $w = aw_1b$، آنگاه w_1 در L است، و $|w_1| = 2n$. بنابراین، بنا به فرض داریم:

$$S \stackrel{*}{\Rightarrow} w_1.$$

آنگاه

$$S \Rightarrow aSb \stackrel{*}{\Rightarrow} aw_1b = w$$

ممکن است، و w می‌تواند از G مشتق شود. آشکارا آرگومان‌های مشابه می توانند انجام گیرد اگر $w = bw_1a$.

اگر w به این شکل نباشد، یعنی با نماد یکسانی آغاز شده و خاتمه یابد، آنگاه آرگومان شمارش به ما می‌گوید که این رشته باید به شکل $w = w_1w_2$ باشد که w_1 و w_2 هر دو در L می‌باشند و دارای طول کمتر یا مساوی $2n$ می باشند. بنابراین می‌بینیم که

$$S \Rightarrow SS \stackrel{*}{\Rightarrow} w_1S \stackrel{*}{\Rightarrow} w_1w_2 = w$$

ممکن است.

ازآنجایی که فرض استقرا برای $n = 1$ آشکارا برقرار است، یک پایه داریم، و ادعا برای هر n درست است، و آرگومان ما تکمیل می‌شود.

معمولاً، یک زبان داده شده دارای گرامرهای زیادی است که آن را تولید نماید. هر چند این گرامرها مختلف باشند، آنها از بعضی جهات معادلند. ما می‌گوییم که دو گرامر G_1 و G_2 **معادل** هستند، اگر آنها زبان یکسانی را تولید نمایند. یعنی، اگر

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

همچنانچه ما بعداً خواهیم دید، همیشه ساده نیست که ببینیم دو گرامر معادل می‌باشند.

مثال ۱-۱۴ : گرامر $G_1 = (\{A, S\}, \{a, b\}, S, P_1)$ را در نظر بگیرید، که P_1 شامل قوانین تولید زیر است:

$$S \to aAb \mid \lambda,$$
$$A \to aAb \mid \lambda.$$

در اینجا ما یک نماد کوته نوشته قراردادی را معرفی می کنیم که در آن چندین قانون تولید با سمت چپ یکسان روی یک خط نوشته می‌شوند، که موارد طرف راست بوسیله | مجزا می‌شوند. در این نماد، $S \to aSb \mid \lambda$ به جای دو قانون $S \to aSb$ و $S \to \lambda$ استفاده می‌شود.

این گرامر معادل با گرامر G در مثال ۱-۱ می‌باشد. اثبات معادل بودن آسان است، زیرا کافی است نشان دهیم که

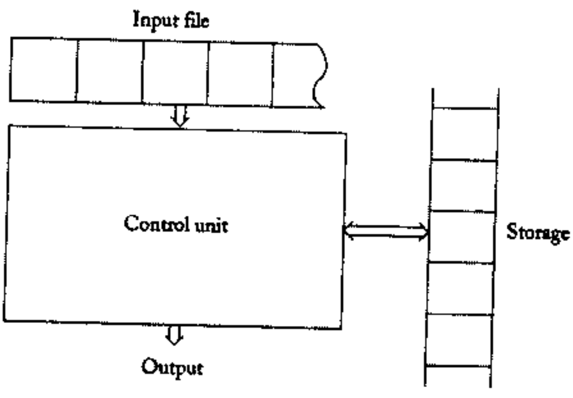
$$L(G_1) = \{a^n b^n : n \geq 0\}.$$

ما این را به عنوان یک تمرین باقی می‌گذاریم.

ماشین‌ها

یک ماشین، یک مدل انتزاعی از یک کامپیوتر رقمی است. هر ماشین، شامل برخی ویژگی‌های ضروری است. دارای مکانیسمی برای خواندن ورودی است. فرض می‌شود که ورودی، رشته‌ای روی الفبای داده شده است، و روی یک **فایل ورودی** نوشته شده است، که ماشین می‌تواند آن را بخواند ولی قادر به تغییر آن نیست. فایل ورودی به سلول‌هایی تقسیم شده است که هر یک از آنها قادر به نگهداری یک نماد می‌باشند. مکانیسم ورودی می‌تواند فایل ورودی را از چپ به راست و در هر لحظه یک نماد بخواند. مکانیسم ورودی می‌تواند انتهای رشته ورودی را (بوسیله حس شرط انتهای فایل) کشف نماید. ماشین می‌تواند خروجی را به چندین شکل تولید نماید. ماشین می‌تواند دارای یک دستگاه **ذخیره سازی موقت** باشد که شامل تعداد نامحدودی از سلول‌ها است که هر یک قادر به نگهداری یک نماد واحد از الفبا می‌باشند (لزوماً با الفبای ورودی یکسان نیست). ماشین می‌تواند محتوای سلول‌های حافظه را خوانده و تغییر دهد. در نهایت، ماشین دارای **واحد کنترل** است، که می‌تواند در هر یک از تعداد محدود از **حالات داخلی** باشد، و می‌تواند حالت را با روشی تعریف شده تغییر دهد. شکل ۱-۳ یک ارائه شماتیکی از ماشین عمومی را نمایش می‌دهد.

فرض می‌شود که یک ماشین در یک قاب زمان گسسته عمل می‌نماید. در هر زمان مفروضی، واحد کنترل در برخی حالت داخلی است، و مکانیسم ورودی یک نماد خاص را بر روی فایل ورودی پویش می‌نماید. حالت داخلی واحد کنترل در مرحله بعدی زمانی بوسیله **حالت بعدی** یا **تابع انتقال** تعیین می‌گردد. این تابع انتقال، حالت بعدی را بر حسب حالت فعلی، نماد فعلی ورودی، و اطلاعاتی که اکنون در حافظه



شکل ۱-۳

موقت هستند، می‌دهد. در حین انتقال از یک بازه زمانی به بعدی، خروجی ممکن است تولید شود یا اطلاعات در حافظه موقت تغییر کند. واژه **پیکربندی** برای مراجعه به یک حالت خاص از واحد کنترل، فایل ورودی، و حافظه موقت استفاده می‌شود. انتقال ماشین از یک پیکربندی به بعدی، یک **حرکت** نامیده می‌شود.

این مدل عمومی همه ماشین‌هایی که در این کتاب بحث می‌شود، پوشش می‌دهد. یک کنترل حالت متناهی برای همه موارد خاص رایج خواهد بود، ولی تفاوت‌ها از روشی که خروجی‌ها تولید می‌شوند، و طبیعت حافظه موقت ناشی می‌شود. طبیعت حافظه موقت تاثیر قوی‌تری روی انواع خاص ماشین‌ها دارد.

برای بحث‌های بعدی، لازم است تا بین **ماشین‌های قطعی و ماشین‌های غیر قطعی** تمایز قائل شویم. یک ماشین قطعی، ماشینی است که هر حرکت منحصراً بوسیله پیکربندی فعلی تعیین می گردد. اگر ما حالت داخلی، ورودی، و محتوای حافظه موقت را بدانیم، ما می‌توانیم رفتار آینده ماشین را دقیقاً پیش بینی نماییم. در یک ماشین غیر قطعی، چنین نیست. در هر نقطه، یک ماشین غیر قطعی ممکن است دارای چندین حرکت ممکن باشد، بنابراین ما می‌توانیم فقط مجموعه‌ای از عملیات ممکن را پیش‌بینی نماییم. رابطه بین ماشین قطعی و غیر قطعی از انواع مختلف، نقش مهمی در مطالعه ما ایفا می‌نماید.

پذیرنده ماشینی است که پاسخ خروجی آن به یک "بلی" یا "خیر" ساده محدود می‌شود. بسته به رشته ورودی، یک پذیرنده ممکن است آن را پذیرفته و یا رد نماید. یک ماشین عمومی‌تر که قادر به تولید رشته‌هایی از نمادها به عنوان خروجی باشد، یک **تراگذر** نامیده می‌شود. اگر چه ما چند مثال ساده از تراگذرها در بخش بعدی ارائه می‌دهیم، علاقه اصلی ما در این کتاب، پذیرنده‌ها می‌باشند.

تمرین‌ها

۱- با استفاده از استقرا روی n نشان دهید که $|u^n| = n|u|$ برای هر رشته u و هر n.

۲- معکوس یک رشته، که در بالا غیر صوری معرفی گردید، می‌تواند بوسیله نقش‌های بازگشتی دقیق‌تر تعریف گردد

$$a^R = a,$$
$$(wa)^R = aw^R,$$

برای هر $a \in \Sigma, w \in \Sigma^*$. ازرابطه زیربرای اثبات آن استفاده نمایید

$$(uv)^R = v^R u^R,$$

برای هر $u, v \in \Sigma^+$.

۳- ثابت کنید که $(w^R)^R = w$ برای هر $w \in \Sigma^*$.

۴- فرض کنید $L = \{ab, aa, baa\}$. کدام یک ازرشته‌های زیر در L^* می‌باشند:

$abaabaaabaa, aaaabaaaa, baaaaabaaaab, baaaaabaa$ ؟

۵- زبان‌های مثال‌های ۱-۱۲ و ۱-۱۳ را در نظر بگیرید. برای کدامیک $L = L^*$ درست است؟

۶- آیا زبان‌هایی وجود دارند که $\overline{L^*} = \overline{L}^*$ ؟

۷- ثابت کنید که

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

برای همه زبان‌های L_1 و L_2.

۸- نشان دهید که $(L^*)^* = L^*$ برای همه زبان‌ها.

۹- ادعاهای زیر را ثابت یا رد کنید.

الف) $(L_1 \cup L_2)^R = L_1^R \cup L_2^R$ برای همه زبان‌های L_1 و L_2.

ب) $(L^R)^* = (L^*)^R$ برای همه زبان‌های L.

۱۰- گرامرهایی روی $\Sigma = \{a,b\}$ پیدا کنید که مجموعه‌های زیر را تولید کنند.

الف) همه رشته‌هایی که دقیقاً یک a دارند،

ب) همه رشته‌هایی که حداقل یک a دارند،

ج) همه رشته‌هایی که بیش از سه a ندارند.

د) همه رشته‌هایی که حداقل سه a دارند.

در هر مورد، دلایل متقاعد کننده‌ای بیاورید که گرامری را که شما ارائه داده‌اید واقعاً زبان مورد نظر را تولید می‌کند.

۱۱- یک توصیف ساده از زبانی ارائه دهید که بوسیله گرامری با قوانین تولید زیر تولید می‌شود

$$S \to aA,$$
$$A \to bS,$$
$$S \to \lambda.$$

۱۲- چه زبانی توسط گرامری با قوانین تولید زیر تولید می‌شود؟

$$S \to Aa,$$
$$A \to B,$$
$$B \to Aa.$$

۱۳- برای هر یک از زبان‌های زیر، گرامری بیابید که آن را تولید نماید.

الف) $L_1 = \{a^n b^m : n \geq 0, m > n\}$

ب) $L_2 = \{a^n b^{2n} : n \geq 0\}$

ج) $L_3 = \{a^{n+2} b^n : n \geq 1\}$

د) $L_4 = \{a^n b^{n-3} : n \geq 3\}$

ه) $L_1 L_2$

و) $L_1 \cup L_2$

ز) L_1^3

ح) L_1^*

ط) $L_1 - \overline{L_2}$

۱۴- گرامرهایی برای زبان‌های زیر روی $\Sigma = \{a\}$ بیابید.*

الف) $L = \{w : |w| \bmod 3 = 0\}$

ب) $L = \{w : |w| \bmod 3 > 0\}$

ج) $L = \{w : |w| \bmod 3 \neq |w| \bmod 2\}$

د) $L = \{w : |w| \bmod 3 \geq |w| \bmod 2\}$

۱۵- گرامری بیابید که زبان زیر را تولید کند.

$$L = \{ww^R : w \in \{a,b\}^+\}.$$

برای پاسخ خود توجیه کاملی ارائه دهید.

۱۶- با استفاده از نماد مثال ۱-۱۳، گرامرهایی برای زبان‌های زیر پیدا کنید. فرض کنید $\Sigma = \{a,b\}$.

الف) $L = \{w : n_a(w) = n_b(w) + 1\}$

ب) $L = \{w : n_a(w) > n_b(w)\}$

ج)* $L = \{w : n_a(w) = 2n_b(w)\}$

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

۱۷- تمارین ۱۶(الف) و ۱۶(د) را برای $\Sigma = \{a,b,c\}$ تکرار کنید.

۱۸- با تکمیل دلایل مثال ۱-۱۴ نشان دهید که $L(G_1)$ حقیقتا زبان داده شده را تولید می‌کند.

۱۹- آیا دو گرامر با قوانین تولید

$$S \to aSb \mid ab \mid \lambda,$$

و

$$S \to aAb \mid ab,$$
$$A \to aAb \mid \lambda,$$

معادلند؟ فرض کنید که در هر دو مورد، S نشانه شروع باشد.

۲۰- نشان دهید که گرامر $G = (\{S\}, \{a,b\}, S, P)$ با قوانین تولید

$$S \to SS \mid SSS \mid aSb \mid bSa \mid \lambda,$$

با گرامر ارائه شده در مثال ۱-۱۳ معادل است.

۲۱- تا کنون، ما مثال‌هایی از گرامرهای نسبتاً ساده ارائه دادیم، هر قانون تولید دارای تنها یک متغیر در سمت چپ بود. همچنانچه خواهیم دید، چنین گرامرهایی خیلی مهم می‌باشند، ولی تعریف ۱-۱ شکل‌های عمومی‌تری را نشان می‌دهد.

گرامر $G = (\{A, B, C, D, E, S\}, \{a\}, S, P)$، با قوانین تولید زیر را در نظر بگیرید:

$$\begin{aligned}
&S \to ABaC,\\
&Ba \to aaB,\\
&BC \to DC \mid E,\\
&aD \to Da,\\
&AD \to AB,\\
&aE \to Ea,\\
&AE \to \lambda.
\end{aligned}$$

سه جمله متفاوت در $L(G)$ را مشتق کنید. از روی این جملات، $L(G)$ را حدس بزنید.*

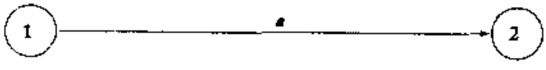
۱-۳ برخی کاربردها*

اگر چه ما روی خلاصه و قسمت‌های ریاضی از زبان‌های صوری و ماشین‌ها تاکید می‌کنیم، اینگونه برداشت می‌شود که این مفاهیم کاربرد گسترده‌ای در علم کامپیوتر دارند. در حقیقت یک موضوع معمولی است که به همه قسمتهای خاص مرتبط می‌شود. در این قسمت ما با برخی از مثال‌ها آشنا می‌شویم که به خواننده این اطمینان را می‌دهد که اینهایی را که ما می‌خوانیم، مجموعه‌ای از انتزاعات نیست، بلکه چیزی است که در فهمیدن مسائل مهم و حقیقی به ما کمک می‌کند.

زبانهای صوری و گرامرها در ارتباط با زبانهای برنامه نویسی کاربرد زیادی دارند. در بیشتر برنامه نویسی‌ها ما با درک شهودی کم یا زیاد درباره زمانی که می‌نویسیم کار می‌کنیم. اگر چه ممکن است در آینده دور از آن استفاده کنیم. ما نیاز داریم به شرح دقیقی از جدول دستورات رجوع کنیم تا بیشتر متن‌های برنامه را بفهمیم. اگر ما یک کامپایلر یا برنامه تصحیح کننده را بنویسیم، یک شرح دقیق از زبانهای مورد نیاز در هر مرحله داریم. در این میان راهی که زبانهای برنامه نویسی می‌توانند تعریف شوند، گرامرهایی هستند که بطور گسترده مورد استفاده قرار می‌گیرند.

گرامرهایی که یک نوع زبان خاص مانند پاسکال را شـــرح می‌دهند بسیار گرانقیمت هستند. توجه کنید که یک زبان کوچک بخشی از یک زبان بزرگتر است.

* چنانکه در مقدمه عنوان شد، ستاره‌ای که به دنبال عنوان بیاید نشان دهنده موضوع اختیاری است.



شکل ۱-۴

مثال ۱-۱۵ : مجموعه‌ای از شناسه‌های مجاز در زبان پاسکال، یک زبان است. بطور غیر صوری، مجموعه‌ای از همه رشته‌هایی است که با یک حرف آغاز می‌شوند و بوسیله تعداد دلخواهی از حروف یا ارقام دنبال می‌شوند. گرامر زیر این تعریف غیر رسمی را دقیق می‌سازد.

$$
\begin{aligned}
&\langle id\rangle \rightarrow \langle letter\rangle \ \ \langle rest\rangle,\\
&\langle rest\rangle \rightarrow \langle letter\rangle \ \ \langle rest\rangle \mid \langle digit\rangle \ \ \langle rest\rangle \mid \lambda,\\
&\langle letter\rangle \rightarrow a \mid b \mid \ldots \mid z\\
&\langle digit\rangle \rightarrow 0 \mid 1 \mid \ldots \mid 9
\end{aligned}
$$

در این گرامر، $\langle id\rangle, \langle letter\rangle, \langle digit\rangle$ و $\langle rest\rangle$ متغیر و $a, b, \ldots, z, 0, 1, \ldots, 9$ پایانه هستند. یک اشتقاق شناسه $a0$ عبارت است از

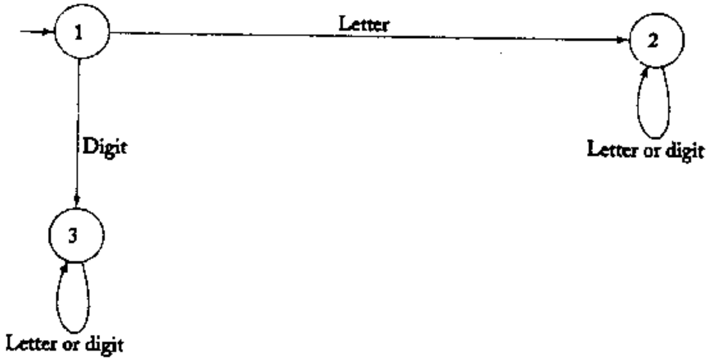
$$
\begin{aligned}
\langle id\rangle &\Rightarrow \langle letter\rangle \ \ \langle rest\rangle\\
&\Rightarrow a \ \ \langle rest\rangle\\
&\Rightarrow a \ \ \langle digit\rangle \ \ \langle rest\rangle\\
&\Rightarrow a0\\
&\Rightarrow a0.
\end{aligned}
$$

تعریف زبانهای برنامه نویسی از بین گرامرها معمول و خیلی مفید است، اما وجود دیگر چیزهایی که در دسترس هستند. مثلاً ما می‌توانیم یک زبان را بوسیله یک پردازنده شرح دهیم. هر قسمت از رشته قسمتی از زبان را می‌پذیرد. در مورد یک روش دقیق بحث خواهیم کرد که ما به یک تعریف صوری‌تر از یک ماشین نیاز داریم. برای مثال می‌توانیم یک روش شهودی ارائه دهیم.

یک ماشین می‌تواند بصورت یک گراف دارای رئوس که نمایش دهنده حالات داخلی است، ارائه شود. برچسب‌های روی یال ها نشان می‌دهد که در این حالت چه اتفاقی افتاده است (بر حسب ورودی و خروجی). برای مثال شکل ۱-۴ انتقالی از حالت۱ به حالت۲ را نشان می‌دهد، هنگامی که نماد ورودی a باشد. با داشتن این تصویر شهودی در ذهن، اجازه دهید تا به روشی دیگر از توصیف شناسه‌های پاسکال بنگریم.

مثال ۱-۱۶ : شکل ۱-۵ یک ماشین است که همه شناسه‌های معتبر پاسکال را می‌پذیرد. برخی تفسیرها ضروری است. فرض می‌کنیم که ماشین ابتدا در حالت۱ باشد. ما این مطلب را بوسیله رسم یک فلش (که ازهیچ راسی آغاز نشده است) به این حالت نشان داده‌ایم. مطابق همیشه، رشته از چپ به راست خوانده

می‌شود، هر کاراکتر در یک مرحله. هنگامی که اولین کاراکتر، یک حرف باشد ماشین به حالت۲ می‌رود، و پس از آن محتوای باقیمانده رشته اهمیتی ندارد. بنابراین حالت۲ نشان دهنده حالت "بله" از



شکل ۱-۵

پذیرنده است. بر عکس، اگر اولین نماد، یک رقم باشد، ماشین به حالت۳ که نشان دهنده حالت "نه" هست می‌رود، و در آنجا باقی می‌ماند. در راه حل ما، فرض می‌کنیم که هیچ ورودی به جز حروف یا ارقام امکان پذیر نیست.

کامپایلرها و دیگر مترجم‌ها که یک برنامه را از یک زبان به زبان دیگر تغییر می‌دهند،در مثال‌ها استفاده گسترده‌ای دارند. زبانهای برنامه نویسی می‌توانند بوسیله گرامرها تعریف شوند. در مثال ۱-۱۵ گرامرها و ماشین‌ها نقش اساسی در فرآیند تصمیم گیری بوسیله قطعه کد خاص که بوسیله یک زبان برنامه نویسی پذیرفته شده ایفا می‌کند. مثال بالا یک اشاره به چگونگی انجام آن می‌نماید. مثال‌های بعدی را نیز مشاهده خواهید کرد.

یک زمینه مهم دیگر کاربرد طراحی رقمی است که در آنجا مفاهیم مربوط به تراگذرها مطرح می‌شود. اگر چه این موضوع در اینجا مورد بحث گسترده واقع نخواهد شد، اما یک مثال ساده داده خواهد شد. اصولاً به کامپیوتر می‌توان به عنوان یک ماشین نگریست. اگر چه چنین نگرشی ممکن است که خیلی مناسب نباشد. فرض کنید که ثبات داخلی و حافظه اصلی کامپیوتر، واحد کنترل ماشین باشد، آنگاه ماشین دارای 2^n حالت داخلی است که n در اینجا تعداد بیت‌های ثبات و حافظه است. حتی با یک n کوچک، این عدد آنقدر بزرگ می‌شود که با این تعداد حالت، امکان کار کردن وجود ندارد، اما اگر به یک واحد خیلی کوچکتر از این نگاه کنیم، آنگاه نظریه ماشین، یک ابزار طراحی سودمند است.

مثال ۱-۱۷ : یک جمع کننده دودویی جزء لاینفک هر کامپیوتر همه منظوره است. چنین جمع کننده‌ای دو رشته از بیت‌ها را که نمایانگر دو عدد هستند به عنوان ورودی دریافت می‌کند، و مجموع آنها را به عنوان خروجی تولید می‌نماید. برای سادگی کار فرض کنید که فقط اعداد مثبت را جمع می‌زنیم و از نمایشی بصورت زیر استفاده می‌کنیم

$$x = a_0 a_1 ... a_n$$

که نمایانگر عدد صحیح $v(x)=\sum_{i=0}^{n} a_i 2^i$ می‌باشد. این نحوه نمایش، عکس نمایش معمول اعداد دودویی است.

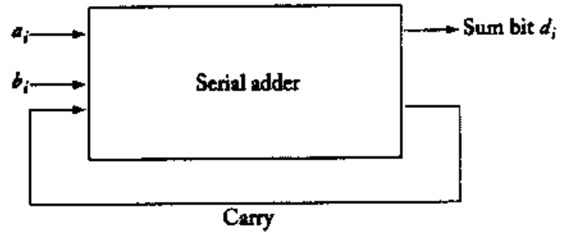
یک جمع کننده سری، دو عدد مانند $x = a_0 a_1 \ldots a_n$ و $y = b_0 b_1 .. b_n$ را بصورت بیت به بیت ازچپ به راست جمع می‌زند. جمع هر بیت، یک رقم برای جمع و یک رقم نقلی برای موقعیت بالاتر تولید می‌نماید. جدول جمع دودویی (شکل ۱-۶) این فرآیند را نشان می‌دهد.

یک بلوک دیاگرام از نوعی که ما هنگامیکه کامپیوترها را در ابتدا مطالعه می کردیم، دیدیم در شکل ۱-۷ نشان داده شده است. این شکل نشان می‌دهد که یک جمع کننده، جعبه‌ای است که دو بیت را به عنوان ورودی می‌پذیرد، و مجموع و بیت نقلی آنها را تولید می‌نماید. این تصویر، عملکرد یک جمع کننده را توضیح می‌دهد، اما چیزی در مورد ساختار درونی آن نمی گوید. یک ماشین (در اینجا یک تراگذر) می‌تواند این مسئله را واضح‌تر بیان نماید.

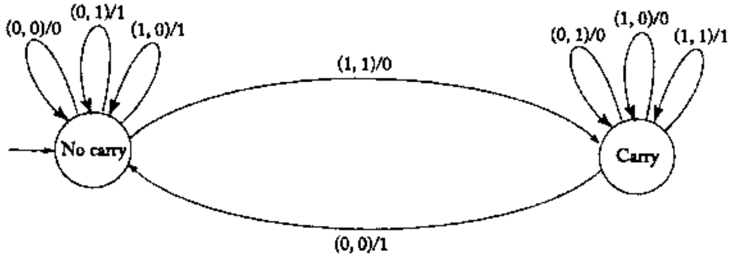
ورودی به تراگذر زوج بیت‌های (a_i, b_i) می‌باشد و خروجی آن بیت مجموع d_i است. باز هم می‌توانیم ماشین را بصورت گراف نمایش دهیم. البته یالهای آن را با $(a_i, b_i)/d_i$ علامت گذاری می کنیم. رقم نقلی از یک بیت به بیت بعدی توسط ماشین از طریق دو حالت داخلی "نقلی" و "بدون

a_i \ b_i	0	1
0	0 No carry	1 No carry
1	1 No carry	0 Carry

شکل ۱-۶



شکل ۱-۷



شکل ۱-۸

نقلی" بخاطر سپرده شود. در ابتدا تراگذر در حالت "بدون نقلی" خواهد بود. ماشین در این حالت باقی می‌ماند تا زمانی که با زوج بیت (1.1) مواجه شود، که در اینصورت یک رقم نقلی تولید می‌شود و باعث می‌شود ماشین به حالت "نقلی" منتقل شود. وجود رقم نقلی در جمع زوج بیت بعدی به حساب می‌آید. تصویر کاملی از جمع کننده سری در شکل ۱-۸ نشان داده شده است. آن را با چندین مثال دنبال کنید تا طرز عملکرد آن را دقیقاً بفهمید.

همچنانچه این مثال نشان می دهد، ماشین به عنوان پلی بین توصیف بسیار سطح بالا و تابعی از یک مدار و پیاده سازی منطقی آن توسط ترانزیستورها، گیت ها، و فلیپ فلاپ ها عمل می نماید. ماشین نشان می دهد که منطق تصمیم به اندازه کافی رسمی است تا خودش را بر عملیات ریاضی منطبق سازد. بدین دلیل، روشهای طراحی منطقی بر مفاهیمی از نظریه ماشین استوارند. خوانندگان علاقه‌مند می‌توانند به مطالب مربوط به این موضوع مثلاً Kovahi 1978 مراجعه کنند.

تمرین‌ها

۱- یک گرامر برای مجموعه اعداد صحیح در C بنویسید.

۲- یک پذیرنده برای اعداد صحیح در C طراحی کنید.

۳- یک گرامر که همه اعداد حقیقی در C را تولید کند، بنویسید.

۴- فرض کنید در یک زبان برنامه نویسی، شناسه‌هایی را که با یک حرف شروع می‌شوند و شامل حداقل یک و حداکثر سه رقم هستند و می‌توانند هر تعداد حرف را دارا باشند، مجاز باشند. یک گرامر و یک پذیرنده برای چنین مجموعه‌ای از شناسه‌ها را ارائه کنید.

۵- یک گرامر برای اعلان var در پاسکال بنویسید.

۶- در سیستم اعداد رومی، اعداد بوسیله رشته‌هایی روی الفبای {M, D, C, L, X, V, I} ارائه می‌شوند. پذیرنده‌ای طراحی کنید که رشته‌هایی را بپذیرد که اعداد رومی را بدرستی نشان دهند. برای سادگی، عمل "تفریق" عادی را که در آن عدد نه بوسیله IX ارائه می‌شود، بوسیله عمل "جمع" معادل که از VIIII استفاده می‌کند، جایگزین نمایید.

www.prozhe.com

۷- ما فرض می‌کنیم که یک ماشین در چارچوبی از مراحل زمانی گسسته کار می‌کند، ولی این منظر تاثیر اندکی روی بحث بعدی ما دارد. اگر چه در طراحی منطقی، عنصر زمان اهمیت قابل توجهی دارد.

به منظور همزمان شدن سیگنال‌هایی که از نقاط مختلف کامپیوتر می‌آیند، مدارات تاخیر مورد نیازند. یک تراگذر تک-تاخیره، وسیله‌ای است که ورودی (که به عنوان رشته‌ای پیوسته از نمادها نگریسته می‌شود) را یک واحد زمانی دیرتر، مجدداً تولید می‌کند. بخصوص، اگر تراگذر، نماد a را به عنوان ورودی در زمان t بخواند، آن نماد را به عنوان خروجی در زمان t+1 مجدداً تولید خواهد کرد. در زمان $t = 0$، تراگذر هیچ خروجی ندارد. ما این مطلب را بدین گونه بیان می‌کنیم که تراگذر ورودی $a_1a_2\dots$ را به خروجی $\lambda a_1a_2\dots$ ترجمه می‌کند.

گرافی رسم کنید که نشان می‌دهد چگونه چنین تراگذر تک-تاخیره‌ای ممکن است روی $\Sigma = \{a,b\}$ طراحی شود.

۸- یک تراگذر n-واحد تاخیره، تراگذری است که ورودی را n واحد زمانی دیرتر، مجدداً تولید می‌کند. یعنی ورودی $a_1a_2\dots$ به $\lambda^n a_1a_2\dots$ ترجمه می‌شود بدین معنا که تراگذر برای n قطعه زمانی نخست، هیچ خروجی تولید نمی‌کند.

الف) یک تراگذر با دو-واحد تاخیر روی $\Sigma = \{a,b\}$ بسازید.

ب) نشان دهید که یک تراگذر با n-واحد تاخیر باید حداقل دارای $|\Sigma|^n$ حالت باشد.

۹- مکمل دوی یک رشته دودویی، که یک عدد صحیح مثبت را نشان می‌دهد ابتدا بوسیله مکمل کردن هر بیت، و سپس افزودن یک به کم ارزش‌ترین بیت حاصل می‌شود. تراگذری طراحی کنید که رشته‌های بیتی را به مکمل دوی آنها ترجمه کند، فرض کنید که عدد دودویی مانند مثال ۱-۱۷ ارائه شده است، که در آن بیت‌های کم ارزش‌تر در سمت چپ رشته قرار دارند.

۱۰- تراگذری طراحی کنید که یک رشته دودویی را به معادل مبنای هشت آن تبدیل کند. برای مثال، رشته بیتی ۰۰۱۱۰۱۱۱۰ باید خروجی ۱۵۶ را تولید کند.

۱۱- فرض کنید $a_1a_2\dots$ یک رشته بیتی ورودی باشد. تراگذری طراحی کنید که توازن هر زیررشته سه بیتی را محاسبه نماید. بخصوص، تراگذر باید خروجی زیر را تولید کند

$$\pi_1 = \pi_2 = 0,$$
$$\pi_i = (a_{i-2} + a_{i-1} + a_i) \bmod 2, i = 3,4,\dots$$

برای مثال، ورودی ۱۱۰۱۱۱ باید خروجی ۰۰۰۰۰۱ را تولید نماید.

۱۲- تراگذری طراحی کنید که رشته‌های بیتی $a_1a_2a_3\dots$ را پذیرفته و مقدار دودویی باقیمانده حاصل تقسیم هر مجموعه از سه بیت متوالی بر پنج را محاسبه نماید. بخصوص، تراگذر باید $m_1, m_2, m_3, \dots$ را تولید کند، که

$$m_1 = m_2 = 0,$$

$$m_i = (4a_i + 2a_{i-1} + a_{i-2}) \bmod 5, \quad i = 3,4,\ldots$$

۱۳- کامپیوترهای رقمی معمولاً با استفاده از انواع روش‌های کدگذاری همه اطلاعات را بوسیله رشته‌های بیتی نمایش می‌دهند. برای مثال، اطلاعات کاراکتری می‌تواند با استفاده از سیستم شناخته شده ASCII کدگذاری شود.

برای این تمرین، بترتیب دو مجموعه الفبای $\{a,b,c,d\}$ و $\{0,1\}$ و یک کدگذاری از اولی به دومی رادر نظر بگیرید، که بوسیله $a \to 00, b \to 01, c \to 10, d \to 11$ تعریف شده است. تراگذری برای کدگشایی رشته‌های روی $\{0,1\}$ به پیام اصلی بسازید. برای مثال، ورودی ۰۱۰۰۱۱ باید خروجی bad را تولید نماید.

۱۴- فرض کنید x و y دو عدد دودویی مثبت باشند. تراگذری طراحی کنید که خروجی آن $\max(x, y)$ باشد.

۲
ماشین‌های متناهی

بحث مقدماتی ما در فصل اول در مورد مفاهیم اساسی محاسبات، بخصوص بحث ماشین، مختصر و غیر صوری بود. در این نقطه، فقط یک درک کلی از ماشین و چگونگی ارائه آن بوسیله یک گراف داریم. در ادامه، باید دقیق‌تر بوده و تعاریف صوری را مهیا نموده، و شروع به توسعه نتایج بسیار دقیق نماییم. ما با پذیرنده‌های متناهی که نوع خاص و ساده‌ای از شمای عمومی تعریف شده در فصل قبل است، آغاز می‌کنیم. این نوع از ماشین، دارای هیچ حافظه موقتی نیست. بدلیل اینکه فایل ورودی نمی‌تواند بازنویسی شود، یک ماشین متناهی به شدت دارای ظرفیت محدودی در به یادآوردن اطلاعات در طول محاسبه می‌باشد. مقدار کمی از اطلاعات می‌تواند در واحد کنترل بوسیله قرار گرفتن این واحد در یک حالت خاص نگهداری شود. ولی بدلیل اینکه تعداد حالات، متناهی است، یک ماشین متناهی فقط با حالاتی سر و کار دارد که اطلاعات مورد ذخیره در آنها در هر زمان به شدت محدود باشد. ماشین موجود درمثال ۱-۱۶ نمونه‌ای از یک پذیرنده متناهی است.

۲-۱ پذیرنده‌های متناهی قطعی

اولین نوع ماشینی که به همراه جزئیات مطالعه می‌کنیم پذیرنده‌های متناهی هستند که دارای عملکرد قطعی می‌باشند. با تعریف صوری دقیقی از پذیرنده‌های قطعی آغاز می‌نماییم.

پذیرنده‌های قطعی و گراف‌های انتقال

تعریف ۲-۱ : یک **پذیرنده متناهی قطعی** یا **dfa** بوسیله پنج تایی

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

تعریف می‌شود که

Q مجموعه‌ای متناهی از حالات داخلی است،

Σ مجموعه‌ای متناهی از نمادها به نام **الفبای ورودی** است،

$\delta : Q \times \Sigma \to Q$ یک تابع کلی به نام **تابع انتقال** است،

$q_0 \in Q$ **حالت اولیه** است،

$F \subseteq Q$ مجموعه‌ای از **حالات نهایی** است.

یک پذیرنده متناهی قطعی با روش زیر عمل می‌کند. فرض می‌شود که در ابتدا در حالت q_0 باشد، و مکانیسم ورودی روی سمت چپ‌ترین نماد از رشته ورودی باشد. در حین هر حرکت ماشین، مکانیسم ورودی یک موقعیت به سمت راست جلو می‌رود، طوریکه هر حرکت یک نماد ورودی را مصرف می‌کند. هنگامی که به انتهای رشته برسیم، اگر ماشین در یکی از حالات نهایی‌اش باشد، رشته پذیرفته است، در غیر اینصورت رشته پذیرفته نمی‌شود. مکانیسم ورودی می‌تواند فقط از چپ به راست حرکت کرده، و در هر مرحله دقیقاً یک نماد را بخواند. انتقالات از یک حالت داخلی به دیگری بوسیله تابع انتقال δ نمایش داده می‌شود. برای مثال، اگر

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

باشد، در اینصورت اگر پذیرنده متناهی قطعی در حالت q_0 بوده، و نماد ورودی فعلی a باشد، ماشین به حالت q_1 خواهد رفت.

در بحث ماشین‌ها، ضروری است که یک تصویر شفاف و شهودی برای کار با آنها داشته باشیم. برای بصری سازی و نمایش ماشین متناهی، از **گراف‌های انتقال** که در آن رئوس، نشان دهنده حالات و یال‌ها نشان دهنده انتقالات می‌باشند، استفاده می‌کنیم. برچسب‌های روی رئوس، نام‌های حالات هستند، درحالیکه برچسب‌های روی یال‌ها، مقادیر فعلی نماد ورودی هستند. برای مثال، اگر q_0 و q_1 حالات داخلی یک پذیرنده متناهی قطعی M باشند، در اینصورت گراف مربوط به M دارای یک گره با برچسب q_0 و گره دیگری با برچسب q_1 خواهد بود. یک یال (q_0, q_1) با برچسب a نشان دهنده انتقال $\delta(q_0, a) = q_1$ می‌باشد. حالت اولیه بوسیله فلش ورودی بدون برچسب که از هیچ راسی آغاز نشده، شناسایی می‌شود. حالات نهایی با دو دایره نمایش داده می‌شوند.

به شیوه صوری تر، اگر $M = (Q, \Sigma, \delta, q_0, F)$ یک پذیرنده متناهی قطعی باشد، گراف انتقال مربوطه آن یعنی G_M دارای دقیقاً $|Q|$ راس بوده که هر یک با $q_i \in Q$ مختلفی برچسب گذاری شده‌اند. برای هر قانون انتقال $\delta(q_i, a) = q_j$، گراف دارای یال (q_i, q_j) با برچسب a می‌باشد. راس مربوط به q_0 **راس اولیه** نامیده می‌شود، در حالیکه رئوسی با برچسب $q_f \in F$ رئوس نهایی هستند. موضوع تبدیل از روی تعریف $(Q, \Sigma, \delta, q_0, F)$ یک پذیرنده متناهی قطعی به نمایش گراف انتقال آن و بالعکس، اهمیت ناچیزی دارد.

مثال ۲-۱: گراف شکل ۲-۱ پذیرنده متناهی قطعی

$$M = (\{q_0, q_1, q_2\}, \{0,1\}, \delta, q_0, \{q_1\})$$

را نمایش می‌دهد که δ بصورت زیر تعریف شده است.

$$\delta(q_0,0) = q_0, \qquad \delta(q_0,1) = q_1,$$
$$\delta(q_1,0) = q_0, \qquad \delta(q_1,1) = q_2,$$
$$\delta(q_2,0) = q_2, \qquad \delta(q_2,1) = q_1.$$

این پذیرنده متناهی قطعی رشته ۰۱ را می‌پذیرد. با شروع از حالت q_0، ابتدا نماد ۰ خوانده می‌شود. با نگاه به یال‌های گراف می‌بینیم که ماشین در حالت q_0 باقی می‌ماند. سپس با خواندن ۱ ماشین به حالت q_1 می‌رود. حالا در انتهای رشته و در عین حال در حالت نهایی q_1 هستیم. بنابراین رشته ۰۱ پذیرفته می‌شود. پذیرنده متناهی قطعی رشته ۰۰ را نمی‌پذیرد، زیرا پس از خواندن دو ۰ متوالی، در حالت q_0 خواهد بود. با استدلال مشابه، می‌بینیم که ماشین رشته‌های ۱۰۱، ۰۱۱۱، و ۱۱۰۰۱ را خواهد پذیرفت، ولی ۱۰۰ یا ۱۱۰۰ را نمی‌پذیرد.

می‌توان تابع انتقال توسعه یافته $\delta^* : Q \times \Sigma^* \to Q$ را معرفی کرد. آرگومان دوم δ^*، به جای یک نماد تنها، یک رشته است، و مقدار آن حالت ماشین را پس از خواندن آن رشته می‌دهد. برای مثال، اگر

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

و

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

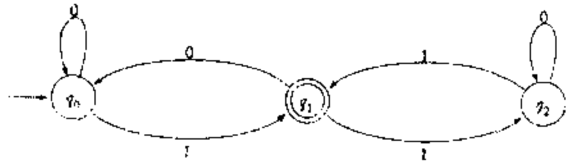
آنگاه

$$\delta^*(q_0, ab) = q_2.$$

بطور صوری، می‌توانیم δ^* را بصورت بازگشتی تعریف کنیم:

$$\delta^*(q, \lambda) = q, \qquad (۲-۱)$$

$$\delta^*(q, wa) = \delta(\delta^*(q, w), a), \qquad (۲-۲)$$



شکل ۲-۱

برای همه $q \in Q, w \in \Sigma^*, a \in \Sigma$. برای دیدن اینکه چرا این تعاریف مناسب است، این تعاریف را برای مورد ساده بالا بکار می‌بریم. ابتدا از رابطه (۲-۲) استفاده می‌کنیم تا بدست آوریم

$$\delta^*(q_0, ab) = \delta(\delta^*(q_0, a), b). \qquad (۲-۳)$$

ولی

$$\begin{aligned} \delta^*(q_0, a) &= \delta(\delta^*(q_0, \lambda), a) \\ &= \delta(q_0, a) \\ &= q_1. \end{aligned}$$

با جایگزینی این در رابطه (۲-۳) داریم

$$\delta^*(q_0, ab) = \delta(q_1, b) = q_2,$$

همانطور که انتظار می‌رفت.

زبان‌ها و پذیرنده‌های متناهی قطعی

با ارائه تعریف دقیقی از یک پذیرنده، آماده هستیم تا آنچه را که می‌خواهیم بوسیله زبان مربوطه بصورت صوری تعریف کنیم. ارتباط واضح است: زبان مجموعه‌ای از همه رشته‌های پذیرفته شده بوسیله ماشین است.

تعریف ۲-۳ : زبان پذیرفته شده بوسیله یک پذیرنده متناهی قطعی $M = (Q, \Sigma, \delta, q_0, F)$، مجموعه همه رشته‌هایی روی Σ است که بوسیله ماشین M پذیرفته می‌شود. با نماد رسمی،

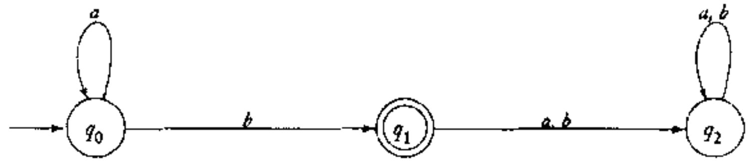
$$L(M) = \{w \in \Sigma^* : \delta^*(q_0, w) \in F\}.$$

توجه کنید که توابع δ و δ^* باید توابع کلی باشند. در هر مرحله، یک حرکت منحصر بفرد تعریف می‌شود، بنابراین آن را ماشین قطعی می‌نامیم. یک پذیرنده متناهی قطعی هر رشته‌ای در Σ^* را پردازش می‌کند و یا آن را می‌پذیرد و یا نمی‌پذیرد. عدم پذیرش بدین معنا است که پذیرنده متناهی قطعی در یک حالت غیر نهایی متوقف می‌شود، طوریکه

$$\overline{L(M)} = \{w \in \Sigma^* : \delta^*(q_0, w) \notin F\}.$$

مثال ۲-۲ : پذیرنده متناهی قطعی در شکل ۲-۲ را در نظر بگیرید.

در رسم شکل ۲-۲ از دو برچسب روی یک یال واحد استفاده نموده‌ایم. چنین یال‌هایی با چندین برچسب برای تند نویسی در مورد دو یا چند انتقال مجزا استفاده می‌شوند: هرگاه نماد ورودی با هر یک از برچسب‌های یال مطابقت داشته باشد، انتقال انجام می‌شود.



شکل ۲-۲

ماشین شکل ۲-۲ در حالت اولیه q_0 می‌ماند تا با اولین b مواجه شود. اگر این نماد، آخرین نماد ورودی باشد، در اینصورت رشته پذیرفته می‌شود. در غیر اینصورت، پذیرنده متناهی قطعی به حالت q_2 می‌رود که هرگز نمی‌تواند از آن خارج شود. حالت q_2 را **حالت تله** گویند. ما بوضوح از روی گراف می‌بینیم که ماشین همه رشته‌هایی را می‌پذیرد که شامل تعداد دلخواهی از a بوده و بوسیله یک b دنبال شوند. همه رشته‌های دیگر پذیرفته نخواهند شد. زبان پذیرفته شده بوسیله ماشین با نماد مجموعه عبارت است از

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

این مثال‌ها نشان می‌دهند که گراف‌های انتقال برای کار با ماشین‌های متناهی، مناسب می‌باشند. در عین حالیکه امکان استدلال بر پایه خواص تابع انتقال و توسعه آنها بر طبق روابط (۲-۱) و (۲-۲) می‌باشد، دنبال کردن نتایج مشکل می‌باشد. در بحث ما، تا حد امکان از گراف‌ها استفاده می‌کنیم که شهودی‌تر است. بدین منظور باید مطمئن باشیم که بوسیله نمایش، منحرف نمی‌شویم، و بحث‌های بر پایه گراف مانند استفاده از خواص صوری تابع δ معتبر هستند. نتیجه مقدماتی زیر این اطمینان را می‌دهد.

قضیه ۲-۱: فرض کنید $M = (Q, \Sigma, \delta, q_0, F)$ یک پذیرنده متناهی قطعی بوده، و G_M گراف انتقال مربوط به آن باشد. در اینصورت برای هر $q_i, q_j \in Q$ و $w \in \Sigma^+$ داریم $\delta^*(q_i, w) = q_j$ اگر و فقط اگر راهی با برچسب w در G_M از q_i به q_j وجود داشته باشد.

اثبات: این ادعا با بررسی موارد ساده‌ای مانند مثال ۲-۱ آشکار است، و می‌تواند با استفاده از مقدمه‌ای روی طول w ثابت شود. فرض کنید که ادعا برای همه رشته‌های v که $|v| \leq n$ باشد، درست است. دراینصورت هر رشته w با طول $n+1$ را در نظر می‌گیریم و آن را بصورت زیر می‌نویسیم

$$w = va.$$

حالا فرض کنید $\delta^*(q_i, v) = q_k$. بدلیل اینکه $|v| = n$ می‌باشد،پس باید راهی در G_M با برچسب v از q_i به q_k وجود داشته باشد. ولی اگر $\delta^*(q_i, w) = q_j$ باشد، دراینصورت M باید دارای انتقال $\delta(q_k, a) = q_j$ باشد، طوریکه با ساخت G_M یال (q_k, q_j) را با برچسب a خواهیم

داشت. بنابراین راهی در G_M بین q_i و q_j با برچسب $va = w$ وجود دارد. بدلیل اینکه درستی نتایج برای $n = 1$ آشکار است، می‌توانیم بوسیله استقرا ادعا کنیم که برای هر $w \in \Sigma^+$ داریم

$$\delta^*(q_i, w) = q_j \qquad (۲-۴)$$

این رابطه اشاره دارد که راهی در G_M از q_i به q_j با برچسب w وجود دارد.

استدلال می‌تواند با روش مستقیم ادامه یابد تا نشان دهد وجود چنین مسیری بر رابطه (۲-۴) اشاره دارد، بنابراین اثبات کامل می‌شود. ■

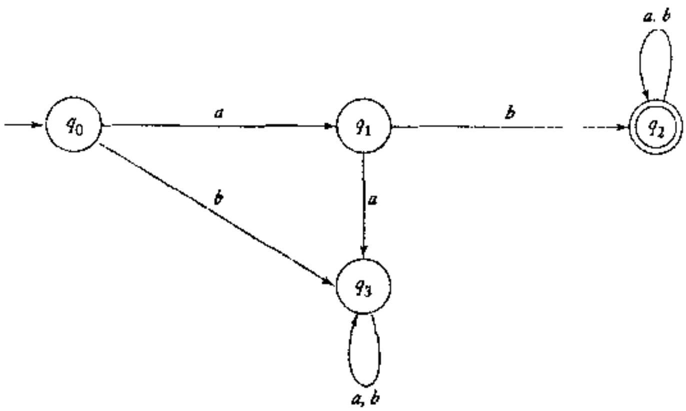
مجدداً، نتایج قضیه بطور شهودی آشکار است بطوریکه یک اثبات صوری، غیر ضروری بنظر می‌رسد. ما به دو دلیل به جزئیات می‌پردازیم. اولاً با وجود مثال نمونه از اثباتی بوسیله استقرا در رابطه با ماشین‌ها، ساده است. ثانیاً نتیجه، به دفعات مورد استفاده قرار می‌گیرد، بنابراین شروع و اثبات آن به عنوان تئوری، اجازه می‌دهد تا با اطمینان کامل با استفاده از گراف‌ها استدلال نماییم. این امر مثال‌ها و اثبات‌های ما را شفاف تر از زمانی می‌سازد که از خواص δ^* استفاده کنیم.

در حالیکه گراف‌ها برای بصری سازی ماشین‌ها مرسوم هستند، روشهای مفید دیگر نمایش نیز وجود دارند. برای مثال، ما می‌توانیم تابع δ را به صورت یک جدول نمایش دهیم. جدول شکل۲-۳ معادل با شکل ۲-۲ می‌باشد. در اینجا برچسب سطر، حالت فعلی است، در حالیکه برچسب ستون نماد ورودی فعلی را ارائه می‌دهد. عنصر موجود در جدول، حالت بعدی را تعریف می‌کند.

از این مثال آشکار است که یک پذیرنده متناهی قطعی می‌تواند به سادگی به عنوان یک برنامه کامپیوتری پیاده‌سازی شود، برای مثال، به عنوان یک جدول جستجوی ساده و یا به عنوان یک دنباله از جملات "if". بهترین پیاده‌سازی یا ارائه به کاربرد خاص بستگی دارد. گراف‌های انتقال برای انواع استدلال‌هایی که می‌خواهیم داشته باشیم، بسیار مرسوم هستند، بنابراین ما از آنها در اکثر بحثمان استفاده می‌کنیم.

	a	b
q_0	q_0	q_1
q_1	q_2	q_2
q_2	q_2	q_2

شکل ۲-۳



شکل ۲-۴

در ساخت ماشین‌ها برای زبانهایی که بطور غیر صوری تعریف شده‌اند، از استدلالی مشابه با آنچه که برای برنامه‌نویسی در زبانهای سطح بالاتر دیده‌ایم، بهره می‌بریم. ولی برنامه نویسی یک پذیرنده متناهی قطعی، کسل‌آور و گاهی اوقات از نظر مفهومی بخاطر این حقیقت که یک ماشین دارای چندین ویژگی قوی می‌باشد، پیچیده است.

مثال۲-۳: یک پذیرنده متناهی قطعی بیابید که مجموعه همه رشته‌هایی روی $\Sigma = \{a,b\}$ شروع شده با پیشوند ab را تشخیص دهد.

تنها موضوع در اینجا، دو نماد نخست رشته می‌باشد. پس از خوانده شدن آنها، نیاز به تصمیم گیری دیگری نمی‌باشد. بنابراین ما می‌توانیم مسئله را با یک ماشین که دارای چهار حالت می‌باشد حل نماییم، یک حالت اولیه، دو حالت برای تشخیص ab که به یک حالت نهایی تله ختم می‌شود، و یک حالت غیر نهایی تله. اگر اولین نماد یک a و دومی یک b باشد، ماشین به حالت نهایی تله می رود، جاییکه بدلیل عدم اهمیت بقیه ورودی در آن می‌ماند. از طرف دیگر، اگر نماد اول یک a نبوده و یا نماد دوم یک b نباشد، ماشین وارد یک حالت غیرنهایی دام می‌شود. این راه حل ساده در شکل۲-۴ نمایش داده شده است.

مثال ۲-۴: یک پذیرنده متناهی قطعی بیابید که همه رشته‌های روی $\{0,1\}$ به جز آنهایی که شامل زیر رشته ۰۰۱ باشند را بپذیرد.

در تصمیم گیری در مورد وقوع زیر رشته ۰۰۱، نه تنها به دانستن نماد ورودی جاری، بلکه به یادآوری اینکه یک یا دو ۰ مقدم بر آن نماد آمده است یا خیر، نیاز داریم. ما می‌توانیم این کار را با گذاشتن ماشین در حالات خاص و برچسب گذاری آنها مطابق آن انجام دهیم. مشابه با نام‌های متغیرها در یک زبان برنامه‌نویسی، نام‌های حالت به دلخواه بوده و می‌توانند به دلایل یادیاری انتخاب شوند. برای مثال، حالتی که دو ۰ بر نمادها مقدم باشد می‌تواند به سادگی بصورت ۰۰ برچسب گذاری شود.

اگر رشته با ۰۰۱ آغاز شود، در اینصورت باید پذیرفته نشود. این مطلب اشاره دارد که باید مسیری با برچسب ۰۰۱ از حالت اولیه به حالت غیر نهایی وجود داشته باشد. برای سهولت، این حالت غیر نهایی با ۰۰۱ برچسب گذاری می‌شود. این حالت باید یک حالت دام باشد، زیرا نمادهای بعدی اهمیتی ندارند. همه حالات دیگر حالات مورد پذیرش هستند.

تا اینجا ساختار اساسی راه حل را ارائه کردیم، ولی هنوز باید امکاناتی برای رخداد زیررشته ۰۰۱ در وسط ورودی اضافه کنیم. ما باید Q و δ را تعریف کنیم طوریکه هرآنچه در تصمیم گیری صحیح نیاز داریم توسط ماشین به یاد آورده شود. در این مورد، هنگامی که یک نماد خوانده می‌شود، ما نیاز به دانستن بخشی از رشته به سمت چپ می‌باشیم، برای مثال، آیا دو نماد قبلی ۰۰ بوده اند یا خیر. اگر ما این حالات را بوسیله نمادهای مربوطه برچسب گذاری نماییم، دیدن اینکه چه انتقالی باید صورت پذیرد بسیار آسان است. برای مثال،

$$\delta(00,0) = 00,$$

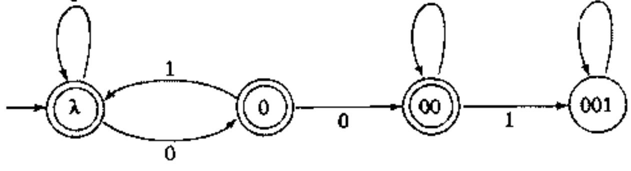
زیرا این وضعیت فقط زمانی رخ می‌دهد که سه ۰ متوالی وجود داشته باشد. ما فقط به دوتای آخری علاقه مندیم، حقیقتی که بوسیله نگهداشتن پذیرنده متناهی قطعی در حالت ۰۰ به یاد می‌آوریم. یک راه حل کامل در شکل ۲-۵ نمایش داده شده است. از این مثال می‌بینیم که چگونه برچسب‌های یادیاری مفید در مورد حالات برای نگهداری اطلاعات استفاده می‌شوند. با ردیابی چندین رشته مانند ۱۰۰۱۰۰ و ۱۰۱۰۱۰۰ می‌بینیم که راه حل واقعاً درست است.

زبان‌های منظم

هر ماشین متناهی زبانی را می‌پذیرد. اگر ما همه ماشین‌های متناهی را در نظر بگیریم، مجموعه‌ای از زبانهای مرتبط با آنها بدست می‌آوریم. چنین مجموعه‌ای از زبانها را یک **خانواده** گوییم. خانواده زبانهایی که بوسیله پذیرنده‌های متناهی قطعی پذیرفته می‌شود، کاملاً محدود است. ساختار و خواص زبانها در این خانواده همچنانچه مطالعه ما ادامه می‌یابد، واضح‌تر می‌گردد. در این زمان ما نامی به این خانواده منتسب می‌کنیم.

تعریف ۲-۳ : زبان L را منظم گویند اگر و فقط اگر پذیرنده متناهی قطعی M وجود داشته باشد طوریکه

$$L = L(M).$$



شکل ۲-۵

مثال ۲-۵: نشان دهید که زبان

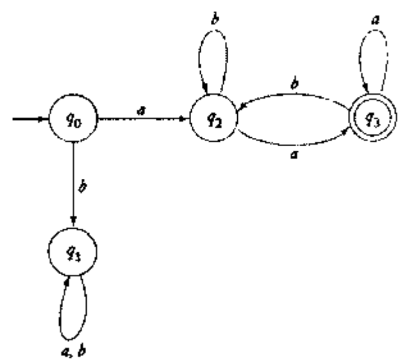
$$L = \{awa : w \in \{a,b\}^*\}$$

منظم است. برای نشان دادن اینکه این زبان یا هر زبان دیگری منظم است، همه کاری که باید انجام دهیم یافتن یک پذیرنده متناهی قطعی برای آن است. ساخت یک پذیرنده متناهی قطعی برای این زبان مشابه مثال ۲-۳ است، ولی کمی پیچیده‌تر است. آنچه که این پذیرنده متناهی قطعی باید انجام دهد، بررسی رشته‌ای است که با a شروع و به a ختم شود، و آنچه که بین آنها می‌آید اهمیتی ندارد. راه حل بوسیله این حقیقت که هیچ راه صریحی برای آزمایش انتهای رشته وجود ندارد، پیچیده می‌شود. به این مشکل اینگونه غلبه می‌کنیم که هر گاه پذیرنده متناهی قطعی با دومین a مواجه شود به حالت نهایی رود. اگر اینجا انتهای رشته نباشد، و b دیگری یافت شود، پذیرنده متناهی قطعی از حالت نهایی خارج می‌شود. پویش بدین طریق ادامه می‌یابد، هر a ماشین را به حالت نهایی‌اش بر می‌گرداند. راه حل کامل در شکل ۲-۶ نشان داده شده است. مجدداً، تعدادی مثال را ردیابی نمایید تا ببینید چرا این ماشین درست کار می‌کند. پس از یک یا دو آزمایش، آشکار خواهد بود که پذیرنده متناهی قطعی یک رشته را می‌پذیرد اگر و فقط اگر با a شروع و به a ختم شود. بدلیل اینکه ما یک پذیرنده متناهی قطعی برای این زبان ساختیم، می‌توانیم از روی تعریف ادعا کنیم که این زبان، منظم است.

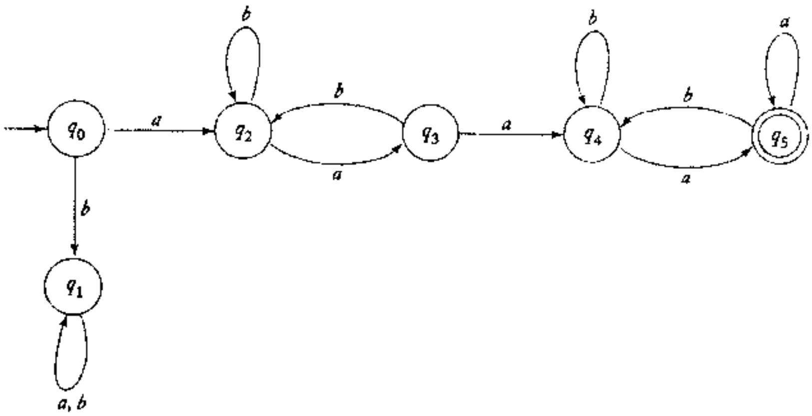
مثال ۲-۶: فرض کنید L زبان موجود در مثال ۲-۵ باشد. نشان دهید که L^2 منظم است. مجدداً بوسیله ساخت یک پذیرنده متناهی قطعی برای این زبان نشان می‌دهیم که این زبان، منظم است. می‌توانیم یک عبارت صریح برای زبان L^2 بنویسیم

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

بنابراین، نیاز به یک پذیرنده متناهی قطعی داریم که دو رشته متوالی با شکل ضرورتاً یکسان (ولی نه الزاماً مقادیر یکسان) را تشخیص دهد. دیاگرام شکل ۲-۶ می‌تواند به عنوان نقطه شروع استفاده شود، ولی راس



شکل ۲-۶



شکل ۲-۷

q_3 باید تغییر کند. این حالت دیگر حالت نهایی نیست، در اینجا ما باید نظر به زیررشته دوم را آغاز کنیم که به شکل awa می‌باشد. برای تشخیص زیررشته دوم، حالات اولین بخش را (با نام‌های جدید) تکرار می‌کنیم، در حالیکه q_3 به عنوان آغاز بخش دوم می‌باشد. بدلیل اینکه رشته کامل می‌تواند هر کجا که aa رخ دهد، به بخش‌های جزئی‌اش تقسیم شود، اجازه می‌دهیم که اولین رخداد دو a متوالی باعث شود تا ماشین وارد بخش دومش شود. اینکار را می‌توانیم بوسیله $\delta(q_3, a) = q_4$ انجام دهیم. راه حل کامل در شکل ۲-۷ نشان داده شده است. این پذیرنده متناهی قطعی زبان L^2 را می‌پذیرد، بنابراین این زبان، منظم است.

مثال آخر این حدس را پیشنهاد می‌کند که اگر زبان L، منظم باشد، در اینصورت ... ,L^2, L^3 نیز منظم هستند. ما بعداً خواهیم دید که این حدس واقعاً درست است.

تمرین‌ها

۱- کدام یک از رشته‌های ۰۰۰۱، ۰۱۰۰۱، ۰۰۰۰۱۱۰ بوسیله پذیرنده متناهی قطعی شکل ۲-۱ پذیرفته می‌شود؟

۲- روی $\Sigma = \{a, b\}$، پذیرنده‌های متناهی قطعی بسازید که مجموعه‌های زیر را بپذیرد

الف) همه رشته‌هایی با دقیقاً یک a،

ب) همه رشته‌هایی با حداقل یک a،

ج) همه رشته‌هایی که بیش از سه a ندارند،

د) همه رشته‌هایی با حداقل یک a و دقیقاً دو b.

ه) همه رشته‌هایی با دقیقاً دو a و بیش از دو b.

۳- نشان دهید اگر شکل ۲-۶ را تغییر دهیم، به گونه‌ای که q_3 به حالت غیر نهایی و q_0, q_1, q_2 به حالات نهایی تبدیل شوند، پذیرنده متناهی قطعی بدست آمده، $\overline{L}$ را می‌پذیرد.

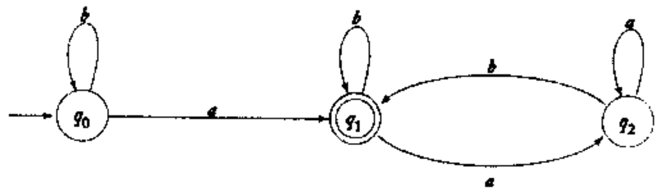
۴- نتایج تمرین قبلی را تعمیم دهید. بخصوص، نشان دهید اگر $M = (Q, \Sigma, \delta, q_0, F)$ و $\widehat{M} = (Q, \Sigma, \delta, q_0, Q - F)$ دو پذیرنده متناهی قطعی باشند، آنگاه $\overline{L(M)} = L(\widehat{M})$.

۵- پذیرنده های متناهی قطعی برای زبانهای زیر بدهید.

الف) $L = \{ab^5wb^4 : w \in \{a,b\}^*\}$

ب) $L = \{w_1abw_2 : w_1 \in \{a,b\}^*, w_2 \in \{a,b\}^*\}$

۶- یک توصیف بصورت نماد مجموعه‌ای برای زبانی که بوسیله ماشین نشان داده شده در دیاگرام زیر پذیرفته می‌شود، بدهید. آیا می‌توانید توصیف متنی از این زبان ارائه دهید؟



پذیرنده های متناهی قطعی برای زبانهای زیر روی $\Sigma = \{a,b\}$ بیابید.

الف) $L = \{w : |w| \bmod 3 = 0\}$

ب) $L = \{w : |w| \bmod 5 \neq 0\}$

ج) $L = \{w : n_a(w) \bmod 3 > 1\}$

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

ه) $L = \{w : (n_a(w) - n_b(w)) \bmod 3 > 0\}$

و) $L = \{w : |n_a(w) - n_b(w)| \bmod 3 < 2\}$

۷- یک دوره در یک رشته، زیر رشته‌ای با طول حداقل دو می‌باشد که دارای نمادهای یکسانی باشد. برای نمونه، رشته $abbbaab$ شامل دوره‌ای از b ها به طول سه و دوره‌ای از a ها به طول دو می‌باشد. پذیرنده‌های متناهی قطعی برای زبانهای زیر روی $\Sigma = \{a,b\}$ بیابید. *

الف) $L = \{w :$ شامل هیچ دوره ای با طول کمتر از چهار نباشد $w\}$

ب) $L = \{w :$ هر دوره ای از a ها دارای طول دو یا سه باشد$\}$

ج) $L = \{w :$ حداکثر دو دوره از a ها با طول سه وجود داشته باشد$\}$

د) $L = \{w :$ دقیقاً دو دوره از a ها با طول سه وجود داشته باشد$\}$

۸- مجموعه رشته‌های روی $\{0,1\}$ که بوسیله نیازمندیهای زیر تعریف می‌شوند، در نظر بگیرید. برای هر کدام، یک پذیرنده متناهی قطعی بسازید.

الف) پس از هر ۰۰ بلافاصله یک ۱ دنبال بیاید. برای مثال، رشته‌های ۱۰۱، ۰۰۱۰، ۰۰۱۰۰۱۱۰۰۱ در زبان هستند، ولی ۰۰۰۱ و ۰۰۱۰۰ در زبان نیستند.

ب) همه رشته‌هایی که شامل ۰۰ بوده ولی شامل ۰۰۰ نباشند.

ج) سمت چپ ترین نماد با سمت راست ترین نماد متفاوت باشد.

د) هر زیر رشته‌ای از چهار نماد که حداکثر دارای دو ۰ باشد. برای مثال، ۰۰۱۱۱۰ و ۰۱۱۰۰۱ در زبان هستند، ولی ۱۰۰۱۰ در زبان نیست زیرا یکی از زیر رشته‌های آن، یعنی ۰۰۱۰ دارای سه ۰ می‌باشد.

ه) همه رشته‌هایی با طول پنج یا بیشتر که در آنها چهارمین نماد از سمت راست از سمت چپ ترین نماد متفاوت باشد.

و) همه رشته‌هایی که دو نماد سمت چپ با دو نماد سمت راست یکسان باشند.

۹- یک پذیرنده متناهی قطعی بسازید که رشته‌های روی $\{0,1\}$ را بپذیرد، اگر و فقط اگر مقدار رشته که به عنوان نمایش دودویی از یک عدد صحیح تفسیر می‌شود، به پیمانه پنج برابر صفر شود. برای مثال، ۰۱۰۱ و ۱۱۱۱ که بترتیب نشاندهنده اعداد صحیح ۵ و ۱۵ می‌باشند، پذیرفته می‌شوند. *

۱۰- نشان دهید که زبان $L = \{vwv : v, w \in \{a,b\}^*, |v| = 2\}$ منظم است.

۱۱- نشان دهید که زبان $L = \{a^n : n \geq 4\}$ منظم است.

۱۲- نشان دهید که زبان $L = \{a^n : n \geq 0, n \neq 4\}$ منظم است.

۱۳- نشان دهید که زبان $L = \{a^n : n = i + jk, i, k \quad fixed, j = 0,1,2,\ldots\}$ منظم است.

۱۴- نشان دهید مجموعه همه اعداد حقیقی در C یک زبان منظم است.

۱۵- نشان دهید اگر L منظم باشد، آنگاه $L - \{\lambda\}$ نیز منظم است.

۱۶- با استفاده از روابط (۲-۱) و (۲-۲) نشان دهید که برای هر $w, v \in \Sigma^*$ داریم:

$$\delta^*(q, wv) = \delta^*(\delta^*(q, w), v)$$

۱۷- فرض کنید L زبان پذیرفته شده بوسیله ماشین شکل ۲-۲ باشد. یک پذیرنده متناهی قطعی بیابید که L^2 را بپذیرد.

۱۸- فرض کنید L زبان پذیرفته شده بوسیله ماشین شکل ۲-۲ باشد. یک پذیرنده متناهی قطعی بیابید که $L^2 - L$ را بپذیرد.

۱۹- فرض کنید L زبان مثال ۲-۵ باشد، نشان دهید که L^* منظم است.

۲۰- فرض کنید G_M یک گراف انتقال برای پذیرنده متناهی قطعی M باشد. موارد زیر را ثابت کنید.

الف) اگر $L(M)$ نامتناهی باشد، آنگاه G_M باید دارای حداقل یک چرخه باشد که مسیری از راس ابتدایی به راس دیگری در چرخه و مسیری از آن راس در چرخه به راس نهایی موجود باشد.

ب) اگر $L(M)$ متناهی باشد، آنگاه چنین چرخه ای وجود ندارد.

۲۱- فرض کنید عملیات *قطع* را تعریف کرده‌ایم که سمت راست ترین نماد از رشته را حذف می‌کند. برای مثال، $truncate(aaaba)$ برابر $aaab$ می‌باشد. این عملیات می‌تواند به زبانها تعمیم یابد :

$$truncate(L)=\{truncate(w): w\in L\}.$$

نشان دهید که چگونه با داشتن یک پذیرنده متناهی قطعی برای هر زبان منظم L ، می‌توان یک پذیرنده متناهی قطعی برای زبان $truncate(L)$ ساخت. سپس ثابت کنید اگر L زبانی منظمی باشد که شامل رشته λ نباشد، آنگاه $truncate(L)$ نیز منظم است.

۲۲- فرض کنید $x=a_0a_1\ldots a_n, y=b_0b_1\ldots b_n, z=c_0c_1\ldots c_n$ اعداد دودویی باشند به گونه‌ای که در مثال ۱-۱۷ تعریف شد. نشان دهید که مجموعه رشته‌هایی از سه تایی های

$$\begin{pmatrix}a_0\\b_0\\c_0\end{pmatrix}\begin{pmatrix}a_1\\b_1\\c_1\end{pmatrix}\cdots\begin{pmatrix}a_n\\b_n\\c_n\end{pmatrix},$$

که در آن a_i,b_i,c_i به گونه‌ای هستند که $x+y=z$ می‌باشد، یک زبان منظم است.

۲۳-با وجود آنکه زبان پذیرفته شده بوسیله یک پذیرنده متناهی قطعی داده شده، منحصر بفرد است، ولی بطور معمول پذیرنده‌های متناهی قطعی زیادی وجود دارند که یک زبان را می‌پذیرند. یک پذیرنده متناهی قطعی بیابید که دقیقاً دارای شش حالت بوده و همان زبانی را بپذیرد که پذیرنده متناهی قطعی در شکل ۲-۴ قبول می‌کند.●

۲-۲ پذیرنده‌های متناهی غیر قطعی

گر اجازه دهیم پذیرنده‌های متناهی بصورت غیر قطعی عمل نمایند، پیچیده‌تر خواهند شد. غیر قطعی ودن، قدرت می‌دهد، ولی در نخستین نگاه، ایده غیر معمولی است. ما بطور معمول به کامپیوترها بصورت کاملا قطعی فکر می‌کنیم، و عنصر انتخاب نابجا بنظر می‌رسد. با این وجود، همچنانکه در ادامه خواهیم ید، غیر قطعی بودن، یک نماد مفید است.

ریف یک پذیرنده غیر قطعی

یر قطعیت به معنای انتخاب حرکات در یک ماشین می‌باشد. به جای مجوز یک حرکت منحصربفرد در ر حالت، اجازه مجموعه‌ای از حرکات ممکن را می‌دهیم. بطور صوری، ما این امر را بوسیله تعریف تابع تقال انجام می‌دهیم بطوریکه برد آن مجموعه‌ای از حالات ممکن باشد.

ریف ۲-۴ : یک پذیرنده متناهی غیر قطعی یا nfa بوسیله پنج تایی

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

تعریف می‌شود که Q, Σ, q_0, F همان تعریفی را دارند که در پذیرنده‌های متناهی قطعی داشتند، ولی

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

توجه کنید که سه تفاوت اساسی بین این تعریف و تعریف پذیرنده متناهی قطعی وجود دارد. در یک پذیرنده غیر قطعی، برد δ در مجموعه توانی 2^Q است، بطوریکه مقدار آن یک عنصر واحد از Q نیست، بلکه زیرمجموعه‌ای از آن است. این زیر مجموعه، مجموعه همه حالات ممکنی که بوسیله انتقال مفروض قابل دسترسی هستند را تعریف می‌کند. برای نمونه، اگر حالت فعلی q_1 باشد، نماد a خوانده شود، و

$$\delta(q_1, a) = (q_0, q_2),$$

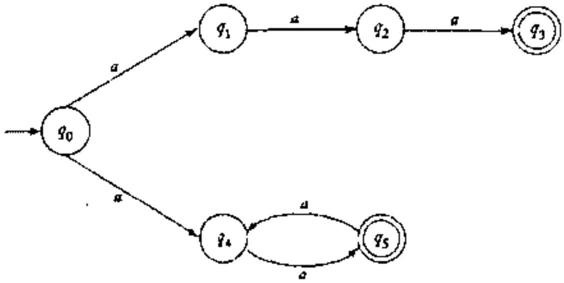
باشد، آنگاه یا q_0 یا q_2 می‌توانند حالت بعدی در پذیرنده متناهی غیر قطعی باشند. همچنین، اجازه می‌دهیم λ به عنوان دومین شناسه از δ ظاهر شود. این امر بدین معناست که پذیرنده متناهی غیرقطعی می‌تواند انتقالی بدون مصرف یک نماد ورودی انجام دهد. اگرچه ما هنوز فرض می‌کنیم که مکانیسم ورودی فقط می‌تواند از چپ به راست حرکت کند، ولی امکان توقف در برخی حرکات، وجود دارد. سرانجام، در یک پذیرنده متناهی غیرقطعی، مجموعه $\delta(q_i, a)$ ممکن است تهی باشد، بدین معنا که هیچ انتقالی برای این حالت خاص تعریف نشده است.

مشابه پذیرنده‌های متناهی قطعی، پذیرنده‌های غیر قطعی می‌توانند بوسیله گراف‌های انتقال نمایش داده شوند. رئوس بوسیله مجموعه Q تعیین می‌شوند، در حالیکه یال (q_i, q_j) با برچسب a در گراف است اگر و فقط اگر $\delta(q_i, a)$ شامل q_j باشد. توجه کنید از آنجایی که a ممکن است رشته تهی باشد، برخی از یال‌ها می‌توانند دارای برچسب λ باشند.

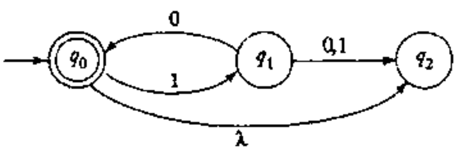
یک رشته بوسیله یک پذیرنده متناهی غیر قطعی پذیرفته می‌شود اگر دنباله‌ای از حرکات ممکن وجود داشته باشد که در انتهای رشته، ماشین را در یک حالت نهایی قرار دهد. یک رشته رد می‌شود (یعنی پذیرفته نمی‌شود) اگر هیچ دنباله ممکنی از حرکات وجود نداشته باشد که بتواند به یک حالت نهایی برسد. بنابراین، غیر قطعیت می‌تواند به عنوان دید حدسی نگریسته شود که بوسیله آن در هر حالت بهترین حرکت می‌تواند انتخاب شود (فرض کنید که پذیرنده متناهی غیر قطعی می‌خواهد هر رشته ای را بپذیرد).

مثال ۲-۷ : گراف انتقال موجود در شکل ۲-۸ را در نظر بگیرید. این گراف یک پذیرنده غیر قطعی را توصیف می‌نماید، زیرا دو انتقال با برچسب a داریم که از حالت q_0 خارج شده است.

مثال ۲-۸ : یک ماشین غیر قطعی در شکل ۲-۹ نشان داده شده است. این ماشین غیر قطعی است زیرا نه تنها چندین یال با برچسب یکسان دارد که از یک راس خارج شده‌اند، بلکه دارای حرکت λ می‌باشد. برخی انتقالات، مانند $\delta(q_2, 0)$ در گراف مشخص نشده‌اند. این موضوع به عنوان یک انتقال به مجموعه تهی تفسیر می‌شود، یعنی $\delta(q_2, 0) = \varnothing$. ماشین رشته λ، ۱۰۱۰، و ۱۰۱۰۱۰ را می پذیرد، ولی ۱۱۰ و



شکل ۲-۸



شکل ۲-۹

۱۰۱۰۰ را نمی‌پذیرد. توجه کنید که برای ۱۰ دو راه مختلف وجود دارد، که یکی به q_0 و دیگری به q_2 منجر می‌شود. اگرچه q_2 یک حالت نهایی نیست، ولی این رشته پذیرفته می‌شود زیرا یک راه وجود دارد که به یک حالت نهایی منجر می‌شود.

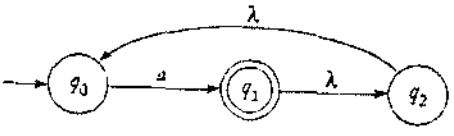
مجدداً، تابع انتقال می‌تواند تعمیم یابد بطوریکه شناسه دوم آن یک رشته باشد. ما به تابع انتقال تعمیم یافته δ^* نیاز داریم که اگر

$$\delta^*(q_i, w) = Q_j,$$

باشد آنگاه Q_j مجموعه همه حالات ممکنی است که ماشین می‌تواند با شروع از حالت q_i پس از خواندن رشته w در آنها باشد. یک تعریف بازگشتی از δ^* مشابه روابط (۲-۱) و (۲-۲) ممکن است، ولی فوق العاده آموزنده نیست. یک تعریف باارزش ساده‌تر می تواند بوسیله گراف‌های انتقال ارائه شود.

تعریف ۲-۵ : برای یک پذیرنده متناهی غیرقطعی، تابع انتقال تعمیم یافته تعریف می شود بطوریکه $\delta^*(q_i, w)$ شامل q_j خواهد بود اگر وفقط اگر راهی در گراف انتقال از q_i به q_j با برچسب w وجود داشته باشد. این امر برای همه $q_i, q_j \in Q$ و $w \in \Sigma^*$ برقرار است.

مثال ۲-۹ : شکل ۲-۱۰ یک پذیرنده متناهی غیرقطعی را نمایش می‌دهد که دارای چندین انتقال λ و چندین انتقال تعریف نشده مانند $\delta(q_2, a)$ می‌باشد.



شکل ۲-۱۰

فرض کنید می‌خواهیم $\delta^*(q_1,a)$ و $\delta^*(q_2,\lambda)$ را پیدا کنیم. راهی با برچسب a شامل دو انتقال λ از q_1 به خودش وجود دارد. با استفاده از برخی یال‌های λ به تعداد دوبار، می‌بینیم که راه‌هایی شامل انتقالات λ به q_0 و q_2 وجود دارد. بنابراین

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

از آنجایی که یک یال λ بین q_2 و q_0 وجود دارد، بلافاصله داریم که $\delta^*(q_2,\lambda)$ شامل q_0 می‌باشد. همچنین، از آنجایی که هر حالتی می‌تواند از خودش بوسیله هیچ حرکتی قابل دسترسی باشد، و در نتیجه از هیچ نماد ورودی استفاده نکند، $\delta^*(q_2,\lambda)$ شامل q_2 نیز می‌باشد. بنابراین

$$\delta^*(q_2,\lambda)=\{q_0,q_2\}.$$

با استفاده از چندین انتقال λ مورد نیاز، می‌توانید بررسی کنید که

$$\delta^*(q_2,aa)=\{q_0,q_1,q_2\}.$$

تعریف δ^* بوسیله راه‌های برچسب دار تا حدودی غیر صوری است، بنابراین مفید خواهد بود اگر نگاه دقیق‌تری به آن داشته باشیم. تعریف ۲-۵ مناسب است، از آنجایی که بین هر یک از رئوس v_i و v_j یا راهی با برچسب w وجود دارد و یا راهی وجود ندارد، نشان دهنده آن است که δ^* بصورت کامل تعریف شده است. آنچه که دیدن آن شاید کمی مشکل‌تر باشد آن است که این تعریف می‌تواند همیشه برای یافتن $\delta^*(q_i,w)$ استفاده شود.

در بخش ۱-۱ ، ما الگوریتمی برای یافتن همه مسیرهای ساده ممکن بین دو راس را توصیف کردیم. ما نمی‌توانیم از این الگوریتم مستقیماً استفاده نماییم، زیرا همچنانچه مثال ۲-۹ نشان می‌دهد، همیشه یک راه برچسب گذاری شده، یک مسیر ساده نیست. ما می‌توانیم الگوریتم مسیر ساده را با حذف این محدودیت که هیچ راس یا یالی نمی‌تواند تکرار شود، تغییر دهیم. حالا الگوریتم جدید با موفقیت همه راه‌های بطول یک، بطول دو، بطول سه و مانند آن را تولید می‌نماید.

هنوز یک مشکل وجود دارد. برای یک رشته مفروض w، راهی با برچسب w چه طولی می‌تواند داشته باشد؟ این مطلب فوراً واضح نیست. در مثال ۲-۹، راهی با برچسب a بین q_1 و q_2 دارای طول چهار است. مشکل بوسیله انتقالات λ ایجاد می‌شود، که راه را طولانی می‌نماید ولی در برچسب اثری ندارد. وضعیت بوسیله این نگرش ذخیره می‌شود : اگر بین دو راس v_i و v_j هر راهی با برچسب w وجود

داشته باشد، آنگاه باید راهی با برچسب w با طول نابیشتر از $\Lambda + (1 + \Lambda)|w|$ وجود داشته باشد، که Λ تعداد یال‌های λ در گراف است. استدلال در این مورد عبارت است از : با وجود آنکه یال‌های λ ممکن است تکرار شوند، همیشه راهی وجود دارد که در آن هر یال λ تکراری از یالی با برچسب یک نماد غیرتهی مجزا شده است. در غیر اینصورت، راه شامل حلقه‌ای با برچسب λ است، که می‌تواند بوسیله یک مسیر ساده بدون تغییر برچسب راه جایگزین شود. ما اثبات صوری این ادعا را به عنوان یک تمرین باقی می‌گذاریم.

با این مشاهده، ما روشی برای محاسبه $\delta^*(q_i, w)$ خواهیم داشت. ما همه راه‌ها بطول حداکثر $\Lambda + (1 + \Lambda)|w|$ که از v_i آغاز می‌شوند ارزیابی می‌نماییم. از بین آنها راه‌هایی را انتخاب می‌کنیم که دارای برچسب λ باشند. گره‌های انتهایی راه‌های انتخاب شده عناصر مجموعه $\delta^*(q_i, w)$ می‌باشند.

همچنان که اشاره کردیم، ممکن است که δ^* بصورت بازگشتی تعریف شود همانند آنچه که برای مورد قطعی انجام دادیم. متاسفانه، نتیجه خیلی شفاف نیست، و دنبال کردن استدلال با تابع انتقال توسعه یافته تعریف شده بدین طریق مشکل است. ما ترجیح می‌دهیم که از تعریف شهودی‌تر و قابل مدیریت‌تر دیگر در تعریف ۲-۵ استفاده نماییم.

همانند پذیرنده‌های محدود قطعی، زبان پذیرفته شده بوسیله یک پذیرنده متناهی غیر قطعی بطور صوری بوسیله تابع انتقال تعمیم یافته تعریف می‌شود.

تعریف ۲-۶ : زبان L پذیرفته شده بوسیله یک پذیرنده متناهی غیر قطعی $M = (Q, \Sigma, \delta, q_0, F)$، به عنوان مجموعه همه رشته‌های پذیرفته شده با مفهوم بالا تعریف می‌شود،

$$L(M) = \{w \in \Sigma^* : \delta^*(q_0, w) \cap F \neq \varnothing\}.$$

به عبارت دیگر، زبان شامل همه رشته‌های w است که برای آن راهی با برچسب w از راس اولیه گراف انتقال به راسی نهایی وجود دارد.

مثال ۲-۱۰ : زبان پذیرفته شده بوسیله ماشین شکل ۲-۹ چیست؟ از روی گراف به سادگی می‌بینیم که تنها راهی که پذیرنده متناهی غیر قطعی می‌تواند در یک حالت نهایی متوقف شود آن است که ورودی یا تکرار رشته ۱۰ یا رشته تهی باشد. بنابراین ماشین زبان $L = \{(10)^n : n \geq 0\}$ را می‌پذیرد.

اگر به این ماشین رشته $w = 110$ ارائه شود، چه اتفاقی می‌افتد؟ پس از خواندن پیشوند ۱۱، ماشین خود را در حالت q_2 می‌یابد، که در آن انتقال $\delta(q_2, 0)$ تعریف نشده است. ما چنین حالتی را یک **پیکربندی مرده** نامیم، و می‌توانیم آن را به عنوان حالتی تصور کنیم که ماشین بسادگی بدون انجام عملیات بیشتری متوقف می‌شود. ولی ما باید همیشه به خاطر داشته باشیم که چنین تصوراتی غیر دقیق هستند و با خود خطر تفسیر نادرست را بهمراه دارند. آنچه که می‌توانیم دقیق بگوییم آن است که

$$\delta^*(q_0, 110) = \varnothing.$$

بنابراین، با پردازش $w = 110$ نمی‌توان به هیچ حالت نهایی رسید، و بنابراین رشته پذیرفته نیست.

چرا عدم قطعیت ؟

در استدلال در مورد ماشین‌های غیر قطعی باید در استفاده از عقاید شهودی کاملاً محتاط باشیم. شهود می‌تواند به سادگی به گمراهی منجر شود، و ما باید قادر به ارائه استدلال‌های دقیق جهت اثبات نتایجمان باشیم. غیر قطعیت یک مفهوم مشکل است. کامپیوترهای رقمی کاملاً قطعی هستند، حالت آنها در هر زمان از روی ورودی و حالت اولیه بطور واحد قابل پیش بینی است. بنابراین طبیعی است که بپرسیم چرا ما ماشین‌های غیر قطعی را مطالعه می‌نماییم؟ ما سعی می کنیم سیستم‌های واقعی را مدل نماییم، بنابراین چرا شامل چنین ویژگی‌های غیر مکانیکی انتخابی هستند؟ ما به این پرسش می‌توانیم به راه‌های مختلف پاسخ دهیم.

بسیاری از الگوریتم‌های قطعی در چند مرحله نیازمند انتخاب هستند. یک مثال نمونه، برنامه بازی است. غالباً، بهترین حرکت مشخص نیست، ولی می‌تواند با استفاده از یک جستجوی کامل با بازگشت به عقب یافت شود. هنگامی که چندین انتخاب ممکن باشد، ما یکی را انتخاب کرده و آنرا دنبال می‌نماییم تا آنجا که واضح شود که آیا بهترین انتخاب بوده است یا خیر. اگر چنین نبود، ما به آخرین نقطه تصمیم عقب نشینی می‌کنیم و دیگر انتخاب‌ها را بررسی می‌کنیم. یک الگوریتم غیر قطعی که بتواند بهترین انتخاب را انجام دهد، قادر خواهد بود مسئله را بدون بازگشت به عقب حل نماید، ولی یک ماشین قطعی می‌تواند با کمی کار بیشتر غیر قطعیت را شبیه سازی نماید. بدین دلیل، ماشین‌های غیر قطعی می‌توانند به عنوان مدل هایی از الگوریتم‌های جستجو و بازگشت به عقب عمل کنند.

گاهی اوقات غیر قطعیت در حل آسان مسائل مفید است. به پذیرنده متناهی غیرقطعی در شکل ۲-۸ بنگرید. واضح است که باید انتخابی صورت گیرد. اولین انتخاب به پذیرش رشته a^3 منجر می‌شود، در حالیکه دومین انتخاب همه رشته‌هایی را می‌پذیرد که دارای تعداد زوج a ها باشند. زبان پذیرفته شده بوسیله پذیرنده متناهی غیر قطعی $\{a^3\} \bigcup \{a^{2n} : n \geq 1\}$ می‌باشد. با وجود آنکه یافتن یک پذیرنده متناهی قطعی برای این زبان ممکن است، غیر قطعیت کاملاً طبیعی است. زبان، اجتماع دو مجموعه کاملاً متفاوت است، و غیر قطعیت به ما اجازه می‌دهد در آغاز تصمیم بگیریم که کدام مورد را می‌خواهیم. راه حل قطعی به این وضوح به تعریف زبان مربوط نمی‌شود. همچنانچه جلو می‌رویم، مثال‌های دیگر و پذیرفته تر از قابل استفاده بودن عدم قطعیت را خواهیم دید.

به همان سبک، غیر قطعیت مکانیسمی موثر برای توصیف مختصر برخی زبان‌های پیچیده می‌باشد. توجه کنید که تعریف یک گرامر شامل یک عنصر غیر قطعی است. در

$$S \to aSb \mid \lambda$$

ما می‌توانیم در هر نقطه‌ای یا قانون اول یا قانون دوم را انتخاب کنیم. این به ما اجازه می‌دهد تا رشته های متفاوت بسیاری را با استفاده از فقط دو قانون، مشخص نماییم.

سرانجام، یک دلیل تکنیکی برای معرفی غیرقطعیت وجود دارد. همچنانچه خواهیم دید، نتایج مشخصی را برای پذیرنده‌های متناهی غیر قطعی ساده تر از پذیرنده‌های متناهی قطعی بنا می‌نهیم. نتیجه مهم بعدی ما نشان می‌دهد که هیچ تفاوت عمده‌ای بین این دو نوع از ماشین‌ها وجود ندارد. در نتیجه، استفاده از عدم قطعیت اغلب استدلال‌های صوری را بدون تاثیر بر کلیت نتایج آسان می کند.

تمرین‌ها

۱- با تمام جزئیات ادعای عنوان شده در بخش قبلی را ثابت کنید که اگر در یک گراف انتقال راهی با برچسب w وجود داشته باشد، آنگاه باید راهی با برچسب w با طول نابیشتر از $\Lambda + (1+\Lambda)|w|$ موجود باشد.

۲- یک پذیرنده متناهی قطعی بیابید که زبان تعریف شده بوسیله پذیرنده متناهی غیر قطعی در شکل ۲-۸ را بپذیرد.

۳- در شکل ۲-۹، $\delta^*(q_0,1011)$ و $\delta^*(q_1,01)$ را بیابید.

۴- در شکل ۲-۱۰، $\delta^*(q_0,a)$ و $\delta^*(q_1,\lambda)$ را بیابید.

۵- برای پذیرنده متناهی غیر قطعی در شکل ۲-۹، $\delta^*(q_0,1010)$ و $\delta^*(q_1,00)$ را بیابید.

۶- یک پذیرنده متناهی غیر قطعی با حداکثر پنج حالت برای مجموعه $\{abab^n : n \geq 0\} \cup \{aba^n : n \geq 0\}$ طراحی کنید.

۷- یک پذیرنده متناهی غیر قطعی با سه حالت بسازید که زبان $\{ab, abc\}^*$ را بپذیرد.

۸- آیا شما فکر می‌کنید تمرین ۷ می‌تواند با کمتر از سه حالت حل شود؟

۹- الف) یک پذیرنده متناهی غیر قطعی با سه حالت بیابید که زبان

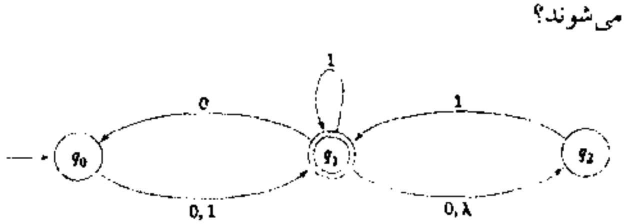
$$L = \{a^n : n \geq 1\} \cup \{b^m a^k : m \geq 0, k \geq 0\}$$

را بپذیرد.

ب) آیا فکر می‌کنید که زبان بخش الف می‌تواند بوسیله یک پذیرنده متناهی غیر قطعی با کمتر از سه حالت پذیرفته شود؟

۱۰- یک پذیرنده متناهی غیر قطعی با چهار حالت برای $L = \{a^n : n \geq 0\} \cup \{b^n a : n \geq 1\}$ بیابید.

۱۱- کدامیک از رشته‌های ۰۰، ۰۱۰۰۱، ۱۰۰۱۰، ۰۰۰، و ۰۰۰۰ بوسیله پذیرنده متناهی غیر قطعی زیر پذیرفته می‌شوند؟



۱۲- مکمل زبان پذیرفته شده بوسیله پذیرنده متناهی غیر قطعی در شکل ۲-۱۰ چیست؟

۱۳- فرض کنید L زبان پذیرفته شده بوسیله پذیرنده متناهی غیر قطعی در شکل ۲-۸ باشد. یک پذیرنده متناهی غیر قطعی بیابید که $L \cup \{a^5\}$ را بپذیرد.

۱۴- توصیف ساده ای از زبان تمرین ۱۲ ارائه دهید.

۱۵- یک پذیرنده متناهی غیر قطعی بیابید که $\{a\}^*$ را بپذیرد بطوریکه اگر در گراف انتقال آن یک یال تنها حذف شود (بدون هیچ تغییر دیگری)، ماشین بدست آمده $\{a\}$ را بپذیرد.

۱۶- آیا تمرین ۱۵ می‌تواند با استفاده از یک پذیرنده متناهی قطعی حل شود؟ اگر چنین است، راه حل را ارائه دهید، اگر نیست استدلالاتی قانع کننده برای نتیجه‌تان ارائه دهید.

۱۷- تغییر زیر در تعریف ۲-۶ را در نظر بگیرید. یک پذیرنده متناهی غیر قطعی با چندین حالت اولیه بوسیله پنج تایی

$$M = (Q, \Sigma, \delta, Q_0, F),$$

تعریف می‌شود که $Q_0 \subseteq Q$ مجموعه‌ای از حالات اولیه ممکن است. زبان پذیرفته شده بوسیله چنین ماشینی بصورت زیر تعریف می‌شود

$$L(M) = \{w : \text{باشد } q_f \text{ شامل } \delta^*(q_0, w), q_0 \in Q_0, q_f \in F \text{ برای هر}\}$$

نشان دهید که برای هر پذیرنده متناهی غیر قطعی با چندین حالت اولیه یک پذیرنده متناهی غیر قطعی با یک حالت اولیه تنها وجود دارد که همان زبان را می‌پذیرد.

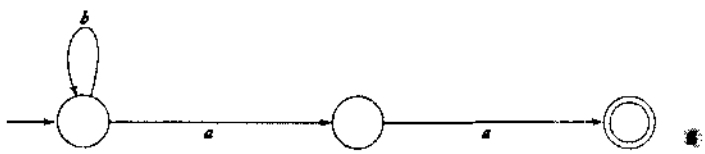
۱۸- فرض کنید در تمرین ۱۷ محدودیت $Q_0 \cap F = \varnothing$ را اعمال می‌کنیم. آیا این محدودیت تاثیری در نتیجه دارد؟

۱۹- با استفاده از تعریف ۲-۵ نشان دهید که برای هر پذیرنده متناهی غیر قطعی داریم

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

برای همه $q \in Q$ و همه $w, v \in \Sigma^*$.

۲۰- یک پذیرنده متناهی غیر قطعی که الف) دارای هیچ انتقالات λ نباشد، و ب) برای همه $q \in Q$ و همه $a \in \Sigma$، $\delta(q, a)$ شامل حداکثر یک عنصر باشد، گاهی اوقات یک پذیرنده متناهی غیر قطعی **غیر کامل** نامیده می‌شود. این امر معقول است زیرا شرایطی قابل تصور است که هیچ انتخابی جهت حرکت وجود نداشته باشد. برای $\Sigma = \{a, b\}$، پذیرنده متناهی قطعی غیر کامل زیر را به یک پذیرنده متناهی قطعی استاندارد تبدیل کنید.



۲-۳ معادل بودن پذیرنده‌های متناهی قطعی و غیر قطعی

ما اکنون به یک سوال اساسی می‌رسیم. پذیرنده‌های متناهی قطعی و پذیرنده‌های متناهی غیر قطعی از چه جهت متفاوتند؟ آشکارا، تعریف آنها تفاوت دارد، ولی این مطلب موجب هیچ تمایز اساسی بین آنها نمی‌شود. برای بررسی این سوال، ما عقیده معادل بودن ماشین‌ها را معرفی می‌کنیم.

تعریف ۲-۷ : دو پذیرنده متناهی M_1 و M_2 را معادل گویند اگر

$$L(M_1) = L(M_2),$$

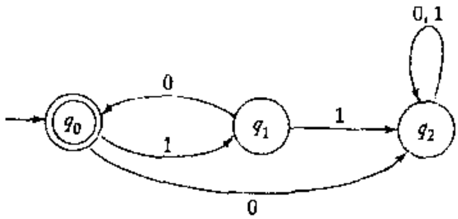
یعنی، اگر هر دو زبان یکسانی را بپذیرند.

همچنانچه ذکر شد، عموماً پذیرنده‌های بسیاری برای یک زبان مفروض وجود دارند، بنابراین هر پذیرنده متناهی قطعی یا پذیرنده متناهی غیر قطعی پذیرنده‌های معادل بسیاری دارند.

مثال ۲-۱۱ : پذیرنده متناهی قطعی در شکل ۲-۱۱ معادل با پذیرنده متناهی غیرقطعی در شکل ۲-۹ می‌باشد، زیرا هر دو زبان $\{(10)^n : n \geq 0\}$ را می‌پذیرند.

هنگامی که ما رده‌های مختلف ماشین‌ها را مقایسه می‌کنیم، سوالی که همیشه مطرح می‌شود این است که آیا یک رده از رده‌های دیگر قوی‌تر است. منظور ما از قوی‌تر بودن آن است که یک ماشین از یک نوع می‌تواند چیزی را انجام دهد که بوسیله هیچ ماشینی از نوع دیگر قابل انجام نباشد. اجازه دهید به این سوال در مورد پذیرنده‌های متناهی نگاه کنیم. از آنجایی که یک پذیرنده متناهی قطعی در اصل یک نوع محدود شده از پذیرنده متناهی غیر قطعی است، واضح است که هر زبانی بوسیله یک پذیرنده متناهی قطعی پذیرفته شود، بوسیله چند پذیرنده متناهی غیر قطعی نیز پذیرفته می‌شود. ولی معکوس آن اینقدر آشکار نیست. ما غیر قطعیت اضافه شده را داریم، بنابراین حداقل قابل تصور است زبانی وجود داشته باشد که بوسیله برخی پذیرنده متناهی غیر قطعی پذیرفته شود، ولی نتوانیم برای آن هیچ پذیرنده متناهی قطعی بیابیم. ولی اینطور نیست. رده‌های پذیرنده‌های متناهی قطعی و پذیرنده‌های متناهی غیر قطعی دارای قدرت یکسانی هستند : برای هر زبان پذیرفته شده بوسیله چند پذیرنده متناهی غیرقطعی، یک پذیرنده متناهی قطعی وجود دارد که همان زبان را بپذیرد.

این نتیجه آشکار نیست و بخصوص باید اثبات شود. استدلال، مشابه اکثر استدلال‌های این کتاب، استنباطی خواهد بود. این بدین معنی است که ما واقعاً راهی برای تبدیل هر پذیرنده متناهی غیر قطعی به پذیرنده متناهی قطعی معادل آن ارائه می‌دهیم. درک ساختار آن سخت نیست، به محض اینکه اصل آشکار شد نقطه شروعی برای استدلالی بسیار دقیق خواهد بود. دلیل منطقی ساخت در ادامه می‌آید. پس از اینکه یک پذیرنده متناهی غیر قطعی یک رشته w را خواند، ما ممکن نیست که دقیقاً بدانیم پذیرنده در چه حالتی خواهد بود، ولی می‌توانیم بگوییم که آن در یک حالت از مجموعه حالات ممکن $\{q_i, q_j, \ldots, q_k\}$ خواهد بود. یک پذیرنده متناهی قطعی معادل پس از خواندن همان رشته باید در



شکل ۲-۱۱

برخی حالت متناهی باشد. چگونه می توانیم این دو وضعیت را تطبیق دهیم؟ جواب یک حقه عالی است : حالات پذیرنده متناهی قطعی را با مجموعه‌ای از حالات برچسب گذاری نمایید بطوریکه پس از خواندن w، پذیرنده متناهی قطعی معادل در یک حالت تنهای برچسب گذاری شده بصورت $\{q_i, q_j, \ldots, q_k\}$ خواهد بود. از آنجایی که برای مجموعه‌ای با $|Q|$ حالت، دقیقا $2^{|Q|}$ زیرمجموعه وجود دارد، پذیرنده متناهی قطعی متناظر تعداد متناهی از حالات را خواهد داشت.

اکثر کار موجود در این ساختار پیشنهادی در تحلیل پذیرنده متناهی غیر قطعی برای بدست آوردن تطابق بین حالات و ورودی‌های ممکن است. قبل از توصیف رسمی این روش، اجازه دهید تا آن را با یک مثال ساده روشن کنیم.

مثال ۲-۱۲ : پذیرنده متناهی غیر قطعی در شکل ۲-۱۲ را به پذیرنده متناهی قطعی معادل تبدیل کنید. پذیرنده متناهی غیر قطعی با حالت q_0 شروع می‌شود، بطوریکه حالت اولیه پذیرنده متناهی قطعی با $\{q_0\}$ برچسب گذاری می‌شود. پس از خواندن یک a، پذیرنده متناهی غیر قطعی می‌تواند در حالت q_1، و یا با انجام انتقال λ در حالت q_2 باشد. بنابراین پذیرنده متناهی قطعی متناظر دارای حالتی با برچسب $\{q_1, q_2\}$ و انتقال

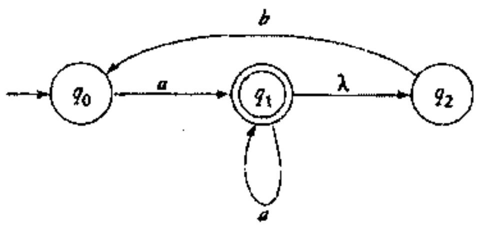
$$\delta(\{q_0\}, a) = \{q_1, q_2\}.$$

می‌باشد. در حالت q_0، پذیرنده متناهی غیر قطعی با ورودی b انتقال مشخصی ندارد، بنابراین

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

حالتی با برچسب $\varnothing$ نشان دهنده یک حرکت غیر ممکن برای پذیرنده متناهی غیر قطعی می‌باشد، و بنابراین به معنای عدم پذیرش رشته می‌باشد. در نتیجه، این حالت در پذیرنده متناهی قطعی باید یک حالت تله غیر نهایی باشد.

ما اکنون به پذیرنده متناهی قطعی حالت $\{q_1, q_2\}$ را معرفی می‌نماییم، بنابراین ما نیاز به یافتن انتقال‌های خروجی از این حالت داریم. به یاد داشته باشید که این حالت از ماشین متناهی قطعی مطابق با دو حالت ممکن از پذیرنده متناهی غیر قطعی است، بنابراین باید مجدداً به پذیرنده متناهی غیر قطعی مراجعه کنیم. اگر پذیرنده متناهی غیر قطعی در حالت q_1 باشد و یک a را بخواند، می تواند به q_1 برود. علاوه بر این



شکل ۲-۱۲

، پذیرنده متناهی غیر قطعی می‌تواند با انجام انتقال λ از q_1 به q_2 برود. اگر پذیرنده متناهی غیر قطعی برای همان ورودی در حالت q_2 باشد، آنگاه انتقال مشخصی وجود ندارد. بنابراین

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

بطور مشابه،

$$\delta(\{q_1, q_2\}, b) = \{q_0\}.$$

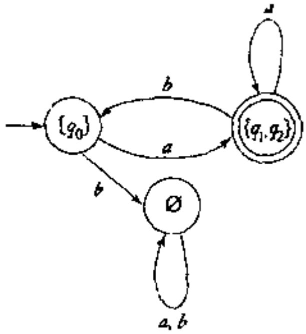
در این نقطه، برای هر حالت همه انتقال‌ها تعریف شده است. نتیجه، که در شکل ۲-۱۳ نشان داده شده است، یک پذیرنده متناهی قطعی معادل با پذیرنده متناهی غیر قطعی است که ما از آن شروع نموده‌ایم. پذیرنده متناهی غیر قطعی در شکل ۲-۱۲ هر رشته‌ای که برای آن $\delta^*(q_0, w)$ شامل q_1 باشد می‌پذیرد. برای پذیرنده متناهی قطعی متناظر جهت پذیرش هر رشته مانند w، هر حالتی که برچسب آن شامل q_1 باشد، باید یک حالت نهایی شود.

قضیه ۲-۲: فرض کنید L زبان پذیرفته شده توسط یک پذیرنده متناهی غیر قطعی $M_N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ باشد. در اینصورت یک پذیرنده متناهی قطعی $M_D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$ وجود دارد بطوریکه $L = L(M_D)$.

اثبات : ماشین M_N داده شده است، از رویه *تبدیل پذیرنده متناهی غیر قطعی به پذیرنده متناهی قطعی* زیر جهت ساخت گراف انتقال G_D برای M_D استفاده می‌کنیم. برای درک این ساخت، به یاد داشته باشید که G_D باید دارای ویژگی‌های خاصی باشد. هر راس باید دقیقاً دارای $|\Sigma|$ یال خروجی باشد، که هر یک با عنصر متفاوتی از Σ برچسب گذاری شده‌اند. در حین ساخت، برخی یال‌ها ممکن است حذف شوند، ولی رویه آنقدر ادامه می‌یابد تا همگی در آنجا قرار گیرند.

رویه : تبدیل پذیرنده متناهی غیر قطعی به پذیرنده متناهی قطعی

۱- یک گراف G_D با راس $\{q_0\}$ بسازید. این راس را به عنوان راس ابتدایی شناسایی کنید.



شکل ۲-۱۳

۲- مراحل زیر را تکرار کنید تا هیچ یالی کم نباشد.

هر راس $\{q_i, q_j, \ldots, q_k\}$ از G_D که هیچ یال خروجی برای برخی $a \in \Sigma$ ندارد در نظر بگیرید.

$\delta^*(q_i, a), \delta^*(q_j, a), \ldots, \delta^*(q_k, a)$ را محاسبه کنید.

سپس اجتماع همه این δ^* ها تشکیل دهید که منجر به مجموعه $\{q_l, q_m, \ldots, q_n\}$ می‌شود.

یک راس برای G_D با برچسب $\{q_l, q_m, \ldots, q_n\}$ ایجاد کنید البته اگر قبلاً بوجود نیامده نباشد.

به G_D یک یال از $\{q_i, q_j, \ldots, q_k\}$ به $\{q_l, q_m, \ldots, q_n\}$ اضافه کنید و آن را با a برچسب گذاری نمایید.

۳- هر حالت از G_D که برچسب آن شامل هر $q_f \in F_N$ باشد به عنوان یک راس نهایی شناسایی کنید.

۴- اگر M_N ورودی λ را می‌پذیرد، راس $\{q_0\}$ در G_D را نیز به عنوان یک راس نهایی بسازید.

واضح است که این رویه همیشه خاتمه می‌یابد. هر گذر حلقه در مرحله ۲ یک یال به G_D اضافه می کند. ولی G_D حداکثر دارای $2^{|Q_N|}|\Sigma|$ یال می باشد، بطوریکه حلقه سرانجام متوقف می‌شود. برای نمایش اینکه ساختار نیز جواب صحیح می‌دهد، ما بوسیله استقرا روی طول رشته ورودی استدلال می‌کنیم.

فرض کنید که برای هر v با طول کوچکتر یا مساوی n، وجود راهی با برچسب v از q_0 به q_i در G_N، ایجاب کند که راهی با برچسب v از $\{q_0\}$ به یک حالت $Q_i = \{\ldots, q_i, \ldots\}$ در G_D وجود داشته باشد. اکنون هر $w = va$ را در نظر بگیرید و به راهی در G_N با برچسب w از q_0 به q_l بنگرید. در اینصورت باید راهی با برچسب v از q_0 به q_i و یالی (یا دنباله‌ای از یال‌ها) با برچسب a از q_i به q_l وجود داشته باشد. برطبق فرض استقرا، در G_D راهی با برچسب v از $\{q_0\}$ به Q_i وجود خواهد داشت. ولی برطبق ساخت، یالی از Q_i به یک حالت وجود خواهد داشت که برچسب آن شامل q_l باشد. بنابراین فرض استقرا برای همه رشته‌ها با طول $n+1$ برقرار است. همچنانچه این مطلب برای $n = 1$ درست است، برای همه n ها نیز درست خواهد بود. در نتیجه هرگاه $\delta_N^*(q_0, w)$ شامل یک

www.prozhe.com

حالت نهایی q_f باشد، برچسب $\delta_D^*(q_0,w)$ نیز شامل آن خواهد بود. برای تکمیل اثبات، استدلال را معکوس می‌کنیم تا نشان دهیم که اگر برچسب $\delta_D^*(q_0,w)$ شامل q_f باشد، آنگاه $\delta_N^*(q_0,w)$ نیز باید شامل آن باشد. ■

استدلال‌های این اثبات اگرچه صحیح و تا اندازه‌ای مختصر هستند، فقط مراحل اصلی را نمایش می‌دهند. ما در بقیه این کتاب این کار را با تاکید روی ایده‌های اساسی در یک اثبات و حذف جزئیات فرعی که خودتان ممکن است بخواهید تکمیل کنید، دنبال می‌نماییم.

ساختار اثبات بالا خسته کننده ولی مهم است. اجازه دهید مثال دیگری بزنیم تا مطمئن شویم که همه مراحل را درک کرده‌ایم.

مثال ۲-۱۳ : پذیرنده متناهی غیر قطعی در شکل ۲-۱۴ را به یک ماشین قطعی معادل تبدیل کنید. از آنجا که $\delta_N(q_0,0)=\{q_0,q_1\}$ ، حالت $\{q_0,q_1\}$ را در G_D معرفی کرده و یالی با برچسب ۰ بین $\{q_0\}$ و $\{q_0,q_1\}$ اضافه می‌کنیم. به همین ترتیب، در نظر گرفتن $\delta_N(q_0,1)=\{q_1\}$ حالت جدید $\{q_1\}$ و یالی با برچسب ۱ بین آن و $\{q_0\}$ را به ما معرفی می‌کند.

اکنون تعدادی از یال‌ها کم است، بنابراین با استفاده از ساختار قضیه ۲-۲ ادامه می‌دهیم. با داشتن $a=۰, i=۰, j=۱$، محاسبه می‌کنیم

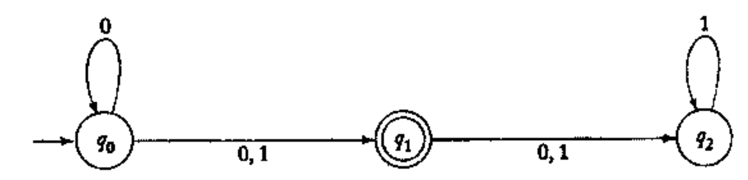
$$\delta_N^*(q_0,0)\cup\delta_N^*(q_1,0)=\{q_0,q_1,q_2\}.$$

این به ما حالت جدید $\{q_0,q_1,q_2\}$ و انتقال

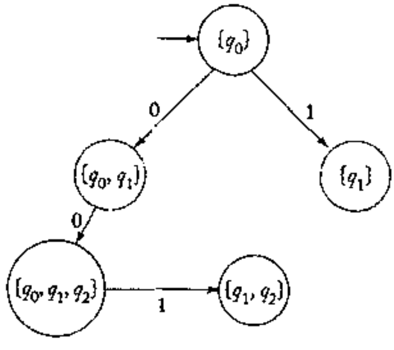
$$\delta_D(\{q_0,q_1\},0)=\{q_0,q_1,q_2\}$$

را ارائه می دهد. سپس، با استفاده از $a=1,\ \ i=0,\ \ j=1,\ \ k=2$،

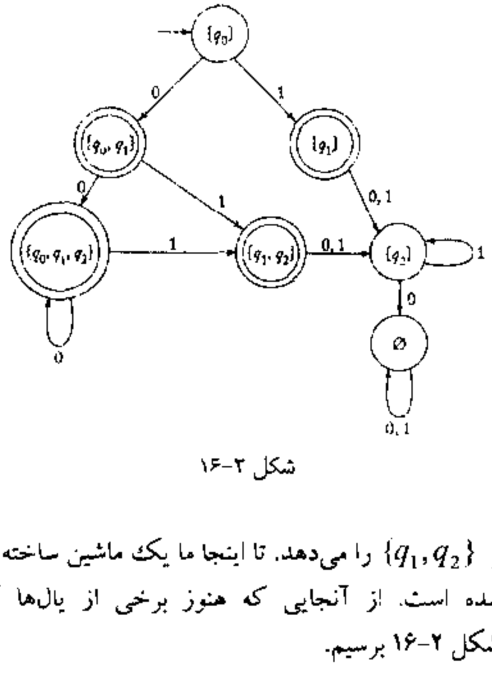
$$\delta_N^*(q_0,1)\cup\delta_N^*(q_1,1)\cup\delta_N^*(q_2,1)=\{q_1,q_2\}$$



شکل ۲-۱۴



شکل ۲-۱۵



شکل ۲-۱۶

که ضرورت معرفی حالت دیگر $\{q_1, q_2\}$ را می‌دهد. تا اینجا ما یک ماشین ساخته شده جزئی داریم که در شکل ۲-۱۵ نشان داده شده است. از آنجایی که هنوز برخی از یال‌ها کم هستند، ما ادامه می‌دهیم تا به راه حل کامل در شکل ۲-۱۶ برسیم.

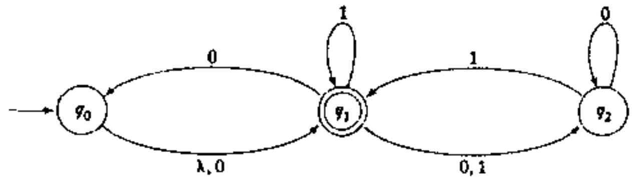
نتیجه مهمی که می توانیم از قضیه ۲-۲ بدست آوریم آن است که هر زبان پذیرفته شده بوسیله یک پذیرنده متناهی غیر قطعی، منظم است.

☐ تمرین‌ها

۱- با استفاده از ساختار قضیه ۲-۲ برای تبدیل پذیرنده متناهی غیر قطعی در شکل ۲-۱۰ به یک پذیرنده متناهی قطعی استفاده کنید. آیا می توانید جواب درست ساده‌تری پیدا کنید؟

۲- پذیرنده متناهی غیر قطعی در تمرین ۱۱، بخش ۲-۲ را به یک پذیرنده متناهی قطعی معادل تبدیل کنید.

۳- پذیرنده متناهی غیر قطعی زیر را به یک پذیرنده متناهی قطعی معادل تبدیل کنید.



۴- استدلال مربوط به اثبات قضیه ۲-۲ را با دقت کامل کنید. به همراه جزئیات نشان دهید که اگر برچسب $\delta_D^*(q_0, w)$ شامل q_f باشد، آنگاه $\delta_N^*(q_0, w)$ نیز شامل q_f است.

۵- آیا درست است که برای هر پذیرنده متناهی غیر قطعی $M = (Q, \Sigma, \delta, q_0, F)$ مکمل $L(M)$ برابر مجموعه $\{w \in \Sigma^* : \delta^*(q_0, w) \bigcap F = \varnothing\}$ است؟ اگر چنین است، آن را اثبات کنید. در غیر اینصورت، یک مثال نقض ارائه دهید.

۶- آیا درست است که برای هر پذیرنده متناهی غیر قطعی $M = (Q, \Sigma, \delta, q_0, F)$ مکمل $L(M)$ برابر مجموعه $\{w \in \Sigma^* : \delta^*(q_0, w) \bigcap (Q - F) \neq \varnothing\}$ است؟ اگر چنین است، آن را اثبات کنید. در غیر اینصورت، یک مثال نقض ارائه دهید.

۷- ثابت کنید برای هر پذیرنده متناهی غیر قطعی با تعداد دلخواه حالات نهایی، یک پذیرنده متناهی غیر قطعی معادل با فقط یک حالت نهایی وجود دارد. آیا می‌توانیم ادعای مشابهی برای پذیرنده های متناهی قطعی داشته باشیم؟

۸- یک پذیرنده متناهی غیر قطعی بدون انتقال λ و با تنها یک حالت نهایی بیابید که مجموعه $\{a\} \bigcup \{b^n : n \geq 1\}$ را بپذیرد.

۹- فرض کنید L زبان منظمی باشد که شامل λ نیست. نشان دهید که یک پذیرنده متناهی غیر قطعی بدون انتقال λ و با تنها یک حالت نهایی وجود دارد که L را می‌پذیرد. *

۱۰- یک پذیرنده متناهی قطعی با چندین حالت اولیه به روشی مشابه با پذیرنده متناهی غیر قطعی مربوط به تمرین ۱۷، بخش ۲-۲ تعریف کنید. آیا همیشه یک پذیرنده متناهی قطعی معادل با تنها یک حالت اولیه وجود دارد؟

۱۱- ثابت کنید که همه زبان های متناهی، منظم هستند.

۱۲- نشان دهید که اگر L منظم باشد، آنگاه L^R نیز منظم است.

۱۳- یک توصیف ساده متنی از زبان پذیرفته شده توسط پذیرنده متناهی قطعی در شکل ۲-۱۶ ارائه کنید. از این توصیف برای یافتن پذیرنده متناهی قطعی دیگری معادل با پذیرنده متناهی قطعی داده شده ولی با تعداد حالات کمتر استفاده نمایید.

۱۴- فرض کنید L هر زبانی باشد. $even(w)$ را به عنوان رشته بدست آمده بوسیله استخراج حروف با موقعیت‌های دارای شماره زوج از w تعریف کنید. یعنی، اگر

$$w = a_1 a_2 a_3 a_4 \ldots,$$

آنگاه

$$even(w) = a_2 a_4 \ldots$$

مطابق با این تعریف، ما می‌توانیم زبان زیر را تعریف کنیم

$$even(L) = \{even(w) : w \in L\}.$$

ثابت کنید که اگر L منظم باشد، آنگاه $even(L)$ نیز منظم است.

۱۵- از روی زبان L می‌توانیم زبان جدید $chop2(L)$ را بوسیله حذف دو نماد سمت چپ از هر رشته‌ای در L ایجاد کنیم. بخصوص،

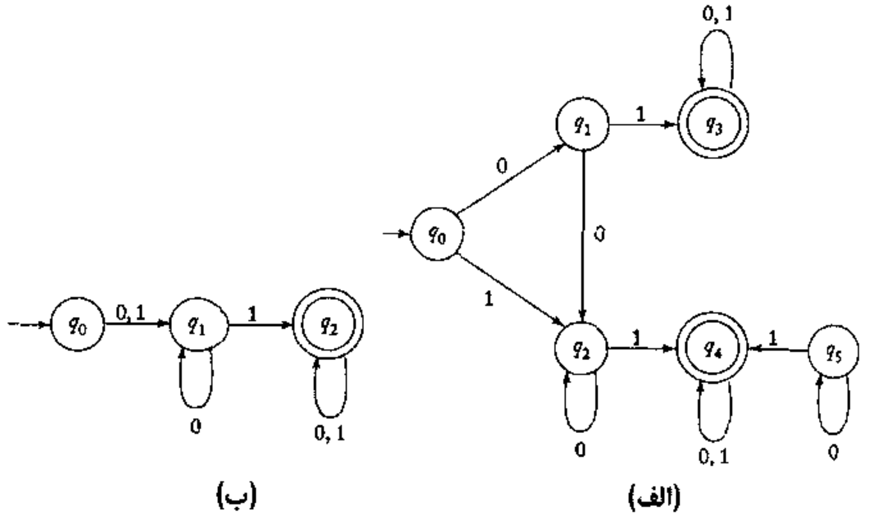
$$chop2(L) = \{w : vw \in L, \quad with \quad |v| = 2\}.$$

نشان دهید که اگر L منظم باشد، آنگاه $chop2(L)$ نیز منظم است.

۲-۴ کاهش تعداد حالات در ماشین‌های متناهی *

هر پذیرنده متناهی قطعی زبان منحصر بفردی را تعریف می‌کند، ولی عکس آن درست نیست. برای یک زبان داده شده، پذیرنده‌های متناهی قطعی بسیاری وجود دارند که آن را می‌پذیرند. ممکن است تفاوت قابل توجهی در تعداد حالات چنین ماشین‌های معادل وجود داشته باشد. بر حسب مسائلی که ما بعدا در نظر خواهیم گرفت، همه راه حل‌ها بصورت یکسان رضایت بخش هستند، ولی اگر نتایج در عمل بکار روند، ممکن است دلایلی برای ترجیح یکی بر دیگری موجود باشد.

مثال ۲-۱۴ : دو پذیرنده متناهی قطعی نشان داده شده در شکل (a) ۲-۱۷ و (b)۲-۱۷ معادل هستند، همچنان که چند رشته آزمایشی سریعاً این موضوع را آشکار می‌کند. به چند ویژگی آشکارای غیر ضروری در شکل (a)۲-۱۷ توجه می‌کنیم. حالت q_5 مطلقاً هیچ نقشی در ماشین ایفا نمی‌کند، زیرا هرگز از حالت اولیه q_0 نمی‌توان به آن رسید. چنین حالتی غیر قابل دسترس است، و می‌تواند (به همراه همه انتقال‌های وابسته به آن) حذف شود، بدون اینکه تاثیری روی زبان پذیرفته شده توسط ماشین داشته باشد. ولی حتی پس از حذف q_5، اولین ماشین دارای قسمت‌های زاید است. حالات قابل دسترس در نتیجه اولین حرکت $\delta(q_0,0)$ عیناً همان حالات قابل دسترس از اولین حرکت $\delta(q_0,1)$ می‌باشد. ماشین دوم این دو انتخاب را با هم ترکیب می‌کند.



شکل ۲-۱۷

از نقطه نظرصرفاً تئوری، دلیلی برای ترجیح ماشین شکل (b)۲-۱۷ بر ماشین شکل (a)۲-۱۷ وجود ندارد. به هر حال، از نظر سادگی، آشکارا ماشین دوم ارجحیت دارد. نمایش یک ماشین به منظور محاسبه نیاز به فضایی متناسب با تعداد حالات دارد. برای کارآیی حافظه، کاهش تعداد حالات تا حد امکان مطلوب است. ما اکنون الگوریتمی را توصیف می‌کنیم که این کار را انجام می‌دهد.

تعریف ۲-۸: دو حالت p و q از یک پذیرنده متناهی قطعی را نامتمایز گویند اگر

$$\delta^*(p,w) \in F \Rightarrow \delta^*(q,w) \in F,$$

و

$$\delta^*(p,w) \notin F \Rightarrow \delta^*(q,w) \notin F,$$

برای همه $w \in \Sigma^*$. از طرف دیگر، اگر رشته ای مانند $w \in \Sigma^*$ وجود داشته باشد بطوریکه

$$\delta^*(p,w) \in F, \quad \delta^*(q,w) \notin F.$$

یا بر عکس، آنگاه حالات p و q را **متمایز** بواسطه رشته w گویند.

واضح است که دو حالت یا نامتمایز و یا متمایز هستند. نامتمایز بودن دارای ویژگی یک رابطه هم ارزی است : اگر p و q نامتمایز باشند و اگر q و r نیز نامتمایز باشند، آنگاه p و r نیز نامتمایز بوده، و هر سه وضعیت نامتمایز هستند.

یک روش برای کاهش حالات یک پذیرنده متناهی قطعی بر اساس یافتن و ترکیب حالات نامتمایز می‌باشد. ابتدا روشی برای یافتن زوج‌هایی از حالات متمایز توصیف می‌کنیم.

رویه : علامت گذاری

۱- همه حالات غیر قابل دسترسی را حذف کنید. این کار می‌تواند با برشمردن همه مسیرهای ساده از گراف پذیرنده متناهی قطعی که از حالت اولیه آغاز می‌شوند، انجام شود. هر حالتی که در بخشی از هیچ از این مسیرها نیست، غیر قابل دسترسی است.

۲- همه زوج‌های حالات (p,q) را در نظر بگیرید. اگر $p \in F$ و $q \notin F$ و یا بر عکس، زوج (p,q) را به عنوان متمایز علامت بزنید.

۳- مرحله زیر را آنقدر تکرار کنید تا زوج‌هایی که قبلاً دارای علامت نبودند، علامت گذاری شوند.

برای همه زوج‌های (p,q) و همه $a \in \Sigma$، $\delta(p,a) = P_a$ و $\delta(q,a) = q_a$ را محاسبه نمایید. اگر زوج‌های (p_a, q_a) به عنوان متمایز علامت گذاری شده باشند، آنگاه (p,q) را به عنوان متمایز علامت گذاری نمایید.

ما ادعا می‌کنیم که این رویه الگوریتمی را برای یافتن همه زوج‌های متمایز بوجود می‌آورد.

قضیه ۲-۳: رویه علامت گذاری، به هر پذیرنده متناهی قطعی $M = (Q, \Sigma, \delta, q_0, F)$ اعمال شود، خاتمه یافته و همه زوج‌های حالات متمایز را تعیین می‌کند.

اثبات : آشکارا، رویه خاتمه می‌یابد، زیرا فقط تعداد متناهی از زوج‌ها وجود دارند که می‌توانند علامت گذاری شوند. همچنین آسان است ببینیم که حالات هر زوج علامت گذاری شده، متمایز هستند. تنها ادعایی که به شرح نیاز دارد آن است که این رویه همه زوج‌های متمایز را می‌یابد.

ابتدا توجه کنید که حالات q_i و q_j بواسطه رشته‌ای با طول n متمایز هستند اگر و فقط اگر انتقال‌های زیر وجود داشته باشند.

$$\delta(q_i, a) = q_k \tag{۲-۵}$$

و

$$\delta(q_j, a) = q_l, \tag{۲-۶}$$

برای یک $a \in \Sigma$، که q_k و q_l بواسطه رشته‌ای با طول $n-1$ متمایز هستند. ما ابتدا از این استفاده می‌کنیم تا نشان دهیم که در تکمیل nامین گذر در حلقه مرحله ۳، همه حالات متمایز بواسطه رشته‌هایی با طول n یا کمتر علامت گذاری شده‌اند. در مرحله ۲، همه زوج‌های نامتمایز بواسطه λ را علامت گذاری می‌نماییم، بنابراین ما پایه‌ای با $n = 0$ برای استقرا داریم. اکنون فرض می‌کنیم که ادعا برای همه $i = 0,1,\ldots,n-1$ درست است. با این فرض استقرا، در شروع nامین گذر حلقه، همه حالات متمایز بواسطه رشته‌هایی با طول حداکثر $n-1$ علامت گذاری شده‌اند. بدلیل روابط (۲-۵) و (۲-۶) بالا، در انتهای این گذر، همه حالات متمایز بواسطه رشته‌هایی با طول حداکثر n علامت گذاری

خواهند شد. سپس بوسیله استقرا می‌توانیم ادعا نماییم که برای هر n، در تکمیل nامین گذر، همه زوج‌های متمایز بواسطه رشته‌هایی با طول n یا کمتر علامت گذاری شده‌اند.

برای نمایش اینکه این رویه همه حالات متمایز را علامت گذاری می‌کند، فرض کنید که حلقه پس از n گذر خاتمه یابد. این بدین معنی است که در حین nامین گذر، هیچ حالات جدیدی علامت گذاری نمی‌شوند. از روی روابط (۲-۵) و (۲-۶) در می‌یابیم که هیچ حالات متمایزی بواسطه رشته‌ای با طول n نمی‌تواند موجود باشد، ولی نه متمایز بواسطه رشته‌ای کوتاه‌تر. ولی اگر هیچ حالات متمایزی بواسطه رشته‌هایی با طول فقط n وجود نداشته باشد، نمی تواند حالاتی متمایز بواسطه رشته‌هایی با طول فقط $n+1$ و مانند آن وجود داشته باشد. به عنوان نتیجه، هنگامی که حلقه خاتمه می‌یابد، همه زوج‌های متمایز علامت گذاری شده‌اند. ■

پس از اجرای الگوریتم علامت گذاری، از نتایج آن برای افراز مجموعه حالت Q از پذیرنده متناهی قطعی به زیر مجموعه‌های مجزای $\{q_i, q_j, \ldots, q_k\}, \{q_l, q_m, \ldots, q_n\}, \ldots$ استفاده می‌کنیم بطوریکه هر $q \in Q$ در دقیقاً یکی از این زیر مجموعه‌ها ظاهر می‌شود که عناصر در هر زیر مجموعه نامتمایز هستند، و هر دو عنصر از زیر مجموعه‌های متفاوت، متمایز هستند. با استفاده از نتایج مشروح در تمرین ۱۱ در انتهای این بخش، می‌توان نشان داد که چنین افرازی می‌تواند همیشه یافت شود. از روی این زیر مجموعه‌ها ماشین کمینه را بوسیله رویه بعدی می‌سازیم.

رویه : کاهش

یک پذیرنده متناهی قطعی $M = (Q, \Sigma, \delta, q_0, F)$ داده شده است، یک پذیرنده متناهی قطعی کاهش یافته $\widehat{M} = (\widehat{Q}, \Sigma, \widehat{\delta}, \widehat{q}_0, \widehat{F})$ به شرح زیر می‌سازیم.

۱- از رویه علامت گذاری برای یافتن همه زوج‌های حالات متمایز استفاده می‌کنیم. سپس از روی این، مجموعه‌های همه حالات نامتمایز را بصورت $\{q_i, q_j, \ldots, q_k\}, \{q_l, q_m, \ldots, q_n\}$ و غیره چنانچه در بالا توصیف شد، می‌یابیم.

۲- برای هر مجموعه $\{q_i, q_j, \ldots, q_k\}$ از چنین حالات نامتمایز، حالتی با برچسب $ij\ldots k$ برای $\widehat{M}$ ایجاد کنید.

۳- برای هر قانون انتقال از M به شکل

$$\delta(q_r, a) = q_p,$$

مجموعه‌هایی که q_r و q_p به آنها تعلق دارند بیابید. اگر $q_r \in \{q_i, q_j, \ldots, q_k\}$ و $q_p \in \{q_l, q_m, \ldots, q_n\}$ باشد، قانون

$$\widehat{\delta}(ij\ldots k, a) = lm\ldots n.$$

را به $\widehat{\delta}$ اضافه کنید.

۴- حالت اولیه $\widehat{q}_0$ حالتی از $\widehat{M}$ است که برچسب آن شامل ۰ است.

۵- $\widehat{F}$ مجموعه همه حالاتی است که برچسب آنها شامل i باشد بطوریکه $q_i \in F$.

مثال ۲-۱۵ : ماشین نشان داده شده در شکل ۲-۱۸ را در نظر بگیرید.

در مرحله ۲، رویه علامت گذاری زوج‌های متمایز $(q_0,q_4),(q_1,q_4),(q_2,q_4)$ و (q_3,q_4) را شناسایی می‌نماید. در گذری در حین انجام حلقه مرحله۳، این رویه

$$\delta(q_1,1) = q_4$$

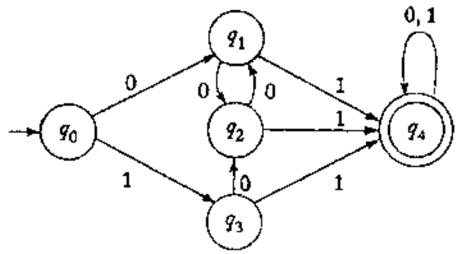
و

$$\delta(q_0,1) = q_3$$

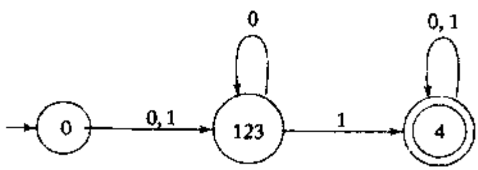
را محاسبه می‌کند.

از آنجایی که (q_3,q_4) یک زوج متمایز است، زوج (q_0,q_1) نیز علامت گذاری می‌شود. با ادامه این روش، سرانجام الگوریتم علامت گذاری زوج‌های

$$(q_0,q_1),(q_0,q_2),(q_0,q_3),(q_0,q_4),$$



شکل ۲-۱۸



شکل ۲-۱۹

$(q_1,q_4),(q_2,q_4)$ و (q_3,q_4) را به عنوان متمایز علامت گذاری می‌کند، و زوج‌های نامتمایز $(q_1,q_2),(q_1,q_3)$ و (q_2,q_3) را باقی می‌گذارد. بنابراین حالات q_1,q_2,q_3 همگی نامتمایز هستند، و همه حالات به مجموعه‌های $\{q_0\},\{q_1,q_2,q_3\}$ و $\{q_4\}$ افراز می‌شوند. اعمال مراحل ۲ و ۳ از رویه *کاهش*، به پذیرنده متناهی قطعی در شکل ۲-۱۹ منجر می‌شود.

www.prozhe.com

قضیه ۲-۴ : یک پذیرنده متناهی قطعی M داده شده است، کاربرد رویه کاهش منجر به پذیرنده متناهی قطعی دیگری به نام $\hat{M}$ می‌شود بطوریکه

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

بعلاوه، $\hat{M}$ کمینه است بدین معنا که هیچ پذیرنده متناهی قطعی دیگری با تعداد حالات کمتری وجود ندارد که $L(M)$ را بپذیرد.

اثبات : دو بخش وجود دارد. اول اینکه نشان دهیم که پذیرنده متناهی قطعی ایجاد شده بوسیله رویه کاهش معادل با پذیرنده متناهی قطعی اولیه است. این کار نسبتا آسان است و می‌توانیم از استدلال‌های استقرائی مشابه با آنهایی که در اثبات معادل بودن پذیرنده‌های متناهی قطعی و پذیرنده‌های متناهی غیر قطعی به کار برده شد، استفاده کنیم. همه کاری که باید انجام دهیم آن است که نشان دهیم $\delta^*(q_i, w) = q_j$ اگر و فقط اگر برچسب $\hat{\delta}^*(q_i, w)$ به شکل $\ldots j \ldots$ باشد. این موضوع را به عنوان تمرین باقی می‌گذاریم.

بخش دوم، نمایش کمینه بودن $\hat{M}$ است که مشکل‌تر است. فرض کنید $\hat{M}$ دارای حالات $\{p_0, p_1, p_2, \ldots, p_m\}$ باشد که p_0 حالت اولیه است. فرض کنید که یک پذیرنده متناهی قطعی به نام M_1 با تابع انتقال δ_1 و حالت اولیه q_0 وجود دارد که معادل با M است، ولی دارای تعداد حالات کمتری است. از آنجایی که هیچ حالت غیر قابل دسترسی در $\hat{M}$ وجود ندارد، باید رشته‌های مجزای $w_1, w_2, \ldots, w_m$ وجود داشته باشند بطوریکه

$$\hat{\delta}^*(p_0, w_i) = p_i, i = 1,2,\ldots,m.$$

ولی از آنجایی که M_1 تعداد حالات کمتری نسبت به M دارد، باید حداقل دو تا از این رشته‌ها مانند w_k و w_l وجود داشته باشند، بطوریکه

$$\delta_1^*(q_0, w_k) = \delta_1^*(q_0, w_l).$$

از آنجایی که p_k و p_l متمایز هستند، باید رشته‌ای مانند x موجود باشد بطوریکه $\hat{\delta}^*(p_0, w_k x) = \hat{\delta}^*(p_k, x)$ یک حالت نهایی است، و $\hat{\delta}^*(q_0, w_l x) = \hat{\delta}^*(p_l, x)$ یک حالت غیر نهایی است (یا بر عکس). به عبارت دیگر، $w_k x$ بوسیله $\hat{M}$ پذیرفته می‌شود و $w_l x$ پذیرفته نمی‌شود. ولی توجه کنید که

$$\begin{aligned}
\delta_1^*(q_0, w_k x) &= \delta_1^*(\delta_1^*(q_0, w_k), x) \\
&= \delta_1^*(\delta_1^*(q_0, w_l), x) \\
&= \delta_1^*(q_0, w_l x).
\end{aligned}$$

بنابراین، M_1 یا هر دوی $w_k x$ و $w_l x$ را می‌پذیرد یا هر دو را نمی‌پذیرد، که با فرض اینکه $\hat{M}$ و M_1 معادلند تناقض دارد. این تناقض اثبات می‌کند که M_1 نمی‌تواند وجود داشته باشد. ■

تمرین‌ها

۱- تعداد حالات پذیرنده متناهی قطعی در شکل ۲-۱۶ را کمینه سازید.

۲- پذیرنده متناهی قطعی کمینه برای زبان‌های زیر بیابید. در هر مورد اثبات کنید که نتیجه، کمینه است.

الف) $L = \{a^n b^m : n \ge 2, m \ge 1\}$

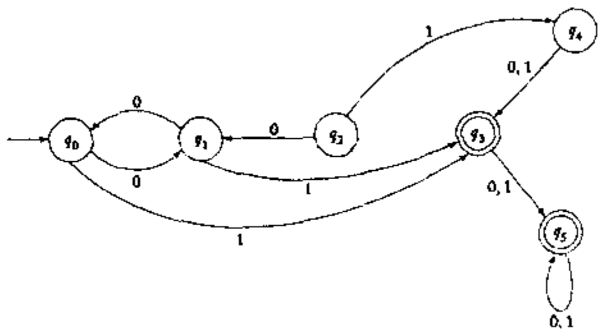
ب) $L = \{a^n b : n \ge 0\} \bigcup \{b^n a : n \ge 1\}$

ج) $L = \{a^n : n \ge 0, m \ne 3\}$

د) $L = \{ a^n : n \ne 2 \ and \ n \ne 4 \}$

۳- نشان دهید که ماشین تولید شده بوسیله رویه *کاهش*، قطعی است.

۴- حالات پذیرنده متناهی قطعی نشان داده شده در دیاگرام زیر را کمینه کنید.



۵- نشان دهید اگر L یک زبان غیر تهی باشد بطوریکه هر w در L دارای طول حداقل n باشد، آنگاه هر پذیرنده متناهی قطعی پذیرنده L باید حداقل دارای $n+1$ حالت باشد.

۶- فرضیه‌ای که در ادامه آمده را ثابت و یا رد کنید. اگر $M = (Q, \Sigma, \delta, q_0, F)$ یک پذیرنده متناهی قطعی کمینه برای زبان منظم L باشد، آنگاه $\hat{M} = (Q, \Sigma, \delta, q_0, Q - F)$ یک پذیرنده متناهی قطعی کمینه برای $\overline{L}$ خواهد بود.

۷- نشان دهید که نامتمایز بودن یک رابطه هم ارزی است ولی متمایز بودن نیست.

۸- مراحل صریح اثبات پیشنهادی از بخش اول قضیه ۲-۴ که در آن $\widehat{M}$ معادل با پذیرنده متناهی قطعی اولیه است را نشان دهید.

۹- یک برنامه کامپیوتری بنویسید که یک پذیرنده متناهی قطعی کمینه برای هر پذیرنده متناهی قطعی داده شده، تولید کند. **

۱۰- ثابت کنید که اگر حالات q_a و q_b نامتمایز باشند، و اگر q_a و q_c متمایز باشند، آنگاه q_b و q_c باید متمایز باشند.

۱۱- فرآیند زیر را در نظر بگیرید، که پس از تکمیل رویه *علامت گذاری* انجام گیرد. با یکی از حالات مانند q_0 شروع کنید. همه حالاتی که متمایز از q_0 علامت گذاری نشده‌اند را در یک مجموعه هم ارز با q_0 قرار دهید. سپس حالت دیگری را بگیرید، که در مجموعه هم ارزی ذکر شده نباشد، و همین کار را انجام دهید. این کار را آنقدر تکرار کنید تا دیگر حالتی باقی نماند. این پیشنهاد را برای ساخت یک الگوریتم تدوین کنید، و ثابت کنید که این الگوریتم واقعاً مجموعه حالت اولیه را به مجموعه‌های هم ارزی افراز می‌کند.

۳ زبان های منظم و گرامرهای منظم

طبق تعریف ما، زبانی منظم است که برای آن یک پذیرنده متناهی وجود داشته باشد. بنابراین هر زبان منظم می‌تواند توسط تعدادی پذیرنده متناهی قطعی یا پذیرنده متناهی غیرقطعی توصیف شود. چنین توصیفی می‌تواند بسیار مفید باشد، برای مثال، اگر بخواهیم منطق تصمیم گیری در خصوص تعلق یک رشته داده شده به یک زبان خاص را نشان دهیم. ولی در بسیاری از موارد، به روشهای دقیق‌تری از توصیف زبانهای منظم نیازمندیم. در این فصل، به روشهای دیگر ارائه زبانهای منظم می‌نگریم. این روشها دارای کاربردهای علمی مهمی هستند، موضوعی که در برخی از مثالها و تمرین‌ها به آن خواهیم پرداخت.

۳-۱ عبارات منظم

یکی از راه‌های توصیف زبانهای منظم استفاده از علائم **عبارات منظم** می‌باشد. این علائم، شامل ترکیبی از رشته های نمادهایی از یک الفبای Σ ، پرانتزها و عملگرهای +، . ، و * است. ساده‌ترین مورد، زبان $\{a\}$ می باشد، که بصورت عبارت منظم a نشان داده می‌شود. یک زبان پیچیده تر، زبان $\{a,b,c\}$ است که با استفاده از + برای نشان دادن اجتماع، عبارت منظم $a+b+c$ را داریم. از . برای اتصال و به روش مشابه از * برای بستار ستاره‌ای استفاده می‌کنیم. عبارت $(a+b\cdot c)^*$ نمایانگر بستار ستاره‌ای $\{a\}\cup\{bc\}$ است، که همان زبان $\{\lambda, a, bc, aa, abc, bca, bcbc, aaa, aabc, \ldots\}$ است.

تعریف رسمی یک عبارت منظم

ما می توانیم عبارات منظم را با اعمال مکرر قوانین بازگشتی خاصی از روی اجزای سازنده ابتدایی بسازیم. این روش شبیه به روش ساخت عبارات آشنای ریاضی است.

تعریف ۳-۱ : بفرض الفبای Σ داده شده باشد. آنگاه

۱- $\varnothing$، λ و $a \in \Sigma$ همگی عبارات منظم هستند که آنها را **عبارات منظم ابتدایی** گویند.
۲- اگر r_1 و r_2 عبارت منظم باشند، آنگاه $r_1 + r_2$، $r_1 \cdot r_2$، r_1^*، و (r_1) نیز عبارات منظم هستند.
۳- رشته ای عبارت منظم است اگر و فقط اگر بتوان آن را از عبارات منظم ابتدایی به وسیله اعمال دفعات متناهی قانون (۲) ایجاد کرد.

مثال ۳-۱: برای $\Sigma = \{a, b, c\}$، رشته

$$(a + b.c)^* \cdot (c + \varnothing)$$

یک عبارت منظم است، زیرا بوسیله اعمال قوانین بالا ایجاد شده است. برای مثال، اگر $r_1 = c$ و $r_2 = \varnothing$ باشد،
در می‌یابیم که $c + \varnothing$ و $(c + \varnothing)$ نیز عبارات منظم هستند. با تکرار این، نهایتاً میتوانیم همه رشته را تولید کنیم. از سوی دیگر، $(a + b+)$ یک عبارات منظم نیست، زیرا به هیچ طریقی نمی‌توان آن را از عبارات منظم ابتدایی ساخت.

زبان‌های مرتبط با عبارات منظم

عبارات منظم می‌توانند برای توصیف برخی زبانهای ساده بکار روند. اگر r یک عبارت منظم باشد، $L(r)$ را زبان مرتبط با r گوییم. این زبان بصورت زیر تعریف می‌شود.

تعریف ۳-۲ : زبان $L(r)$ که با عبارت منظم r نشان داده می‌شود توسط قوانین زیر تعریف می‌شود.
۱- $\varnothing$ یک عبارت منظم است که نشان دهنده یک مجموعه تهی است،
۲- λ یک عبارت منظم است که نشان دهنده $\{\lambda\}$ است،
۳- برای هر $a \in \Sigma$، a یک عبارت منظم نشان دهنده $\{a\}$ است.
اگر r_1 و r_2 عبارات منظم باشند، آنگاه
۴- $L(r_1 + r_2) = L(r_1) \cup L(r_2)$،
۵- $L(r_1 \cdot r_2) = L(r_1)L(r_2)$،
۶- $L((r_1)) = L(r_1)$،
۷- $L(r_1^*) = (L(r_1))^*$.

چهار قانون آخر از این تعریف برای کاهش $L(r)$ به اجزای ساده‌تر بطور بازگشتی استفاده می‌شوند. سه قانون اول شرایط پایانی برای این بازگشت پذیری هستند. برای دیدن اینکه یک زبان چه عباراتی را نمایش میدهد، این قوانین را به طور مکرر اعمال می کنیم.

مثال ۳-۲ : زبان $L(a^* \cdot (a+b))$ را با نماد مجموعه‌ای نشان دهید.

$$L(a^* \cdot (a+b)) = L(a^*)L(a+b)$$
$$= (L(a))^*(L(a) \cup L(b))$$
$$= \{\lambda, a, aa, aaa, \ldots\}\{a,b\}$$
$$= \{a, aa, aaa, \ldots, b, ab, aab, \ldots\}$$

در اینجا یک مشکل برای قانونهای ۴ تا ۷ در تعریف ۳-۲ وجود دارد. آنها اگر r_1 و r_2 داده شده باشند، دقیقاً یک زبان را تعریف می‌کنند، ولی ممکن است در شکستن یک عبارت پیچیده به اجزایش برخی ابهامات وجود داشته باشد. برای مثال، عبارت منظم $a \cdot b + c$ را در نظر بگیرید. می‌توانیم ببینیم که این عبارت از $r_1 = a \cdot b$ و $r_2 = c$ تشکیل شده است. در این مورد، $L(a \cdot b + c) = \{ab, c\}$ می‌باشد. ولی در تعریف ۳-۲ هیچ مانعی برای در نظر گرفتن $r_1 = a$ و $r_2 = b + c$ وجود ندارد. در این حال نتیجه متفاوتی می گیریم، $L(a \cdot b + c) = \{ab, ac\}$. برای غلبه بر این موضوع، می‌توانیم همه عبارات را پرانتزگذاری کامل نماییم، ولی این روش، نتایج پیچیده‌ای را میدهد. به جای آن، از قراردادهای آشنای ریاضیات و زبانهای برنامه نویسی استفاده می‌کنیم. ما مجموعه‌ای از قوانین تقدم را برای ارزیابی وضع کنیم که در آن بستار ستاره‌ای بر اتصال و اتصال بر اجتماع تقدم دارد. .همچنین، نماد اتصال را می‌توان حذف کرد، یعنی می‌توانیم به جای $r_1 \cdot r_2$ بنویسیم $r_1 r_2$.

با کمی تمرین، می‌توانیم بسرعت ببینیم چه زبانی توسط یک عبارت منظم خاص نشان داده می‌شود.

مثال ۳-۳ : برای $\Sigma = \{a, b\}$ عبارت

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

منظم است که نمایانگر زبان

$$L(r) = \{a, bb, aa, abb, ba, bbb, \ldots\}$$

است. ما میتوانیم این را با در نظر گرفتن قسمتهای مختلف r ببینیم. اولین قسمت، $(a+b)^*$ نمایانگر هر رشته از a ها و b ها است. دومین قسمت، $(a+bb)$ نمایانگر یک a و یا دو b است. در نتیجه، $L(r)$ مجموعه همه رشته‌های روی $\{a, b\}$ است که به یک a یا یک bb ختم می‌شوند.

مثال ۳-۴ : عبارت

$$r = (aa)^*(bb)^*b$$

نمایانگر مجموعه‌ای از رشته‌ها با تعداد زوج a است که بوسیله تعداد فردی b دنبال می‌شود. یعنی

$$L(r) = \{a^{2n}b^{2m+1} : n \geq 0, m \geq 0\}.$$

رفتن از یک توصیف غیر رسمی یا نماد مجموعه‌ای به یک عبارت منظم، کمی مشکل‌تر می‌باشد.

مثال ۳-۵: برای $\Sigma = \{0,1\}$، یک عبارت منظم r بدهید به گونه‌ای که

$$L(r) = \{w \in \Sigma^* : \text{دارای حداقل یک زوج صفر متوالی باشد } w\}$$

یک شخص می‌تواند بوسیله استدلالی شبیه این به یک جواب برسد: هر رشته در $L(r)$ باید شامل 00 در جایی باشد، ولی اینکه قبل از آن یا بعد از آن چه بیاید کاملاً دلخواه است. هر رشته دلخواه روی $\{0,1\}$ را می‌توان بوسیله $(0+1)^*$ نشان داد. با قرار دادن این دیدگاه‌ها در کنار هم، ما به راه حل زیر می‌رسیم.

$$r = (0+1)^* 00 (0+1)^*.$$

مثال ۳-۶: یک عبارت منظم را برای زبان زیر بیابید.

$$L = \{w \in \{0,1\}^* : \text{دارای هیچ زوج صفر متوالی نباشد } w\}$$

اگرچه این مشابه مثال۳-۵ به نظر میرسد، اما ساخت جواب مشکلتر است. یک دیدگاه مفید آن است که هر گاه یک 0 ظاهر شود، باید فوراً بوسیله یک 1 دنبال شود. چنین زیر رشته‌ای ممکن است با تعداد دلخواهی از 1 ها آغاز و یا دنبال شود. این موضوع پیشنهاد می‌کند که جواب، شامل تکرار رشته‌ها به شکل $1...101...1$ باشد، یعنی زبان نشان داده شده توسط عبارت منظم $(1^*011^*)^*$. به هرحال، جواب هنوز کامل نیست، زیرا رشته‌های منتهی به ۰ یا رشته‌هایی که تنها شامل ۱ ها باشد به حساب آورده نشده است. با در نظر گرفتن این موارد خاص به جواب می‌رسیم.

$$r = (1^*011^*)^*(0+\lambda) + 1^*(0+\lambda).$$

اگر کمی متفاوت استدلال کنیم، ممکن است ما به جواب دیگری برسیم. اگر L را به عنوان تکرار رشته‌های 1 و 01 در نظر بگیریم، به عبارت کوتاهتری می‌رسیم.

$$r = (1+01)^*(0+\lambda)$$

اگر چه دو عبارت، متفاوت به نظر می‌رسند، هر دو جواب صحیح هستند، زیرا نمایانگر زبان یکسانی هستند. عموماً تعداد نا محدودی عبارت منظم برای هر زبان داده شده وجود دارد.

توجه کنید که این زبان مکمل زبان موجود در مثال ۳-۵ است. به هر حال، عبارات منظم خیلی شبیه نیستند و ارتباط نزدیکی بین زبانها را نشان نمی‌دهند.

مثال آخر علامت معادل بودن عبارات منظم را نشان میدهد. گوییم دو عبارت منظم باهم معادلند اگر زبان یکسانی را نشان دهند. یک شخص می‌تواند قوانین مختلفی را برای ساده سازی عبارات منظم وضع کند (تمرین ۱۸ در تمرینات این بخش را ببینید)، ولی بدلیل اینکه ما نیاز کمی برای چنین دستکاری هایی داریم، آن را دنبال نمی‌کنیم.

تمرین‌ها

۱- تمامی رشته‌های موجود در $L((a+b)^* b(a+ab)^*)$ با طول کمتر از 4 را بیابید.

۲- آیا عبارت $((0+1)(0+1)^*)^* 00(0+1)^*$ نمایانگر زبان موجود در مثال ۳-۵ است؟

۳- نشان دهید که $r = (1+01)^*(0+1^*)$ بر زبان موجود در مثال ۳-۶ دلالت دارد. دو عبارت معادل دیگر بیابید.

۴- یک عبارت منظم برای مجموعه $\{a^n b^m : (n+m)$ زوج است $\}$ بیابید.

۵- عبارات منظمی برای زبانهای زیر بدهید.

الف) $L_1 = \{a^n b^m, n \geq 4, m \leq 3\}$،

ب) $L_2 = \{a^n b^m : n < 4, m \leq 3\}$،

ج) مکمل زبان L_1،

د) مکمل زبان L_2.

۶- چه زبانهایی بوسیله عبارات $(\varnothing^*)^*$ و $a\varnothing$ نشان داده می‌شوند؟

۷- یک توصیف متنی ساده از زبان $L((aa)^* b(aa)^* + a(aa)^* ba(aa)^*)$ بدهید.

۸- یک عبارت منظم برای L^R ارائه دهید، جاییکه L زبان تمرین ۱ باشد.

۹- یک عبارت منظم برای $L = \{a^n b^m : n \geq 1, m \geq 1, nm \geq 3\}$ بدهید.

۱۰- یک عبارت منظم برای $L = \{ab^n w : n \geq 3, w \in \{a,b\}^+\}$ بیابید.

۱۱- یک عبارت منظم برای مکمل زبان مثال ۳-۴ بیابید.

۱۲- یک عبارت منظم برای $L = \{vwv : v, w \in \{a,b\}^*, |v| = 2\}$ بیابید.

۱۳- یک عبارت منظم برای زبان زیر بیابید.

$L = \{w \in \{0,1\}^* :$ دارای دقیقاً یک جفت صفر متوالی باشد $w\}$

۱۴- عبارات منظمی برای زبانهای زیر روی $\Sigma = \{a,b,c\}$ بدهید.

الف) همه رشته‌هایی که دقیقاً شامل یک a باشند،

ب) همه رشته‌هایی که بیش از سه a نداشته باشند،

ج) همه رشته‌هایی که از هر نماد در Σ حداقل یک رخداد را دارا باشند،

د) همه رشته‌هایی که شامل هیچ دوره‌ای از a ها با طول بیش از دو نباشند،

ه) همه رشته‌هایی که طول همه دوره‌های a های آن مضارب سه باشند. *

۱۵- عبارات منظمی را برای زبانهای زیر بر روی $\Sigma = \{0,1\}$ بنویسید.

الف) همه رشته‌هایی که به 01 ختم می‌شوند،

ب) همه رشته‌هایی که به 01 ختم نمی‌شوند،

ج) همه رشته‌هایی که شامل تعداد زوج از 0 ها باشند،

د) همه رشته هایی با حداکثر دو بار رخداد زیر رشته 00 (توجه کنید که تحت تفسیر معمول زیررشته ۰۰۰ دارای دو رخداد 00 می‌باشد)

ه) همه رشته‌هایی که شامل زیررشته ۱۰۱ نباشند. *

ه) همه رشته‌هایی که شامل زیررشته ۱۰۱ نباشند. ☆

۱۶- عبارات منظمی برای زبانهای زیر روی $\{a,b\}$ بیابید.

الف) $L = \{w : |w| \bmod 3 = 0\}$ ⓢ

ب) $L = \{w : n_a(w) \bmod 3 = 0\}$

ج) $L = \{w : n_a(w) \bmod 5 > 0\}$

۱۷- قسمت‌های الف، ب، و ج از تمرین ۱۶ را برای $\Sigma = \{a,b,c\}$ تکرار کنید.

۱۸- تعیین کنید آیا ادعاهای زیر برای همه عبارات منظم r_1 و r_2 درست هستند یا خیر. نماد $\equiv$ نمایانگر معادل بودن عبارات منظم می‌باشد بدین معنی که هر دو عبارت بر زبان یکسانی دلالت می‌کنند.

الف) $(r_1^*)^* \equiv r_1^*$،

ب) $r_1^*(r_1 + r_2)^* \equiv (r_1 + r_2)^*$،

ج) $(r_1 + r_2)^* \equiv (r_1^* r_2^*)^*$، ⓢ

د) $(r_1 r_2)^* \equiv r_1^* r_2^*$.

۱۹- یک روش عمومی ارائه کنید که بوسیله آن هر عبارت منظم r بتواند به $\hat{r}$ تغییر یابد، بطوریکه $(L(r))^R = L(\hat{r})$.

۲۰- با دقت تمام ثابت کنید که عبارات مثال۳-۶ واقعاً بر آن زبان مشخص شده دلالت دارند.

۲۱- در مورد عبارت منظم r که شامل λ یا $\varnothing$ نباشد، مجموعه‌ای از شرایط لازم و کافی ارائه دهید که r باید داشته باشد تا $L(r)$ نامتناهی باشد. ⓢ

۲۲- زبانهای صوری می توانند برای توصیف انواع اشکال دوبعدی استفاده شوند. زبانهای کد-زنجیره‌ای بر روی الفبای $\Sigma = \{u,d,r,l\}$ تعریف می‌شوند، جاییکه این نمادها بترتیب نمایانگر خطوط مستقیم به طول واحد در جهات بالا، پایین، راست، وچپ می‌باشند. یک مثال از این علائم $urdl$ می‌باشد که نمایانگر مربعی با اضلاعی به طول واحد است. تصاویری از اشکال نشان داده شده بوسیله عبارات $(rd)^*$، $(urddru)^*$، و $(ruldr)^*$ رسم کنید.

۲۳- در تمرین ۲۲ شرایط کافی روی عبارت برای اینکه شکل یک محیط بسته باشد یعنی نقاط ابتدایی و انتهایی یکسانی داشته باشد چیست؟ آیا این شرایط لازم هم می‌باشند؟ ⓢ

۲۴- یک پذیرنده متناهی غیر قطعی بیابید که زبان $L(aa^*(a+b))$ را بپذیرد.

۲۵- یک عبارت منظم بیابید که به همه رشته‌های بیتی که مقدارشان بصورت اعداد صحیح دودویی تفسیر شده، و بزرگتر یا مساوی ۴۰ باشند دلالت کند.

۲۶- یک عبارت منظم برای همه رشته‌های بیتی که با ۱ آغاز شده، و بعنوان یک عدد صحیح دودویی تفسیر شده، و مقادیرشان بین ۱۰ و ۳۰ نمی‌باشد بیابید.

۳-۲ ارتباط بین عبارات منظم و زبان‌های منظم

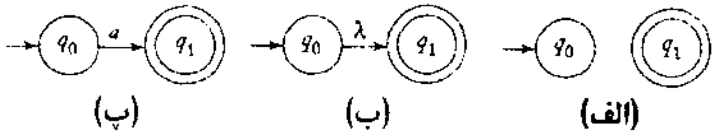
همانطور که از نام‌ها پیداست ارتباط بین عبارات منظم و زبانهای منظم، ارتباط نزدیکی است. این مفهوم اساساً یکسانند. برای هر زبان منظم یک عبارت منظم وجود دارد، و برای هر عبارت منظم یک منظم وجود دارد. ما این موضوع را در دو قسمت نشان می‌دهیم.

عبارات منظم بر زبان‌های منظم دلالت دارند

ابتدا نشان می‌دهیم که اگر r یک عبارت منظم باشد آنگاه $L(r)$ یک زبان منظم است. طبق تعریف زبانی منظم است که توسط یک پذیرنده متناهی قطعی پذیرفته شود. به دلیل معادل بودن پذیرنده متناهی قطعی و پذیرنده های متناهی غیر قطعی، یک زبان همچنین منظم است اگرتوسط یک پذ متناهی غیر قطعی پذیرفته شود. اکنون نشان می‌دهیم که اگر هر عبارت منظم r را داشته باشیم، می‌ یک پذیرنده متناهی غیر قطعی بسازیم که $L(r)$ را بپذیرد. ساختار آن بر تعریف بازگشتی $L(r)$ ا است. ما ابتدا یک ماشین خودکار ساده‌ای برای بخشهای (۱) ،(۲)، و(۳) از تعریف ۳-۲ در صفح می‌سازیم، سپس نشان می‌دهیم که آنها چگونه می‌توانند با هم ترکیب شوند تا بخشهای پیچیده تر (۵)، و(۷) را پیاده سازی نمایند.

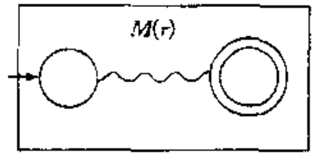
قضیه ۳-۱ : فرض کنید r یک عبارت منظم باشد. آنگاه پذیرنده متناهی غیر قطعی وجود که $L(r)$ را می پذیرد. در نتیجه، $L(r)$ یک زبان منظم است.

اثبات: ما با ماشینی آغاز می‌کنیم که عبارات منظم ساده $\varnothing$، λ، و $a \in \Sigma$ را می‌پذیرد عبارات بترتیب در شکل۳-۱، (a)، (b)، و (c) نشان داده شده‌اند. حال فرض کنید که ماشین $M(r_1)$ و $M(r_2)$ را داریم که بترتیب زبانهای نشان داده شده با عبارات منظم r_1 و r_2 رامی پذ نیازی به ساخت این ماشین‌ها نیست، ولی آنها را به اختصار مانند شکل۳-۲ نشان می‌دهیم. در این کلی، راس گراف در سمت چپ، حالت اولیه را نشان می‌دهد و رأس گراف در سمت راست، نهایی را نشان می‌دهد. در تمرین۷، بخش ۲-۳ ادعا کردیم که برای هر پذیرنده متناهی غیرقطعی، پذ متناهی غیر قطعی معادلی با یک وضعیت نهایی وجود دارد. بنابراین بدون از دست دادن چیزی می فرض کرد که فقط یک حالت نهایی وجود دارد. با داشتن ماشین‌های $M(r_1)$ و $M(r_2)$ که بدین نمایش داده شده‌اند، می‌توانیم ماشین‌هایی برای عبارات منظم $r_1 + r_2$ ، r_1r_2 ، و r_1^* بسازیم. ساخت شکل‌های ۳-۳ تا ۳-۵ نمایش داده شده‌اند. همچنانچه در شکل‌ها نشان داده شده است، حالات او نهایی حالتشان را از دست داده و بوسیله حالات اولیه و نهایی جدید جایگزین می‌شوند. با ردیف چندین مرحله، می‌توانیم ماشین-هایی برای عبارات منظم با پیچیدگی دلخواه بسازیم.

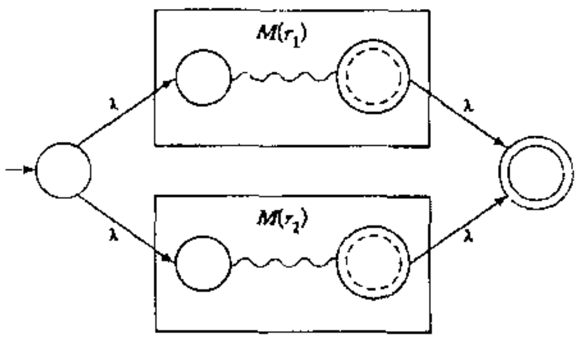


شکل ۳-۱ (الف) پذیرنده متناهی غیر قطعی که $\varnothing$ را می‌پذیرد.

(ب) پذیرنده متناهی غیر قطعی که $\{\lambda\}$ را می‌پذیرد. (پ) پذیرنده متناهی غیر قطعی که $\{a\}$ را می‌پذیرد.

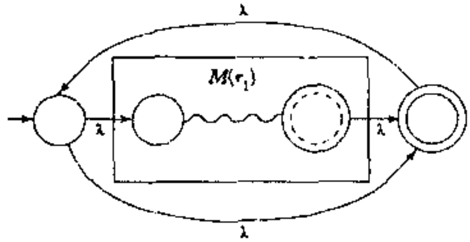


شکل ۳-۲ نمایش طرح کلی یک پذیرنده متناهی غیرقطعی که $L(r)$ را می پذیرد.



شکل ۳-۳ ماشین برای $L(r_1 + r_2)$.

شکل ۳-۴ ماشین برای $L(r_1 r_2)$.



شکل ۳-۵ ماشین برای $L(r_1^*)$.

از روی تفسیر گراف‌های شکل های ۳-۳ تا ۳-۵ باید روشن باشد که این روش ساخت، کار می‌کند. به بیان دقیق‌تر، ما می توانیم یک روش صوری برای ساخت حالات و انتقالات ماشین ترکیبی از روی حالات و انتقالات بخش‌ها ارائه دهیم، سپس بوسیله استقرا روی تعداد عملگرها ثابت می‌کنیم که روش ساخت منجر به ماشینی خواهد شد که زبان نشان داده شده بوسیله هر عبارت منظم خاصی را می‌پذیرد. ما در اینجا روی این مسئله تاکید نمی‌کنیم، ولی واضح است که نتایج همواره درست است.

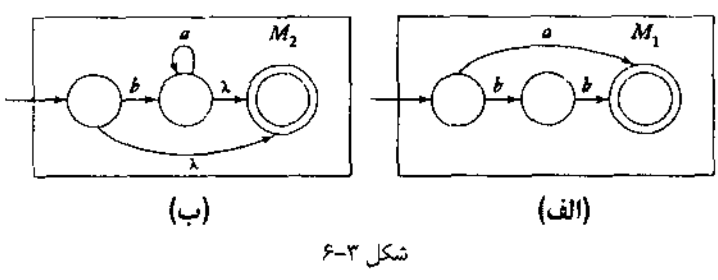
مثال ۳-۷: یک پذیرنده متناهی غیر قطعی بیابید که زبان $L(r)$ را بپذیرد، در جایی که

$$r = (a + bb)^*(ba^* + \lambda).$$

ماشین‌ها برای عبارات $(a + bb)$ و $(ba^* + \lambda)$ مستقیماً از اصول اولیه ساخته می‌شوند، که در شکل ۳-۶ نشان داده شده‌اند. با قراردادن اینها با استفاده از قضیه ۳-۱ در کنار یکدیگر، به راه حل نمایش داده شده در شکل ۳-۷ خواهیم رسید.

عبارات منظم برای زبان‌های منظم

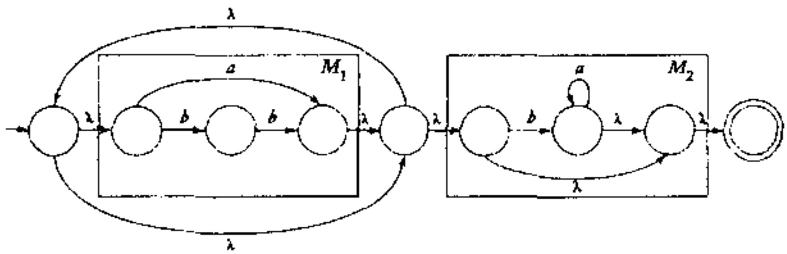
این حدس منطقی است که عکس قضیه ۳-۱ باید برقرار باشد، و برای هر زبان منظم باید یک عبارت منظم متناظر وجود داشته باشد. از آنجا که هر زبان منظم یک پذیرنده غیر قطعی مربوط به خود و از اینرو یک گراف انتقال دارد، همه آن کاری که لازم است انجام دهیم یافتن یک عبارت منظم با قابلیت تولید



(ب) (الف)

شکل ۳-۶

(الف) M_1 زبان $L(a + bb)$ را می‌پذیرد. (ب) M_2 زبان $L(ba^* + \lambda)$ را می‌پذیرد.



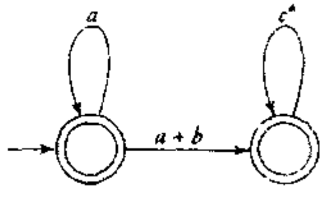
شکل ۳-۷ ماشینی که $L((a+bb)^*(ba^*+\lambda))$ را می‌پذیرد.

برچسب‌های همه راهها از q_0 به هر حالت نهایی است. این کار چندان دشوار به نظر نمی‌رسد ولی به دلیل وجود چرخه‌هایی که اغلب می‌توانند بطور دلخواه و با هر ترتیبی پیمایش شوند، پیچیده است. این امر موجب ایجاد مسئله کتاب داری می‌شود که باید با دقت با آن برخورد کرد. چندین راه برای انجام این کار وجود دارد، یکی از شهودی ترین روشها نیازمند بررسی جانبی در آن چیزی است که گراف‌های انتقال تعمیم یافته نامیده می‌شود. از آنجا که این ایده در این جا به طور محدودی به کار می‌رود و هیچ نقشی در بحث بعدی ما ندارد، ما به طور غیر رسمی به آن خواهیم پرداخت.

یک گراف انتقال تعمیم یافته، گراف انتقالی است که یال‌های آن با عبارات منظم برچسب گذاری شده‌اند، در غیر این صورت این گراف با گراف انتقال معمولی یکسان است. برچسب هر راه از حالت اولیه تا حالت نهایی عبارت است از اتصال چندین عبارت منظم، و در نتیجه خود نیز یک عبارت منظم است. رشته‌های نشان داده شده توسط این عبارات منظم، زیرمجموعه‌ای از زبانی است که توسط گراف انتقال تعمیم یافته پذیرفته شده باشد، و زبان کامل، اجتماع همه این زیر مجموعه‌های تولید شده است.

مثال ۳-۸: شکل ۳-۸ یک گراف انتقال تعمیم یافته را نشان می‌دهد. زبان پذیرفته شده توسط آن $L(a^* + a^*(a+b)c^*)$ است که از روی مشاهده گراف باید آشکار باشد. یال (q_0, q_0) با برچسب a چرخه‌ای است که می‌تواند هر تعداد از a هارا تولید کند، یعنی $L(a^*)$ را نشان می‌دهد. ما می‌توانیم این یال را توسط a^* برچسب گذاری نماییم بدون اینکه زبان پذیرفته شده توسط گراف تغییر یابد.

اگر برچسب یال‌های گراف هر پذیرنده متناهی غیر قطعی به طور مناسب تفسیر شوند آن را می‌توان به صورت یک گراف انتقال تعمیم یافته نشان داد. یک یال با برچسب یک نماد واحد a به عنوان یالی برچسب گذاری شده با عبارت a تفسیر می‌شود، در حالیکه یالی با بر چسب چندین نماد $a, b, \ldots$ به عنوان یالی برچسب گذاری شده با عبارت $a+b+\ldots$ تفسیر می‌شود. با این دید، نتیجه می‌گیریم که برای هر زبان منظم یک گراف انتقال تعمیم یافته وجود دارد که آن را می‌پذیرد. بالعکس، هر زبان پذیرفته شده توسط یک گراف انتقال تعمیم یافته، منظم است. از آنجا که برچسب هر راه در یک گراف انتقال تعمیم یافته یک عبارت منظم است، به نظر می‌رسد که این امر نتیجه فوری قضیه ۳-۱ می‌باشد. اگرچه



شکل۳ - ۸

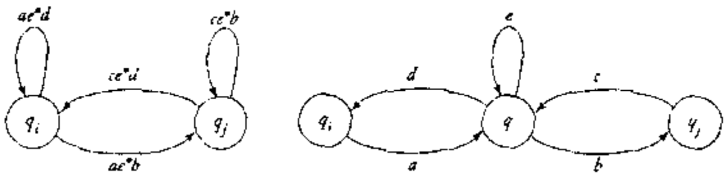
نکات ظریفی در این بحث وجود دارد. ما قصد پیگیری آنها را در اینجا نداریم، ولی خواننده را به تمرین ۱۶ بخش ۴-۳ برای جزییات ارجاع می‌دهیم.

معادل بودن در گراف‌های انتقال تعمیم یافته بر حسب زبان پذیرفته شده تعریف می‌شود. یک گراف انتقال تعمیم یافته با حالات $\{q, q_i, q_j, \ldots\}$ را در نظر بگیرید، که در آن q نه یک حالت نهایی و نه یک حالت اولیه است، و ما می‌خواهیم یک گراف انتقال تعمیم یافته معادل با یک حالت کمتر بوسیله حذف q تولید کنیم. ما در صورتی می‌توانیم این کار را انجام دهیم که زبان نشان داده شده توسط مجموعه‌ای از برچسب‌ها که با حرکت ما از q_0 به q_f به وجود می‌آیند را تغییر ندهیم. ساختاری که این کار را انجام می‌دهد درشکل ۳-۹ نشان داده شده است، که در آن حالت q حذف می‌شود و برچسب‌های یال $a, b, \ldots$ نشانگر عبارات کلی هستند. مورد نشان داده شده کلی ترین مورد است از این جهت که q یال‌هایی خروجی به هر سه رأس q_i, q_j, q دارد. در مواردی که یک یال در (a) حذف می‌شود ما یال معادل آنرا در (b) حذف می کنیم.

ساختار شکل ۳-۹ نشان می‌دهد که کدام یال‌ها باید معرفی شوند طوریکه زبان گراف انتقال تعمیم یافته هنگام حذف q و همه یال‌های ورودی و خروجی آن تغییر نکند. فرآیند کامل نیازمند آن است که این کار در مورد همه جفت‌های (q_i, q_j) در $Q - \{q\}$ قبل از حذف q انجام شود. اگرچه ما این امر را رسماً ثابت نمی‌کنیم، ولی می توان نشان داد که این ساختار به یک گراف انتقال تعمیم یافته منجر می‌شود. با قبول این امر، ما آماده هستیم تا نشان دهیم چگونه هر پذیرنده متناهی غیر قطعی می‌تواند به یک عبارت منظم مربوط باشد.

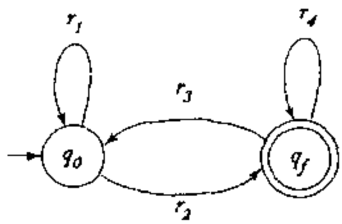
قضیه ۳-۲ : فرض کنید L یک زبان منظم باشد. آنگاه یک عبارت منظم r وجود دارد طوریکه $L = L(r)$ باشد.

اثبات : فرض کنید که M یک پذیرنده متناهی غیر قطعی باشد که L را می‌پذیرد. می توانیم بدون فقدان تعمیم فرض کنیم که M فقط یک حالت نهایی دارد و اینکه $q_0 \notin F$. ما از گراف M به عنوان یک گراف انتقال تعمیم یافته تفسیر می‌کنیم و ساختار بالا را روی آن اعمال می‌کنیم. برای حذف یک راس با برچسب q، از طرح کلی در شکل ۳-۹ برای همه زوج مرتب‌های (q_i, q_j) استفاده می‌کنیم. پس از آنکه همه یال‌های جدید اضافه شدند، q و همه یال‌های متلاقی با آن می‌توانند حذف شوند. ما این فرآیند را ادامه می‌دهیم، یک راس را پس از دیگری حذف می‌کنیم تا به وضعیتی برسیم که



(ب) (الف)

شکل ۳-۹



شکل ۳-۱۰

در شکل ۳-۱۰ نشان داده شده است. یک عبارت منظم که زبان پذیرفته شده توسط این گراف را نشان می‌دهد عبارت است از

$$r = r_1^* r_2 (r_4 + r_3 r_1^* r_2)^*. \qquad (۳-۱)$$

از آنجایی که همه توالی گراف‌های انتقال تعمیم یافته معادل با گراف اولیه هستند، ما می‌توانیم بوسیله استقرا بر روی تعداد حالات در گراف انتقال تعمیم یافته ثابت کنیم که عبارت منظم در (۳-۱) زبان L را نشان می‌دهد. ■

مثال ۳-۹: پذیرنده متناهی غیر قطعی شکل ۳-۱۱ (a) را در نظر بگیرید. گراف انتقال تعمیم یافته متناظر پس از حذف حالت q_1 در شکل ۳-۱۱ (b) نشان داده شده است. با شناسایی $r_1 = b + ab^*a, r_2 = ab^*b, r_3 = \varnothing, r_4 = a + b$ به عبارت منظم

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

برای ماشین اولیه می‌رسیم. روش ساخت مشروح در قضیه ۳-۲ خسته کننده بوده و به جواب‌های خیلی طولانی منجر می‌شود، ولی کاملا منظم بوده و همیشه کار می‌کند.

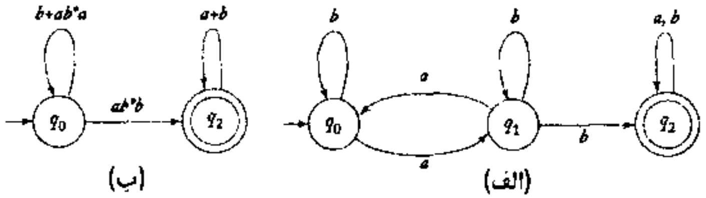
مثال ۳-۱۰: یک عبارت منظم برای زبان زیر بیابید.

$L = \{w \in \{a,b\}^*$: $n_a(w)$ زوج باشد و $n_b(w)$ فرد باشد$\}$

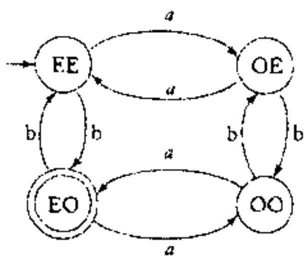
یک تلاش برای ساخت یک عبارت منظم مستقیماً از روی توصیف به همه انواع مشکلات منجر می‌شود. از طرف دیگر، یافتن یک پذیرنده متناهی غیر قطعی برای آن مادامیکه از برچسب گذاری کارای رئوس

www.prozhe.com

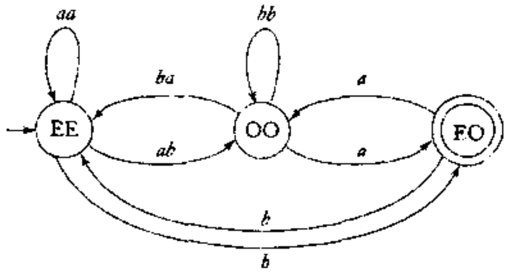
استفاده می‌کنیم، آسان است. ما رئوس را با EE برای نمایش تعداد زوج a ها و b ها، با OE برای تعداد فرد a ها و زوج b ها، با ها، و مانند آن برچسب گذاری می‌نماییم. بدین طریق ما به سادگی به راه حل شکل ۳-۱۲ می‌رسیم.



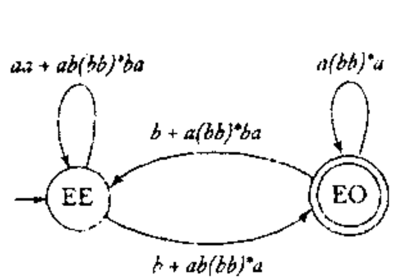
شکل ۳-۱۱



شکل ۳-۱۲



شکل ۳-۱۳



شکل ۳-۱۴

ما حالا می‌توانیم فرآیند تبدیل را روی یک عبارت منظم به روشی مکانیکی اعمال کنیم. ابتدا، حالتی با برچسب OE را حذف می‌کنیم، که گراف انتقال تعمیم یافته در شکل ۳-۱۳ را می‌دهد.
سپس، ما راسی با برچسب OO را حذف می‌کنیم. این کار شکل ۳-۱۴ را نتیجه می‌دهد.
سرانجام، رابطه (۳-۱) را با داشتن موارد زیر بکار می‌بریم.

$$r_1 = aa + ab(bb)^* ba,$$
$$r_2 = b + ab(bb)^* a,$$
$$r_3 = b + a(bb)^* ba,$$
$$r_4 = a(bb)^* a.$$

عبارت نهایی طولانی و پیچیده است، ولی راه بدست آوردن آن نسبتاً روشن و قابل فهم است.

عبارات منظم برای توصیف الگوهای ساده

در مثال ۱۵-۱ و تمرین ۱۵ در بخش ۲-۱، ما ارتباط بین پذیرنده‌های متناهی و برخی از اجزای ساده‌تر زبانهای برنامه سازی مانند شناسه‌ها، یا اعداد صحیح و حقیقی را کشف کردیم. ارتباط بین ماشین‌های متناهی و عبارات منظم بدین معنا است که می‌توانیم از عبارات منظم برای توصیف این ویژگی‌ها استفاده نماییم. دیدن این مورد، آسان است، برای مثال مجموعه کلیه اعداد صحیح قابل قبول در پاسکال را میتوان با عبارت منظم زیر تعریف کرد

$$sdd^*,$$

که در آن s نمایانگر علامت، با مقادیر ممکن $\{+,-,\lambda\}$، و d نمایانگر یک رقم از میان ارقام ۰ تا ۹ می‌باشد.

اعداد صحیح پاسکال یکی از موارد ساده‌ای است که گاهی اوقات یک "الگو" نامیده می‌شود، اصطلاحی که به معنای مجموعه‌ای از عناصر است که دارای صفات مشترک باشند. تطبیق الگو به معنای انتساب یک عنصر داده شده به یکی از چندین نوع است. اغلب، کلید تطبیق الگوی موفق، یافتن راهی موثر برای توصیف الگوهاست. این امر زمینه‌ای پیچیده و گسترده در علم کامپیوتر است که ما تنها می‌توانیم به طور مختصر بدان اشاره نماییم. گر چه مثال زیر یک مثال ساده شده است، ولی با وجود این آموزنده است، و چگونگی کاربرد ایده‌هایی که تا کنون در مورد آنها صحبت کرده‌ایم را در تطبیق الگوها تشریح می‌کند.

مثال۳-۱۱ : یکی از کاربردهای تطبیق الگوها ویرایش متن است. تمام ویرایشگرهای متن اجازه می‌دهند تا فایل‌ها برای رخداد یک رشته داده شده مورد پویش قرار گیرند. اغلب ویرایشگرها این مورد را به اجازه جستجوی الگوها توسعه می‌دهند. برای مثال، ویرایشگر ed در سیستم عامل یونیکس دستور زیر را تشخیص می‌دهد

$$/aba^*c/$$

و به عنوان دستور جستجوی فایل برای یافتن اولین رخداد از رشته ab که بدنبال آن به تعداد دلخواه a ها و بدنبال آن یک c بیاید تفسیر میکند. از این مثال در می‌یابیم که ویرایشگر یونیکس عبارات منظم را تشخیص میدهد (اگر چه از طریق دیگری به جز آنچه که در اینجا استفاده شد برای مشخص کردن عبارات منظم استفاده میکند).

یک وظیفه همراه با چالش در چنین کاربردی، نوشتن یک برنامه کارا برای تشخیص الگوهای رشته است. جستجو در یک فایل برای یافتن رخداد یک رشته داده شده یک تمرین برنامه سازی بسیار ساده است، ولی در اینجا وضعیت پیچیده‌تر است. ما با رشته‌های سروکار داریم که تعداد نامحدودی از الگوهای پیچیده را به دلخواه دارند، به علاوه این الگوها از قبل ثابت نیستند و در زمان اجرا ایجاد می‌شوند. توصیف الگوها بخشی از ورودی است، بنابراین فرآیند تشخیص باید منعطف باشد. برای حل این مشکلات، اغلب از ایده‌هایی در نظریه ماشین‌ها استفاده می‌شود.

اگر الگو توسط یک عبارت منظم مشخص شود، آنگاه برنامه تطبیق الگو می‌تواند این توصیف را دریافت نموده و آن را به یک پذیرنده متناهی غیر قطعی معادل با استفاده از روش ساخت در قضیه ۳-۱ تبدیل کند. سپس قضیه ۲-۲ می‌تواند جهت کاهش این پذیرنده متناهی غیر قطعی به پذیرنده متناهی قطعی استفاده شود. این پذیرنده متناهی قطعی، در قالب یک جدول انتقال یک الگوریتم تطبیق الگو کارا می‌باشد. همه کاری که برنامه نویس باید انجام دهد مهیا کردن راه‌اندازی است که چهار چوب عمومی برای استفاده از جدول را ارائه دهد. بدین طریق ما بطور خودکار می‌توانیم تعداد زیادی از الگوها را که در زمان اجرا تعریف می‌شوند مورد بررسی قرار دهیم.

همچنین کارآیی برنامه باید در نظر گرفته شود. ساخت ماشین‌های متناهی از روی عبارات منظم با استفاده از قضایای ۲-۱ و ۳-۱ به ماشین‌هایی با تعداد حالات زیاد منجر می‌شود. اگر فضای حافظه یک مسئله باشد، روش کاهش حالت توصیف شده در بخش ۲-۴ مفید خواهد بود.

تمرین‌ها

۱- با استفاده از ساختار قضیه ۳-۱ یک پذیرنده متناهی غیرقطعی بیابید که زبان $L(ab^*aa + bba^*ab)$ را بپذیرد.

۲- یک پذیرنده متناهی غیرقطعی بیابید که مکمل زبان تمرین ۱ را بپذیرد.

۳- یک پذیرنده متناهی غیرقطعی ارائه دهید که زبان $L((a+b)^*b(a+bb)^*)$ را بپذیرد.

۴- پذیرنده های متناهی غیر قطعی بیابید که زبان‌های زیر را بپذیرند.

الف) $L(aa^* + aba^*b^*)$

ب) $L(ab(a+ab)^*(a+aa))$

ج) $L((abab)^* + (aaa^* + b)^*)$

د) $L(((aa^*)^*b)^*)$

۵- پذیرنده‌های متناهی غیر قطعی بیابید که زبان‌های زیر را بپذیرند.

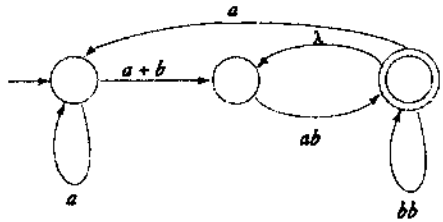
الف) $L = L(ab^*a^*) \bigcup L((ab)^*ba)$,

ب) $L = L(ab^*a^*) \cap L((ab)^*ba)$.

۶- یک پذیرنده متناهی غیرقطعی برای تمرین ۱۵(و) ، بخش ۳-۱ بیابید. با استفاده از آن یک عبارت منظم برای زبان ارائه دهید.

۷- قوانین صریحی برای ساختاری پیشنهاد شده در شکل ۳-۹ ، در هنگامی که یال‌های مختلف در ۳-۹ (الف) حذف شوند، ارائه دهید.

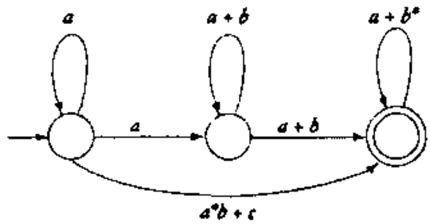
۸- گراف انتقال تعمیم یافته زیر را در نظر بگیرید.



الف) یک گراف انتقال تعمیم یافته معادل بیابید که فقط دارای دو حالت باشد.

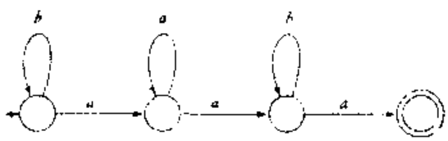
ب) زبان پذیرفته شده بوسیله این گراف چیست؟

۹- چه زبانی توسط گراف انتقال تعمیم یافته زیر پذیرفته می‌شود؟

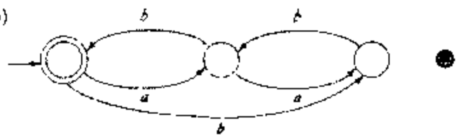


۱۰- عبارات منظمی برای زبان‌های پذیرفته شده بوسیله ماشین‌های زیر بیابید.

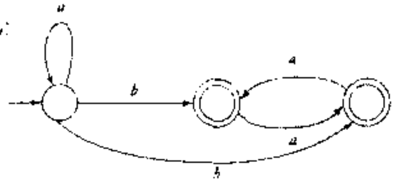
(الف)



(ب)



(پ)



۱۱- مثال ۳-۱۰ را دوباره انجام دهید، که در آن ابتدا حالت OO حذف شود.

۱۲- یک عبارت منظم برای زبان‌های زیر روی $\{a,b\}$ بیابید.

الف) $L = \{w :$ هر دو زوج باشند $n_a(w)$ و $n_b(w)\}$

ب) $L = \{w : (n_a(w) - n_b(w)) \bmod 3 = 1\}$

ج) $L = \{w : (n_a(w) - n_b(w)) \bmod 3 \neq 0\}$

د) $L = \{w :$ زوج باشد $2n_a(w) + 3n_b(w)\}$

۱۳- یک عبارت منظم بیابید که مجموعه همه رشته‌های سه تایی را تولید کند که جمع اعداد دودویی را بدرستی مانند تمرین ۲۳ بخش ۲-۱ تعریف می‌کند.

۱۴- ثابت کنید که ساختار پیشنهادی توسط شکل ۳-۹ گراف‌های انتقال تعمیم یافته را تولید می‌کند.

۱۵- یک عبارت منظم برای مجموعه همه اعداد حقیقی در پاسکال بنویسید.

۱۶- یک عبارت منظم برای مجموعه‌های پاسکال که اعضای آن اعداد صحیح هستند، بیابید.

۱۷- در برخی کاربردها، مانند برنامه نویسی که املا را بررسی می‌نمایند، ما ممکن است به تطبیق دقیق الگو نیاز نداشته باشیم، و فقط یک تقریب کافی است. هنگامی که نماد یک تطابق تقریبی مشخص شد، نظریه ماشین‌ها می‌تواند برای ساخت تطبیق دهنده‌های تقریبی الگو به کار رود. برای تشریح این مورد، الگوهایی را در نظر بگیرید که از عبارت اصلی بوسیله درج یک نماد مشتق شده‌اند. بفرض L یک زبان منظم روی Σ باشد، تعریف می‌کنیم

$$insert(L) = \{uav : a \in \Sigma, uv \in L\}.$$

در حقیقت، $insert(L)$ شامل همه رشته‌های ایجاد شده از L است که بوسیله درج یک نماد ساختگی در هر جای کلمه ایجاد می‌شود.

الف) اگر یک پذیرنده متناهی غیر قطعی L داشته باشیم، نشان دهید که چگونه می توان یک پذیرنده متناهی غیر قطعی برای $insert(L)$ ساخت؟ *

ب) بحث کنید چگونه می‌توان از این پذیرنده، برای نوشتن یک برنامه شناسایی الگو برای $insert(L)$ استفاده کرد، که درآن ورودی یک عبارت منظم برای L است. **

۱۸- مشابه تمرین قبل، همه کلماتی را در نظر بگیرید که می‌توانند از روی L بوسیله حذف یک نماد تنها از رشته، شکل گیرند. بطور صوری این عملیات را *رهاسازی* زبان‌ها تعریف می‌کنیم. با فرض داشتن یک پذیرنده متناهی غیرقطعی، یک پذیرنده متناهی غیرقطعی برای $drop(L)$ بسازید. *

۱۹- با استفاده از ساختار قضیه ۳-۱، پذیرنده‌های متناهی غیر قطعی برای $L(a\varnothing)$ و $L(\varnothing^*)$ بیابید. آیا نتیجه با تعریف این زبان‌ها سازگار است؟

۳-۳ گرامرهای منظم

روش سوم جهت توصیف زبان‌های منظم استفاده از گرامرهای ساده خاص است. گرامرها اغلب یک روش دیگر از مشخص کردن زبان‌ها است. هر گاه یک خانواده زبان را بوسیله ماشین یا روشی دیگر تعریف می‌کنیم، به دانستن اینکه چه نوع گرامری با آن خانواده ارتباط دارد علاقه‌مندیم. ابتدا، به گرامرهایی که زبان‌های منظم را تولید می‌کنند، نگاه می‌کنیم.

گرامرهای خطی راست و خطی چپ

تعریف ۳-۳ : گرامر $G = (V, T, S, P)$ را **خطی راست** گویند اگر همه قواعد به شکل

$$A \to xB,$$
$$A \to x,$$

که $A, B \in V$ و $x \in T^*$. یک گرامر را **خطی چپ** گویند اگر همه قواعد آن به شکل

$$A \to Bx,$$

یا

$$A \to x$$

باشند.

یک **گرامر منظم** گرامری است که یا خطی راست و یا خطی چپ باشد.

توجه کنید که در گرامر منظم، حداکثر یک متغیر در سمت راست هر قانون ظاهر می‌شود. بعلاوه، متغیر باید همواره سمت راست ترین یا سمت چپ ترین نماد در سمت راست هر قانون باشد.

مثال ۳-۱۲ : گرامر $G_1 = (\{S\}, \{a, b\}, S, P_1)$، که P_1 عبارت است از

$$S \to abS \mid a$$

خطی راست است. گرامر $G_2 = (\{S, S_1, S_2\}, \{a, b\}, S, P_2)$ با قوانین

$$S \to S_1ab,$$
$$S_1 \to S_1ab \mid S_2,$$
$$S_2 \to a,$$

خطی چپ است. هم G_1 و هم G_2، گرامرهای منظم هستند.

دنباله

$$S \Rightarrow abS \Rightarrow ababS \Rightarrow ababa$$

www.prozhe.com

اشتقاقی در G_1 است. از روی تنها این نمونه حدس این مطلب آسان است که $L(G_1)$ زبان نشان داده شده بوسیله عبارت منظم $r=(ab)^*a$ می‌باشد. بطریق مشابه می‌توانیم ببینیم که $L(G_2)$ زبان منظم $L(aab(ab)^*)$ می‌باشد.

مثال ۳-۱۳ : گرامر $G=(\{S,A,B\},\{a,b\},S,P)$ با قوانین

$$S \to A,$$
$$A \to aB \mid \lambda,$$
$$B \to Ab,$$

منظم نیست. اگر چه هر قانونی یا به شکل خطی راست و یا خطی چپ است، ولی این گرامر نه خطی راست و نه خطی چپ است، و بنابراین منظم نیست. این گرامر مثالی از یک **گرامر خطی** است. یک گرامر خطی گرامری است که حداکثر یک متغیر می‌تواند در سمت راست هر قانون آن ظاهر شود، بدون اینکه محدودیتی در محل قرار گرفتن این متغیر وجود داشته باشد. واضح است که یک گرامر منظم، همیشه خطی است، ولی همه گرامرهای خطی، منظم نیستند.

هدف بعدی ما این است که نشان دهیم گرامرهای منظم با زبان‌های منظم مرتبط هستند، و برای هر زبان منظم یک گرامر منظم وجود دارد. بنابراین، گرامرهای منظم روشی دیگر از بحث در مورد زبان‌های منظم هستند.

گرامرهای خطی راست زبان‌های منظم را تولید می‌کنند

ابتدا، نشان می‌دهیم که زبان تولید شده بوسیله یک گرامر خطی راست، همواره منظم است. برای انجام این کار، یک پذیرنده متناهی غیر قطعی می‌سازیم که اشتقاق‌های یک گرامر خطی راست را تقلید می‌نماید. توجه کنید که شکل‌های جمله‌ای از یک گرامر خطی راست دارای شکل خاصی هستند که در آنها دقیقاً یک متغیر وجود دارد که به عنوان سمت راست ترین نماد ظاهر می‌شود. اکنون فرض کنید که یک مرحله در اشتقاقی را داریم

$$ab\ldots cD \Rightarrow ab\ldots cdE,$$

که با استفاده از قانون $D \to dE$ تولید شده است. پذیرنده متناهی قطعی متناظر می‌تواند این مرحله را بوسیله رفتن از حالت D به حالت E هنگامی که با نماد ورودی d مواجه می‌شود، تقلید نماید. در این طرح کلی، حالت ماشین متناظر با متغیری در شکل جمله‌ای می‌باشد، در حالیکه قسمتی از رشته که تا کنون پردازش شده همان پیشوند پایانه‌ای از شکل جمله‌ای است. این ایده ساده پایه‌ای برای قضیه بعدی است.

قضیه ۳-۳ : فرض کنید $G=(V,T,S,P)$ یک گرامر خطی راست باشد. در اینصورت $L(G)$ یک زبان منظم است.

اثبات: فرض می‌کنیم $V=\{V_0, V_1, \ldots\}$، که $S=V_0$، و قوانینی به شکل $V_0 \rightarrow v_1 V_i, V_i \rightarrow v_2 V_j, \ldots$ یا $V_n \rightarrow v_l, \ldots$ داریم. اگر w رشته‌ای در $L(G)$ باشد، آنگاه بدلیل شکل قوانین در G، اشتقاق باید به شکل زیر باشد

$$\begin{aligned} V_0 &\Rightarrow v_1 V_i \\ &\Rightarrow v_1 v_2 V_j \\ &\overset{*}{\Rightarrow} v_1 v_2 \ldots v_k V_n \\ &\Rightarrow v_1 v_2 \ldots v_k v_l = w. \end{aligned} \tag{۲-۳}$$

ماشینی که باید ساخته شود، اشتقاق را با مصرف به نوبت هر یک از این v ها مجدداً تولید می‌کند. حالت اولیه ماشین دارای برچسب V_0 می‌باشد، و برای هر متغیر V_i یک حالت غیر نهایی با برچسب V_i وجود خواهد داشت. برای هر قانون

$$V_i \rightarrow a_1 a_2 \ldots a_m V_j,$$

ماشین انتقال‌هایی برای اتصال V_i و V_j خواهد داشت، یعنی δ بصورت زیر تعریف می‌شود

$$\delta^*(V_i, a_1 a_2 \ldots a_m) = V_j.$$

برای هر قانون

$$V_i \rightarrow a_1 a_2 \ldots a_m,$$

انتقال متناظر از ماشین بصورت زیر خواهد بود

$$\delta^*(V_i, a_1 a_2 \ldots a_m) = V_f,$$

که V_f یک حالت نهایی است. حالات میانی که برای انجام این کار مورد نیاز می‌باشند، اهمیتی ندارند و می‌توانند دارای برچسب‌های دلخواه شوند. طرح کلی در شکل ۳-۱۵ نشان داده شده است. ماشین کامل از ردیف کردن چنین قسمت‌های مجزایی بدست می‌آید.

اکنون فرض کنید که $w \in L(G)$ بطوریکه رابطه (۳-۲) ارضا شود. در پذیرنده متناهی غیر قطعی ساخته شده، مسیری از V_0 به V_i با برچسب v_1، مسیری از V_i به V_j با برچسب v_2 و مانند آن وجود خواهد داشت بطوریکه

$$V_f \in \delta^*(V_0, w),$$

و w توسط M پذیرفته می‌شود.

بطور معکوس، فرض کنید که w توسط M پذیرفته می‌شود. بدلیل روشی که M ساخته می‌شود، ماشین برای پذیرش w، باید گذری در دنباله حالات $V_0, V_i, \ldots$ تا V_f با استفاده از مسیرهایی با برچسب $v_1, v_2, \ldots$ داشته باشد. بنابراین w باید به شکل زیر باشد

$$w = v_1 v_2 \ldots v_k v_l$$

و اشتقاق

$$V_0 \Rightarrow v_1 V_i \Rightarrow v_1 v_2 V_j \overset{*}{\Rightarrow} v_1 v_2 \ldots v_k V_k \Rightarrow v_1 v_2 \ldots v_k v_l$$

ممکن است. بنابراین w در $L(G)$ هست، و قضیه اثبات می‌شود. ■

مثال۳-۱۴: یک ماشین متناهی بسازید که زبان تولید شده بوسیله گرامر زیر را بپذیرد.

$$V_0 \to aV_1,$$
$$V_1 \to abV_0 \mid b.$$

گراف انتقال را با رئوس V_0, V_1 و V_f شروع می‌کنیم. اولین قانون تولید یک یال با برچسب a بین V_0 و V_1 ایجاد می‌کند. برای قانون دوم، ما نیاز به معرفی یک راس اضافی داریم بطوریکه بتوانیم مسیری با برچسب ab بین V_1 و V_0 داشته باشیم. سرانجام، ما نیاز به اضافه کردن یالی با برچسب b بین V_1 و V_f داریم، که به ماشین نشان داده شده در شکل ۳-۱۶ منجر می‌شود. زبان تولید شده بوسیله گرامر و پذیرفته شده بوسیله ماشین، زبان منظم $L((aab)^*ab)$ می‌باشد.

▲

گرامرهای خطی راست برای زبان‌های منظم

برای نشان دادن اینکه زبان منظم می‌تواند بوسیله یک گرامر خطی راست تولید شود، از پذیرنده متناهی قطعی برای زبان شروع می‌کنیم و ساختار نشان داده شده در قضیه ۳-۳ را معکوس می‌کنیم. اکنون حالات پذیرنده متناهی قطعی، متغیرهای گرامر، و نمادهایی که باعث انتقال می‌شوند پایانه های قوانین خواهند بود.

قضیه ۳-٤ : اگر L یک زبان منظم روی الفبای Σ باشد، آنگاه یک گرامر خطی راست $G = (V, \Sigma, S, P)$ وجود دارد بطوریکه $L = L(G)$.

اثبات : فرض کنید $M = (Q, \Sigma, \delta, q_0, F)$ یک پذیرنده متناهی قطعی باشد که L را می‌پذیرد. فرض می‌کنیم $Q = \{q_0, q_1, \ldots, q_n\}$ و $\Sigma = \{a_1, a_2, \ldots, a_m\}$ باشد. گرامر خطی راست $G = (V, \Sigma, S, P)$ را با

$$V = \{q_0, q_1, \ldots, q_n\}$$

و $S = q_0$ بسازید. برای هر انتقال

$$\delta(q_i, a_j) = q_k$$

از M، قانون زیر را در P قرار می‌دهیم

$$q_i \to a_j q_k. \qquad (۳-۳)$$

بعلاوه، اگر q_k در F باشد، به P قانون زیر را اضافه می‌کنیم

$$q_k \to \lambda \qquad (۳-۴)$$

ما ابتدا نشان می‌دهیم که G تعریف شده بدین طریق می‌تواند هر رشته‌ای در L را تولید کند. $w \in L$ را به شکل $w = a_i a_j \ldots a_k a_l$.

در نظر بگیرید. برای اینکه M این رشته را بپذیرد باید حرکات زیر را انجام دهد

$$\delta(q_0, a_i) = q_p,$$
$$\delta(q_p, a_j) = q_r,$$
$$\vdots$$
$$\delta(q_s, a_k) = q_t,$$
$$\delta(q_t, a_l) = q_f \in F.$$

با توجه به ساختار، گرامر برای هر یک از این δ ها، یک قانون خواهد داشت. بنابراین می توانیم اشتقاق زیر را داشته باشیم

$$q_0 \Rightarrow a_i q_p \Rightarrow a_i a_j q_r \overset{*}{\Rightarrow} a_i a_j \ldots a_k q_t \quad (۵-۳)$$
$$\Rightarrow a_i a_j \ldots a_k a_l q_f \Rightarrow a_i a_j \ldots a_k a_l,$$

با داشتن گرامر G و $w \in L(G)$.

بر عکس، اگر $w \in L(G)$ باشد، آنگاه اشتقاق آن باید به شکل (۳-۵) باشد، ولی این مطلب ایجاب می کند که

$$\delta^*(q_0, a_i a_j \ldots a_k a_l) = q_f,$$

و اثبات تکمیل می‌شود. ■

به منظور ساخت یک گرامر، مفید است توجه کنید که محدودیت اینکه M پذیرنده متناهی قطعی باشد برای اثبات قضیه ۳-۴ ضروری نیست. اگر M یک پذیرنده متناهی غیر قطعی باشد، همان ساختار با اندکی تغییر می‌تواند استفاده شود.

مثال ۳-۱۵ : یک گرامر خطی راست برای $L(aab^*a)$ بسازید. تابع انتقال برای یک پذیرنده متناهی غیر قطعی، به همراه قوانین تولید گرامر متناظر در شکل ۳-۱۷ نشان داده شده است. این نتیجه بسادگی بوسیله دنبال کردن ساختار قضیه ۳-۴ بدست آمده است. رشته $aaba$ می تواند از روی گرامر ساخته شده بصورت زیر مشتق شود

$$q_0 \Rightarrow aq_1 \Rightarrow aaq_2 \Rightarrow aabq_2 \Rightarrow aabaq_f \Rightarrow aaba.$$

$\delta(q_0, a) = \{q_1\}$	$q_0 \to aq_1$
$\delta(q_1, a) = \{q_2\}$	$q_1 \to aq_2$
$\delta(q_2, b) = \{q_2\}$	$q_2 \to bq_2$
$\delta(q_2, a) = \{q_f\}$	$q_2 \to aq_f$
$q_f \in F$	$q_f \to \lambda$

شکل ۳-۱۷

هم ارزی بین زبان‌های منظم و گرامرهای منظم

دو قضیه قبلی ارتباط بین زبان‌های منظم و گرامرهای خطی راست را برقرار می‌سازد. می‌توان ارتباط مشابهی بین زبان‌های منظم و گرامرهای خطی چپ برقرار کرد، و بدین وسیله هم ارزی کامل بین گرامرهای منظم و زبان‌های منظم را نشان داد.

قضیه ۳-۵ : زبان L منظم است اگر و فقط اگر یک گرامر خطی چپ G وجود داشته باشد بطوریکه $L = L(G)$ باشد.

اثبات : ما فقط ایده اساسی را مطرح می‌کنیم. گرامر خطی چپ با قوانین تولید به شکل

$$A \rightarrow Bv,$$

یا

$$A \rightarrow v,$$

مفروض است. از روی آن یک گرامر خطی راست $\hat{G}$ می‌سازیم که در آن هر قانون از G با

$$A \rightarrow v^R B,$$

یا

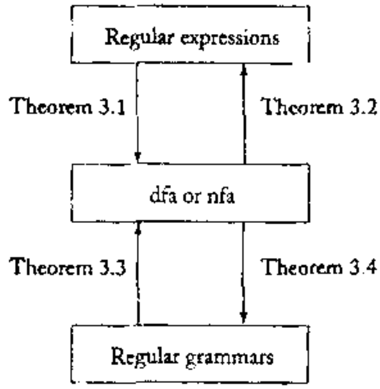
$$A \rightarrow v^R,$$

بترتیب جایگزین شده است. با چند مثال سریعاً روشن خواهد شد که $L(G) = (L(\hat{G}))^R$. سپس از تمرین ۱۲، بخش ۳-۲ استفاده می‌کنیم که می‌گوید معکوس هر زبان منظم نیز منظم است. از آنجایی که $\hat{G}$ خطی راست است، $L(\hat{G})$ منظم است. ولی در اینصورت $L((\hat{G}))^R$ و $L(G)$ نیز منظم هستند. ■

با قرار دادن قضایای ۳-۴ و ۳-۵ در کنار یکدیگر، به هم ارزی زبان های منظم و گرامرهای منظم می‌رسیم.

قضیه ۳-۶ : یک زبان L منظم است اگر و فقط اگر یک گرامر منظم G وجود داشته باشد بطوریکه $L = L(G)$ باشد.

اکنون چندین روش برای توصیف زبان‌های منظم داریم: پذیرنده‌های متناهی قطعی، پذیرنده های متناهی غیر قطعی، عبارات منظم، و گرامرهای منظم. اگر چه در هر مورد ممکن است یکی از این روش‌ها مناسب تر باشد، همه آنها دارای قدرت یکسانی هستند. همه آنها تعریف کامل و غیر مبهمی از یک زبان منظم ارائه می‌دهند. ارتباط بین همه این مفاهیم همچنانکه در شکل ۳-۱۸ نشان داده شده است، بوسیله چهار قضیه در این بخش برقرار می‌شود.



شکل ۳-۱۸

تمرین‌ها

۱- یک پذیرنده متناهی قطعی بسازید که زبان تولید شده بوسیله گرامر زیر را بپذیرد.

$$S \rightarrow abA,$$
$$A \rightarrow baB,$$
$$B \rightarrow aA \mid bb.$$

۲- یک گرامر منظم بیابید که زبان $L(aa^*(ab+a)^*)$ را تولید کند.

۳- یک گرامر خطی چپ برای زبان تمرین ۱ بسازید.

۴- گرامرهای خطی راست و خطی چپ را برای زبان زیر بسازید.

$$L = \{a^n b^m : n \geq 2, m \geq 3\}.$$

۵- یک گرامر خطی راست برای زبان $L((aab^*ab)^*)$ بسازید.

۶- یک گرامر منظم بیابید که زبان روی $\Sigma = \{a,b\}$ را تولید کند بطوریکه شامل همه رشته‌هایی باشد که بیشتر از سه a نداشته باشند.

۷- در قضیه ۳-۵، ثابت کنید که $L(\hat{G}) = (L(G))^R$.

۸- ساختاری پیشنهاد کنید که بوسیله آن یک گرامر خطی چپ بتواند مستقیماً از روی یک پذیرنده متناهی غیر قطعی بدست آید.

۹- یک گرامر خطی چپ برای زبان تمرین ۵ بیابید.

۱۰- یک گرامر منظم برای زبان $L = \{a^n b^m :$ زوج است $n+m\}$ بیابید.

۱۱- یک گرامر منظم بیابید که زبان زیر را تولید کند.

$$L = \{w \in \{a,b\}^* : \text{زوج است } n_a(w) + 3n_b(w)\}.$$

۱۲- گرامرهای منظمی برای زبان‌های زیر روی $\{a,b\}$ بیابید.

الف) $L = \{w:$ هر دو زوج هستند $n_b(w)$ و $n_a(w)\}$

ب) $L = \{w : (n_a(w) - n_b(w)) \bmod 3 = 1\}$

ج) $L = \{w : (n_a(w) - n_b(w)) \bmod 3 \neq 0\}$

د) $L = \{w :$ فرد است $|n_a(w) - n_b(w)|\}$

۱۳- نشان دهید که برای هر زبان منظم که شامل λ نباشد، یک گرامر خطی راست وجود دارد که قوانین تولید آن به اشکال

$$A \to aB$$

یا

$$A \to a,$$

محدود باشد که $A, B \in V$ و $a \in T$.

۱۴- نشان دهید که هر گرامر منظم G که برای آن $L(G) \neq \varnothing$ باشد، باید حداقل دارای یک قانون تولید به شکل

$$A \to x,$$

باشد که $A \in V$ و $x \in T^*$.

۱۵- یک گرامر منظم بیابید که مجموعه همه اعداد حقیقی پاسکال را تولید کند.

۱۶- فرض کنید $G_1 = (V_1, \Sigma, S_1, P_1)$ یک گرامر خطی راست و $G_2 = (V_2, \Sigma, S_2, P_2)$ یک گرامر خطی چپ باشد، و همچنین V_1 و V_2 مجزا باشند. گرامر خطی $G = (\{S\} \cup V_1 \cup V_2, \Sigma, S, P)$ را در نظر بگیرید که S در $V_1 \cup V_2$ نبوده و $P = \{S \to S_1 \mid S_2\} \cup P_1 \cup P_2$ باشد. نشان دهید که $L(G)$ منظم است.

۴ خواص زبان‌های منظم

ما زبان‌های منظم را تعریف کردیم، راه‌های ارائه آنها را مطالعه نمودیم، و چند مثال از کاربرد آنها دیدیم. اکنون این سوال را مطرح می‌کنیم که زبان‌های منظم تا چه حدی عمومیت دارند. آیا همه زبان های صوری، منظم هستند؟ شاید هر مجموعه‌ای بتواند بوسیله یک ماشین متناهی ولو پیچیده پذیرفته شود. همانگونه که به زودی خواهیم دید، پاسخ به این فرضیات قطعاً خیر هست. ولی برای درک اینکه چرا اینگونه است، باید بیشتر در مورد طبیعت زبان‌های منظم تحقیق کنیم و ببینیم که این خانواده چه خواصی دارد؟

اولین سوالی که مطرح می‌کنیم این است که وقتی عملیاتی روی زبان‌های منظم انجام می‌دهیم چه اتفاقی می‌افتد. عملیاتی که ما در نظر می‌گیریم عملیات ساده بر روی مجموعه‌ها است مانند اتصال، و همه عملیاتی است که هر رشته از زبان را تغییر می‌دهند مانند نمونه‌ای که در تمرین ۲۲، بخش ۲-۱ آورده شد. آیا زبان نتیجه شده هنوز منظم است؟ به این سوال به عنوان یک سوال **بستاری** مراجعه می‌کنیم. اگر چه خواص بستاری اغلب جنبه نظری دارد، به ما کمک می‌کند تا بین خانواده‌های زبان‌های مختلف که با آنها مواجه خواهیم شد، تمایز قائل شویم.

مجموعه دوم از سوالات مربوط به خانواده‌های زبان‌ها درباره توانایی تصمیم گیری ما در مورد خواص خاصی بحث می‌کند. برای مثال، آیا می‌توانیم بگوییم یک زبان متناهی است یا خیر؟ همچنانچه خواهیم دید، چنین سوالاتی در مورد زبان‌های منظم بسادگی پاسخ داده می‌شوند، ولی پاسخ‌ها برای خانواده‌های دیگر زبان‌ها آسان نیست.

سرانجام، سوال مهمی را در نظر می‌گیریم: چگونه می‌توانیم بگوییم که آیا یک زبان داده شده منظم هست یا خیر؟ اگر زبانی در واقع منظم باشد، می‌توانیم همواره این مطلب را بوسیله ارائه یک پذیرنده متناهی قطعی، عبارت منظم، یا گرامر منظم برای آن نشان دهیم. ولی اگر چنین نباشد، ما نیاز داریم به گونه دیگری عمل کنیم. یک روش برای نشان دادن اینکه یک زبان منظم نیست، مطالعه صفات عمومی زبان‌های منظم یعنی خصوصیاتی که در بین همه زبان‌های منظم مشترک هستند، می‌باشد. اگر برخی از چنین خواصی را بدانیم، و اگر بتوانیم نشان دهیم که زبان مورد بحث آن خاصیت را ندارد، آنگاه

می‌توانیم بگوییم که زبان منظم نیست.

در این فصل، به انواع خواص زبان‌های منظم می‌نگریم. این خواص به مقدار زیادی به ما می گویند که چه کارهایی را زبان‌های منظم می‌توانند و چه کارهایی را نمی‌توانند انجام دهند. بعداً، هنگامی که به همان سوالات در مورد خانواده‌های دیگر زبان‌ها بنگریم، شباهت‌ها و تفاوت‌هایی در این خواص به ما امکان می‌دهند تا بین خانواده‌های مختلف زبان‌ها تمایز قائل شویم.

۴-۱ خواص بستاری زبان‌های منظم

این سوال را در نظر بگیرید: دو زبان منظم L_1 و L_2 مفروضند. آیا اجتماع آنها نیز منظم است؟ جواب در موارد خاصی ممکن است واضح باشد، ولی در اینجا می‌خواهیم مسئله را به طور کلی بررسی نماییم. آیا این مطلب برای همه زبان‌های منظم L_1 و L_2 درست است؟ پاسخ این سوال بلی هست، حقیقتی که بدین گونه بیان می‌کنیم که خانواده زبان‌های منظم تحت اجتماع **بسته** است. ما می‌توانیم سوالات مشابه را درباره انواع دیگر عملیات روی زبان‌ها بپرسیم. این سوالات ما را به مطالعه عمومی خواص بستاری زبان‌ها رهنمون می‌سازد.

خواص بستاری خانواده‌های مختلف زبان‌ها تحت عملیات مختلف از لحاظ نظری قابل توجه است. در نگاه اول، ممکن است اهمیت عملی که این خواص دارا هستند، واضح نباشد. در حقیقت، برخی از این خواص اهمیت عملی بسیار کمی دارند، ولی نتایج کاربردی بسیاری دارند. بوسیله نگاه به طبیعت عمومی خانواده‌های زبان‌ها، خواص بستاری به ما در پاسخ به سوالات دیگر و عملی‌تر کمک می‌کنند. ما بعداً نمونه‌هایی از این را (قضیه ۴-۷ و مثال ۴-۱۳) در این فصل خواهیم دید.

بستار تحت عملیات ساده روی مجموعه

با نگاه به بسته بودن زبان‌های منظم تحت عملیات معمول مجموعه، مانند اجتماع و اشتراک شروع می‌کنیم.

قضیه ۴-۱: اگر L_1 و L_2 زبان‌های منظم باشند، آنگاه $\overline{L_1}, L_1L_2, L_1 \cap L_2, L_1 \cup L_2$ و L_1^* نیز منظم هستند. گوییم که خانواده زبان‌های منظم تحت اجتماع، اشتراک، اتصال، متمم گیری، و بستار ستاره‌ای بسته است.

اثبات : اگر L_1 و L_2 منظم باشند، آنگاه عبارات منظم r_1 و r_2 وجود دارند بطوریکه $L_1 = L(r_1)$ و $L_2 = L(r_2)$ باشد. طبق تعریف، $r_1 + r_2, r_1r_2$ و r_1^* عبارات منظمی هستند که بترتیب زبان‌های $L_1 \cup L_2, L_1L_2$ و L_1^* را نشان می‌دهند. بنابراین، بسته بودن تحت اجتماع، اتصال، و بستار ستاره‌ای فوراً حاصل می‌شود.

برای نشان دادن بسته بودن تحت متمم گیری، فرض کنید $M = (Q, \Sigma, \delta, q_0, F)$ یک پذیرنده متناهی قطعی باشد که L_1 را می‌پذیرد. در اینصورت پذیرنده متناهی قطعی

$$\bar{M} = (Q, \Sigma, \delta, q_0, Q - F)$$

$\overline{L_1}$ را می‌پذیرد. این مطلب نسبتاً قابل فهم است. ما قبلاً این نتیجه را در تمرین ۴ در بخش ۲-۱ پیشنهاد کرده بودیم. توجه کنید که در تعریف یک پذیرنده متناهی قطعی، فرض می‌کنیم δ^* یک تابع کلی باشد، بطوریکه $\delta^*(q_0, w)$ برای همه $w \in \Sigma^*$ تعریف شده است. در نتیجه یا $\delta^*(q_0, w)$ یک حالت نهایی است که در این مورد $w \in L$ هست، یا $\delta^*(q_0, w) \in Q - F$ و $w \in \overline{L}$ می‌باشد.

نمایش بسته بودن تحت اشتراک، مقداری کار بیشتری می‌طلبد. فرض کنید $L_1 = L(M_1)$ و $L_2 = L(M_2)$ باشد، که $M_1 = (Q, \Sigma, \delta_1, q_0, F_1)$ و $M_2 = (P, \Sigma, \delta_2, p_0, F_2)$ پذیرنده های متناهی قطعی هستند. از روی M_1 و M_2 ماشین ترکیبی $\hat{M} = (\hat{Q}, \Sigma, \hat{\delta}, (q_0, p_0), \hat{F})$ را می‌سازیم، که مجموعه حالت $\hat{Q} = Q \times P$ شامل زوج‌های (q_i, p_j) می‌باشد، و تابع انتقال $\hat{\delta}$ به گونه‌ای است که هرگاه M_1 در حالت q_i و M_2 در حالت p_j باشد، $\hat{M}$ در حالت (q_i, p_j) است. این مطلب بدین گونه انجام می‌شود که

$$\hat{\delta}((q_i, p_j), a) = (q_k, p_l),$$

می‌باشد هر گاه

$$\delta_1(q_i, a) = q_k$$

و

$$\delta_2(p_j, a) = p_l$$

باشد. $\hat{F}$ به عنوان مجموعه‌ای از همه (q_i, p_j) تعریف می‌شود، بطوریکه $q_i \in F_1$ و $p_j \in F_2$ می‌باشد. سپس موضوع ساده‌ای است که نشان دهیم $w \in L_1 \cap L_2$ اگر و فقط اگر بوسیله $\hat{M}$ پذیرفته شود. در نتیجه $L_1 \cap L_2$ منظم است. ■

اثبات بسته بودن تحت اشتراک، مثال خوبی از اثبات ساختاری است. این اثبات نه تنها نتیجه مطلوب را برقرار می‌سازد، بلکه صریحا چگونگی ساخت یک پذیرنده متناهی برای اشتراک دو زبان منظم را نشان می‌دهد. اثبات ساختاری که در طول این کتاب ظاهر شده است با اهمیت هستند زیرا بینشی در مورد نتایج به ما می‌دهند و اغلب به عنوان نقطه شروعی برای الگوریتم‌های عملی محسوب می‌شوند. در اینجا، همانند بسیاری از موارد، استدلال‌های کوتاه‌تر ولی غیر ساختاری (یا حداقل صریحاً غیر ساختاری) وجود دارند. در مورد بسته بودن تحت اشتراک، با قانون دمورگان، معادله (۱،۳) شروع می‌کنیم، و از دو طرف متمم می‌گیریم. در اینصورت برای هر زبان L_1 و L_2 داریم

$$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L}_2}$$

اکنون، اگر L_1 و L_2 منظم باشند، آنگاه با توجه به بسته بودن تحت متمم گیری، $\overline{L_1}$ و $\overline{L_2}$ نیز منظم هستند. با استفاده از بسته بودن تحت اجتماع، بدست می‌آوریم که $\overline{L_1} \cup \overline{L_2}$ نیز منظم است. با استفاده مجدد از بسته بودن تحت متمم گیری می‌بینیم که

$$\overline{\overline{L_1} \cup \overline{L}_2} = L_1 \cap L_2$$

منظم است.

مثال بعدی، تغییراتی روی همین ایده است.

مثال ۴-۱ : نشان دهید که خانواده زبان‌های منظم تحت تفاضل بسته است. به عبارت دیگر، می‌خواهیم نشان دهیم که اگر L_1 و L_2 منظم باشند، آنگاه $L_1 - L_2$ نیز ضرورتاً منظم است.

هویت مجموعه مورد نیاز فوراً از تعریف تفاضل مجموعه واضح است، یعنی

$$L_1 - L_2 = L_1 \cap \overline{L}_2.$$

این حقیقت که L_2 منظم است ایجاب می‌کند که $\overline{L}_2$ نیز منظم باشد. سپس بدلیل بسته بودن زبان‌های منظم تحت اشتراک، می‌دانیم که $L_1 \cap \overline{L}_2$ منظم است، و استدلال کامل می‌شود.

حواص بستاری دیگری می‌توانند مستقیماً بوسیله استدلال‌های مقدماتی، بدست آیند.

قضیه ٤-۲ : خانواده زبان‌های منظم تحت معکوس کردن بسته است.

اثبات : اثبات این قضیه به عنوان تمرینی در بخش ۲-۳ پیشنهاد شده بود. در اینجا به جزئیات آن می‌پردازیم. فرض کنید که L یک زبان منظم باشد. در اینصورت یک پذیرنده متناهی غیر قطعی با تنها یک حالت نهایی برای آن می‌سازیم. طبق تمرین ۷، بخش ۲-۳ این کار همواره امکان‌پذیر است. در گراف انتقال برای این پذیرنده متناهی غیر قطعی، راس اولیه را راس نهایی، و راس نهایی را راس اولیه می‌سازیم، و جهت همه یال‌ها را معکوس می‌نماییم. موضوع نسبتاً ساده‌ای است که نشان دهیم پذیرنده متناهی غیر قطعی تغییر یافته w^R را می‌پذیرد اگر و فقط اگر پذیرنده متناهی قطعی اولیه w را بپذیرد. بنابراین، پذیرنده متناهی غیر قطعی تغییر یافته L^R را می‌پذیرد، و بسته بودن تحت معکوس کردن اثبات می‌شود. ■

بستار تحت سایر عملیات

علاوه بر عملیات استاندارد روی زبان‌ها، می‌توان عملیات دیگری تعریف کرد و خواص بستاری آنها را مورد بررسی قرار داد. نتایج بسیاری در این زمینه وجود دارد که ما تنها دو نمونه را انتخاب می‌کنیم. بقیه نتایج در تمرینات انتهای این بخش بررسی می‌شوند.

تعریف ٤-۱ : فرض کنید Σ و Γ الفبا باشند. در اینصورت تابع

$$h : \Sigma \rightarrow \Gamma^*$$

را همریختی نامند. به عبارتی، همریختی یک جایگزینی است که یک حرف تنها با یک رشته جایگزین

می‌شود. دامنه تابع h به رشته‌ها با روشی واضح توسعه می‌یابد. اگر

$$w = a_1 a_2 \ldots a_n,$$

آنگاه

$$h(w) = h(a_1)h(a_2)\ldots h(a_n).$$

اگر L زبانی روی Σ باشد، آنگاه **تصویر همریختی** آن به صورت زیر تعریف می‌شود

$$h(L) = \{h(w) : w \in L\}.$$

مثال ۴-۲: فرض کنید $\Sigma = \{a,b\}$ و $\Gamma = \{a,b,c\}$ و h بصورت زیر تعریف شده باشد

$$h(a) = ab,$$
$$h(b) = bbc.$$

در اینصورت $h(aba) = abbbcab$. تصویر همریختی $L = \{aa, aba\}$، زبان $h(L) = \{abab, abbbcab\}$ می‌باشد.

اگر یک عبارت منظم r برای زبان L داشته باشیم، آنگاه یک عبارت منظم برای $h(L)$ می‌تواند به سادگی با اعمال همریختی بر روی هر نماد Σ از r بدست آید.

مثال ۴-۳: فرض کنید $\Sigma = \{a,b\}$ و $\Gamma = \{b,c,d\}$ و h بصورت زیر تعریف شده باشد

$$h(a) = dbcc,$$
$$h(b) = bdc.$$

اگر L زبان منظمی باشد که با عبارت زیر نشان داده شود

$$r = (a + b^*)(aa)^*,$$

آنگاه

$$r_1 = (dbcc + (bdc)^*)(dbccdbcc)^*$$

نشان دهنده زبان منظم $h(L)$ می‌باشد.

نتیجه عمومی زیر از بستار زبان‌های منظم تحت هر همریختی از این مثال با روشی واضح بدست می‌آید.

قضیه ٤-٣: فرض کنید h یک همریختی باشد. اگر L یک زبان منظم باشد، آنگاه تصویر همریختی آن یعنی $h(L)$ نیز منظم است. بنابراین خانواده زبان‌های منظم تحت همریختی‌های دلخواه بسته است.

اثبات : فرض کنید L یک زبان منظم باشد که با عبارت منظم r نشان داده می‌شود. $h(r)$ را با

جایگزینی $h(a)$ به ازای هر نماد $a \in \Sigma$ از r بدست می‌آوریم. می توان مستقیماً با مراجعه به تعریف یک عبارت منظم نشان داد که نتیجه، یک عبارت منظم است. به همین آسانی می‌توان دید که عبارت بدست آمده نشان دهنده $h(L)$ است. همه آنچیزی که باید نشان دهیم این است که برای هر $w \in L(r)$، $h(w)$ متناظر در $L(h(r))$ است و بر عکس برای هر v در $L(h(r))$، یک w در L وجود دارد بطوریکه $v = h(w)$ است. جزئیات را به عنوان تمرین باقی می‌گذاریم، و ادعا می‌کنیم که $h(L)$ منظم است. ■

تعریف ٤-٢ : فرض کنید L_1 و L_2 زبان‌هایی روی الفبای یکسانی باشند. در اینصورت **خارج قسمت راست** L_1 به L_2 بصورت زیر تعریف می‌شود

$$L_1 / L_2 = \{x : xy \in L_1 \text{ برای برخی } y \in L_2\}. \qquad (۴-۱)$$

برای تشکیل خارج قسمت راست L_1 به L_2، همه رشته‌هایی در L_1 را که دارای پسوندی متعلق به L_2 هستند در نظر می‌گیریم. هر یک از چنین رشته‌هایی پس از حذف این پسوند، متعلق به L_1 / L_2 می‌باشند.

مثال ۴-۴ : اگر

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

و

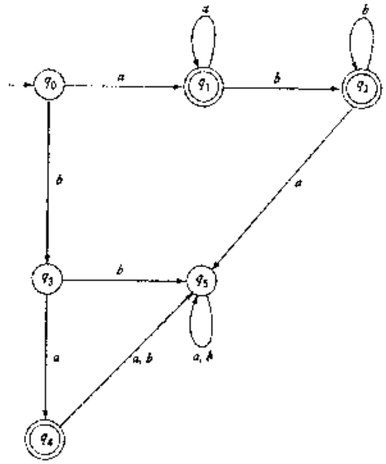
$$L_2 = \{b^m : m \geq 1\},$$

آنگاه

$$L_1 / L_2 = \{a^n b^m : n \geq 1, m \geq 0\}.$$

رشته های موجود در L_2 شامل یک یا بیشتر b هستند. بنابراین، با حذف یک یا بیشتر b از رشته هایی در L_1 که به حداقل یک b به عنوان پسوند ختم می‌شوند، به جواب می‌رسیم.

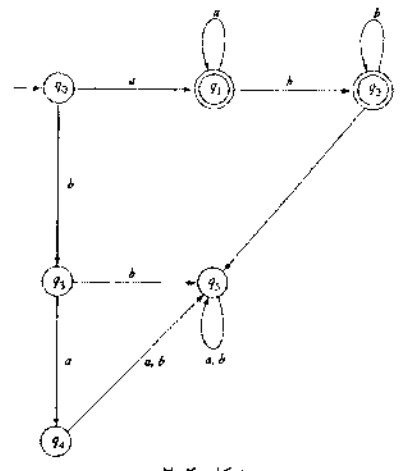
توجه کنید که در اینجا L_1، L_2، و L_1 / L_2 همگی منظم هستند. این مطلب پیشنهاد می‌کند که خارج قسمت راست هر دو عبارت منظم نیز منظم است. ما این را در قضیه بعد بوسیله ساختاری که پذیرنده‌های متناهی قطعی برای L_1 و L_2 را گرفته و از روی آنها پذیرنده متناهی قطعی برای L_1 / L_2 می‌سازد، اثبات خواهیم کرد. قبل از اینکه ساختار را به طور کامل توصیف کنیم، اجازه دهید ببینیم چگونه در این مثال به کار می‌رود. با پذیرنده متناهی قطعی مانند ماشین $M_1 = (Q, \Sigma, \delta, q_0, F)$ در شکل ۴-۱ برای L_1 شروع می‌کنیم. از آنجایی که یک ماشین برای L_1 / L_2 باید هر پیشوندی از رشته‌های L_1 را بپذیرد، سعی می‌کنیم M_1 را به گونه‌ای تغییر دهیم که x را بپذیرد اگر یک y وجود داشته باشد که در رابطه (۴-۱) صدق کند.



شکل ۴-۱

مشکل زمانی رخ می‌دهد که بخواهیم یک y به گونه‌ای بیابیم که $xy \in L_1$ و $y \in L_2$ باشد. برای حل آن، برای هر $q \in Q$ تعیین می‌کنیم که آیا راهی به یک حالت نهایی با برچسب v وجود دارد بطوریکه $v \in L_2$ باشد. اگر چنین باشد، هر x بطوریکه $\delta(q_0, x) = q$ باشد، در L_1 / L_2 خواهد بود. ما ماشین را مطابق آن تغییر می‌دهیم تا q یک حالت نهایی باشد.

برای کاربرد این در مورد مسئله ارائه شده، هر حالت $q_0, q_1, q_2, q_3, q_4, q_5$ را بررسی می‌کنیم تا ببینیم آیا راهی با برچسب bb^* به هر یک از حالات q_1, q_2 یا q_4 وجود دارد. می‌بینیم که فقط q_1 و q_2 واجد شرایط هستند. q_0, q_3, q_4 این شرایط را ندارند. ماشین نتیجه شده برای L_1 / L_2 در شکل ۴-۲ نشان داده شده است. آن را بررسی کنید تا ببینید که ساختار به درستی کار می‌کند. این ایده را در قضیه بعد تعمیم می‌دهیم.



شکل ۴-۲

قضیه ٤-٤ : اگر L_1 و L_2 زبان‌های منظم باشند، آنگاه L_1/L_2 نیز منظم است. می‌گوییم که خانواده زبان‌های منظم تحت خارج قسمت راست نسبت به یک زبان منظم، منظم است.

اثبات : فرض کنید $L_1 = L(M)$، که $M = (Q,\Sigma,\delta,q_0,F)$ یک پذیرنده متناهی قطعی است. ما پذیرنده متناهی قطعی دیگر $\hat{M} = (Q,\Sigma,\delta,q_0,\hat{F})$ را بصورت زیر می‌سازیم. برای هر $q_i \in Q$، تعیین می‌کنیم آیا یک $y \in L_2$ وجود دارد بطوریکه

$$\delta^*(q_i,y) = q_f \in F.$$

این کار می‌تواند بوسیله نگاه به پذیرنده‌های متناهی قطعی $M_i = (Q,\Sigma,\delta,q_i,F)$ انجام شود. ماشین M_i همان ماشین M است که در آن حالت q_0 با q_i جایگزین شده است. اکنون تعیین می‌کنیم آیا یک y در $L(M_i)$ وجود دارد که در L_2 نیز باشد. برای این کار، می‌توانیم از ساختاری که برای اشتراک دو زبان منظم که در قضیه ۴-۱ داده شد جهت یافتن گراف انتقال برای $L_2 \cap L(M_i)$ استفاده کنیم. اگر هر مسیری بین راس اولیه آن و هر راس نهایی وجود داشته باشد، آنگاه $L_2 \cap L(M_i)$ تهی نیست. در آن مورد، q_i را به $\hat{F}$ اضافه کنید. با تکرار این کار برای هر $q_i \in Q$، $\hat{F}$ را تعیین نموده و بدین وسیله $\hat{M}$ را می‌سازیم.

برای اثبات اینکه $L(\hat{M}) = L_1/L_2$ هست، فرض کنید x هر عنصری از L_1/L_2 باشد. در اینصورت باید یک $y \in L_2$ وجود داشته باشد بطوریکه $xy \in L_1$ باشد. این مورد ایجاب می‌کند که

$$\delta^*(q_0,xy) \in F,$$

بطوریکه باید یک $q \in Q$ داشته باشیم بطوریکه

$$\delta^*(q_0,x) = q$$

و

$$\delta^*(q,y) \in F.$$

بنابراین، طبق ساختار، $q \in \hat{F}$، و $\hat{M}$، x را می‌پذیرد، زیرا $\delta^*(q_0,x)$ در $\hat{F}$ است.

بر عکس برای هر x پذیرفته شده بوسیله $\hat{M}$، داریم

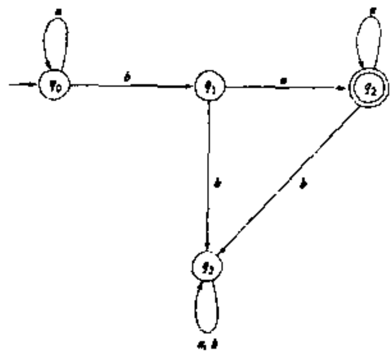
$$\delta^*(q_0,x) = q \in \hat{F}.$$

ولی مجدداً بر طبق ساختار، این مطلب ایجاب می‌کند که یک $y \in L_2$ وجود داشته باشد بطوریکه $\delta^*(q,y) \in F$ باشد. بنابراین xy در L_1، و x در L_1/L_2 است. بنابراین نتیجه می‌گیریم که

$$L(\hat{M}) = L_1/L_2,$$

و از این داریم که L_1/L_2 منظم است. ■

مثال ۴-۵: L_1/L_2 را برای زبان‌های زیر بیابید.



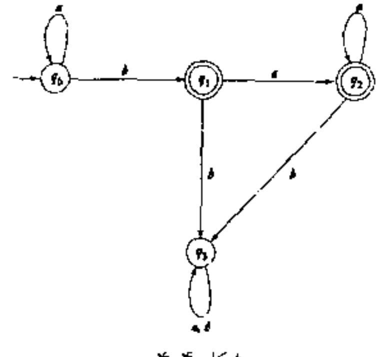
شکل ۴-۳

$$L_1 = L(a^*baa^*),$$
$$L_2 = L(ab^*).$$

ابتدا یک پذیرنده متناهی قطعی می‌یابیم که L_1 را بپذیرد. این کار آسان است، و یک راه حل در شکل ۴-۳ داده شده است. مثال به اندازه کافی ساده است، بطوریکه می‌توانیم از مقررات ساختار صرف نظر کنیم. از روی گراف شکل ۴-۳ واضح است که

$$L(M_0) \cap L_2 = \varnothing,$$
$$L(M_1) \cap L_2 = \{a\} \neq \varnothing,$$
$$L(M_2) \cap L_2 = \{a\} \neq \varnothing,$$
$$L(M_3) \cap L_2 = \varnothing.$$

بنابراین، ماشینی که L_1 / L_2 را می‌پذیرد، تعیین می‌شود. نتیجه در شکل ۴-۴ نشان داده شده است. این ماشین زبان نشان داده شده بوسیله عبارت منظم $a^*b + a^*baa^*$ را می‌پذیرد، که می‌تواند بصورت a^*ba^* ساده شود. بنابراین $L_1 / L_2 = L(a^*ba^*)$ می‌باشد.



شکل ۴-۴

تمرین‌ها

۱- جزئیات اثبات ساختاری در مورد بستار تحت اشتراک در قضیه ۴-۱ را مشخص نمایید.

۲- از ساختار قضیه ۴-۱ برای یافتن پذیرنده‌های متناهی قطعی که زبان‌های زیر را بپذیرند، استفاده کنید.

الف) $L((a+b)a^*) \bigcap L(baa^*)$. S

ب) $L(ab^*a^*) \bigcap L(a^*b^*a)$

۳- در مثال ۴-۱ بستار تحت تفاضل برای زبان‌های منظم را نشان دادیم، ولی اثبات غیر ساختاری بود. یک استدلال ساختاری برای این نتیجه ذکر کنید.

۴- در اثبات قضیه ۴-۳، نشان دهید که $h(r)$ یک عبارت منظم است. سپس نشان دهید که $h(r)$ نشان دهنده $h(L)$ می‌باشد.

۵- نشان دهید خانواده زبان‌های منظم تحت اجتماع و اشتراک متناهی، بسته است، یعنی اگر $L_1, L_2, \ldots, L_n$ منظم باشند، آنگاه

$$L_U = \bigcup_{i=\{1,2,\ldots,n\}} L_i$$

و

$$L_I = \bigcap_{i=\{1,2,\ldots,n\}} L_i$$

نیز منظم هستند.

۶- **تفاضل متقارن** دو مجموعه S_1 و S_2 بصورت زیر تعریف می‌شود

$S_1 \ominus S_2 = \{x : x \in S_1$ یا $x \in S_2$، ولی x در هر دو مجموعه S_1 و S_2 نیست$\}$

نشان دهید که خانواده زبان‌های منظم تحت تفاضل متقارن بسته است.

۷- nor دو زبان بصورت زیر تعریف می‌شود

$$nor(L_1, L_2) = \{w : w \notin L_1 \quad and \quad w \notin L_2\}.$$

نشان دهید که خانواده زبان‌های منظم تحت عملیات nor بسته است. S

۸- مکمل یا cor دو زبان بصورت زیر تعریف می‌شود

$$cor(L_1, L_2) = \{w : w \in \overline{L_1} \quad or \quad w \in \overline{L_2}\}.$$

نشان دهید که خانواده زبان‌های منظم تحت عملیات cor بسته است.

۹- کدام یک از روابط زیر برای همه زبان‌های منظم و همه همریختی‌ها درست است؟

الف) $h(L_1 \bigcup L_2) = h(L_1) \bigcup h(L_2)$

ب) $h(L_1 \bigcap L_2) = h(L_1) \bigcap h(L_2)$

ج) $h(L_1 L_2) = h(L_1)\, h(L_2)$

۱۰- فرض کنید $L_1 = L(a^*baa^*)$ و $L_2 = L(aba^*)$ باشد. L_1 / L_2 را بیابید.

۱۱- نشان دهید که رابطه $L_1 = L_1L_2 / L_2$ برای همه زبان‌های L_1 و L_2 درست نیست.

۱۲- فرض کنید می‌دانیم که $L_1 \cup L_2$ منظم است و اینکه L_1 متناهی است. آیا می‌توانیم نتیجه بگیریم که L_2 منظم است؟ *

۱۳- اگر L یک زبان منظم باشد، ثابت کنید که $L_1 = \{uv : u \in L, |v| = 2\}$ نیز منظم است.

۱۴- اگر L یک زبان منظم باشد، ثابت کنید که $\{uv : u \in L, v \in L^R\}$ نیز منظم است.

۱۵- خارج قسمت چپ زبان L_1 با توجه به L_2 بصورت زیر تعریف می‌شود

$$L_2 / L_1 = \{y : x \in L_2, xy \in L_1\}.$$

نشان دهید خانواده زبان‌های منظم تحت خارج قسمت چپ نسبت به یک زبان منظم بسته است.

۱۶- نشان دهید که اگر جمله "اگر L_1 منظم باشد و $L_1 \cup L_2$ نیز منظم باشد، آنگاه L_2 باید منظم باشد" برای همه زبان‌های L_1 و L_2 درست باشد، آنگاه همه زبان‌ها باید منظم باشند.

۱۷- *دنباله* یک زبان بصورت مجموعه همه پسوندهای رشته‌های آن تعریف می‌شود، یعنی

$$tail(L) = \{y : xy \in L, \quad x \in \Sigma^* \text{ برای برخی}\}$$

نشان دهید که اگر L منظم باشد، $tail(L)$ نیز منظم است.

۱۸- *عنوان* یک زبان، مجموعه همه پیشوندهایی از رشته‌های آن است، یعنی

$$head(L) = \{x : xy \in L, \quad y \in \Sigma^* \text{ برای برخی}\}$$

نشان دهید که خانواده زبان‌های منظم تحت این عملیات بسته است.

۱۹- یک عملیات سومین روی رشته‌ها و زبان‌ها بصورت زیر تعریف می‌کنیم

$$third(a_1a_2a_3a_4a_5a_6\ldots) = a_3a_6\ldots$$

به همراه توسعه مناسبی از این تعریف بر روی زبان‌ها، بستار خانواده زبان‌های منظم تحت این عملیات را ثابت کنید.

۲۰- برای رشته $a_1a_2\ldots a_n$ عملیات *جابجایی* را بصورت زیر تعریف می‌کنیم

$$shift(a_1a_2\ldots a_n) = a_2\ldots a_na_1.$$

از روی این، می‌توانیم عملیات مذکور را روی یک زبان بصورت زیر تعریف کنیم

$$shift(L) = \{v : v = shift(w), \quad w \in L \text{ برای برخی}\}.$$

نشان دهید منظم بودن تحت عملیات *جابجایی* حفظ می‌شود.

۲۱- تعریف می‌کنیم

$$exchange(a_1a_2\ldots a_{n-1}a_n)=a_na_2\ldots a_{n-1}a_1,$$

و

$$exchange(L)=\{v:v=exchange(w),\ \ w\in L\ \text{برای برخی}\}.$$

نشان دهید که خانواده زبان‌های منظم تحت تعویض بسته است.

۲۲- بر زدن دو زبان L_1 و L_2 بصورت زیر تعریف می‌شود

$$shuffle(L_1,L_2)=\{w_1v_1w_2v_2\ldots w_mv_m : w_1w_2\ldots w_m\in L_1$$
$$,v_1v_2\ldots v_m\in L_2, w_i, v_i\in\Sigma^*\}$$

نشان دهید که خانواده زبان های منظم تحت عملیات بر زدن بسته است. *

۲۳- عملیات *بدون۵* بر روی یک زبان L بصورت مجموعه همه رشته‌هایی از L که در آن پنجمین نماد از سمت چپ حذف شده است (رشته‌هایی با طول کمتر از پنج تغییری نمی‌کنند) تعریف شده است. نشان دهید که خانواده زبان‌های منظم تحت عملیات minus5 بسته است. *

۲۴- عملیات سمت چپ زبان L بصورت زیر تعریف می‌شود

$$leftside(L)=\{w: ww^R\in L\}.$$

آیا خانواده زبان‌های منظم تحت این عملیات بسته است؟ *

۲۵- کمینه یک زبان L بصورت زیر تعریف می‌شود

$$\min(L)=\{w\in L: w=uv\ \text{وجود نداشته باشد بطوریکه}\ u\in L, v\in\Sigma^+\ \text{هیچ}\}$$

نشان دهید خانواده زبان‌های منظم تحت عملیات کمینه بسته است.

۲۶- فرض کنید G_1 و G_2 دو گرامر منظم باشند. نشان دهید چگونه می‌توان گرامرهای منظمی برای زبان‌های زیر بدست آورد.

الف) $L(G_1)\cup L(G_2)$ $

ب) $L(G_1)L(G_2)$ $

ج) $L(G_1)^*$ $

۲-۴ سوالات مقدماتی درباره زبان‌های منظم

اکنون به یک موضوع بسیار اساسی می‌رسیم : یک زبان L و یک رشته w داده شده است، آیا می‌توانیم تعیین کنیم که آیا رشته w یک عنصر از L هست یا خیر؟ این سوال، سوال عضویت است و روشی برای پاسخ به آن را الگوریتم عضویت نامند. از زبان‌هایی که نتوانیم الگوریتم‌های عضویت کارآیی برای آنها بیابیم استفاده بسیار کمی می‌توان نمود. سوال وجود و طبیعت الگوریتم‌های عضویت در بحث‌های بعدی، مورد علاقه بسیاری خواهد بود، موضوعی که غالباً مشکل است. اگر چه برای زبان‌های منظم، موضوع ساده‌ای است.

ابتدا باید در نظر بگیریم هنگامی که می‌گوییم " یک زبان منظم داده شده است ... " دقیقاً چه منظوری داریم. در بسیاری از استدلال‌ها، مهم است که این مورد غیر مبهم باشد. راه‌های متعددی برای توصیف زبان‌های منظم داریم : توصیف‌های کلامی غیر رسمی، نماد مجموعه، ماشین‌های متناهی، عبارات منظم، و گرامرهای منظم. فقط سه مورد اخیر برای استفاده در قضایا کاملاً خوب تعریف شده‌اند. بنابراین گوییم که یک زبان منظم بصورت **نمایش استاندارد** داده شده است اگر و فقط اگر بوسیله یک ماشین متناهی، یک عبارت منظم، یا یک گرامر منظم توصیف شده باشد.

قضیه ٤-٥ : یک نمایش استاندارد از هر زبان منظم L روی Σ و هر $w \in \Sigma^*$ داده شده است، الگوریتمی برای تعیین اینکه آیا w در L هست یا خیر، وجود دارد.

اثبات : زبان را بوسیله پذیرنده متناهی قطعی ارائه می‌کنیم، سپس w را آزمایش می‌کنیم تا ببینیم آیا بوسیله این ماشین پذیرفته می‌شود. ■

سوالات مهم دیگری که مطرح است اینکه آیا یک زبان متناهی یا نامتناهی است، آیا دو زبان یکسان هستند، و آیا یک زبان زیر مجموعه دیگری است. حداقل برای زبان‌های منظم این سوالات بسادگی پاسخ داده می‌شوند.

قضیه ٤-٦ : برای تعیین اینکه آیا یک زبان منظم، که بصورت نمایش استاندارد داده شده است تهی، متناهی یا نامتناهی است، الگوریتمی وجود دارد.

اثبات : اگر زبان را بصورت یک گراف انتقال یک پذیرنده متناهی قطعی نمایش دهیم. جواب واضح است. اگر مسیر ساده‌ای از راس اولیه به هر راس نهایی وجود داشته باشد، آنگاه زبان تهی نیست.
برای تعیین اینکه آیا یک زبان، نامتناهی هست یا خیر، همه رئوسی که پایه یک چرخه هستند می‌یابیم. اگر هر یک از این‌ها روی مسیری از راس اولیه به یک راس نهایی باشند، زبان نامتناهی است. در غیر اینصورت متناهی است. ■

سوال هم ارزی دو زبان نیز یک موضوع مهم عملی است. اغلب چندین تعریف از یک زبان برنامه سازی وجود دارد، و نیاز داریم که بدانیم آیا بر خلاف ظاهر متفاوت آنها، زبان یکسانی را مشخص می‌کنند. عموما این مسئله مشکل است: استدلال حتی برای زبان‌های منظم، واضح نیست. امکان مقایسه جمله به جمله وجود ندارد، زیرا این کار فقط برای زبان‌های متناهی قابل انجام است. ضمناً جواب دادن بوسیله نگاه به عبارات منظم، گرامرهای منظم، یا پذیرنده‌های متناهی قطعی آسان نیست. یک راه حل دقیق از خواص بستاری که قبلاً اثبات شده است استفاده می‌کند.

قضیه ٤-٧ : نمایش‌های استاندارد دو زبان منظم L_1 و L_2 داده شده است، الگوریتمی وجود دارد که تعیین می‌کند آیا $L_1 = L_2$ هست یا خیر.

اثبات : با استفاده از L_1 و L_2 زبان زیر را تعریف می‌کنیم

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

بر طبق بستار، L_3 منظم است، و ما می‌توانیم یک پذیرنده متناهی قطعی M بیابیم که L_3 را بپذیرد. هنگامیکه M را داشتیم، می توانیم از الگوریتم قضیه ۴-۶ استفاده کنیم تا تعیین کنیم آیا L_3 تهی است. ولی از روی تمرین۸، بخش ۱-۱ می‌بینیم که $L_3 = \varnothing$ اگر و فقط اگر $L_1 = L_2$ باشد. ■

این نتایج بر خلاف اینکه واضح و غیر تعجب برانگیز هستند، اساسی هستند. برای زبان‌های منظم، سوالات مطرح شده بوسیله قضایای ۴-۵ تا ۴-۷ می‌توانند بسادگی پاسخ داده شوند، ولی در مواجهه با خانواده‌های بزرگتر زبان‌ها همیشه اینگونه نیست. به سوالاتی شبیه این‌ها در موارد متعددی برخورد خواهیم کرد. هنگامی کمی جلوتر برویم، خواهیم دید که جواب‌ها به طور فزاینده مشکل‌تر می‌شوند، و سرانجام یافتن آنها غیر ممکن خواهد بود.

☐ تمرین‌ها

برای همه تمارین در این بخش، فرض کنید که زبان‌های منظم به شکل نمایش استاندارد داده شده باشند.

۱- نشان دهید که برای هر w و هر زبان منظم L_1 و L_2 داده شده، الگوریتمی برای تعیین اینکه آیا $w \in L_1 - L_2$ هست یا خیر، وجود دارد.

۲- نشان دهید که برای هر زبان منظم L_1 و L_2، الگوریتمی برای تعیین آیا $L_1 \subseteq L_2$ هست وجود دارد.

۳- نشان دهید که برای هر زبان منظم L، الگوریتمی برای تعیین اینکه آیا $\lambda \in L$ هست وجود دارد.

۴- نشان دهید که برای هر زبان منظم L_1 و L_2، الگوریتمی برای تعیین اینکه آیا $L_1 = L_1 / L_2$ هست یا خیر، وجود دارد.

۵- یک زبان را زبان *مقلوب* گویند اگر $L = L^R$ باشد. الگوریتمی برای تعیین اینکه آیا یک زبان منظم داده شده، یک زبان مقلوب است بیابید.

۶- الگوریتمی برای تعیین اینکه آیا یک زبان منظم L شامل هر رشته w هست یا خیر بطوریکه $w^R \in L$ باشد ارائه دهید.

۷- الگوریتمی ارائه دهید که با داشتن سه زبان منظم L, L_1, L_2، تعیین کند آیا $L = L_1 L_2$ هست یا خیر.

۸- الگوریتمی ارائه دهید که با داشتن هر زبان منظم L، تعیین کند آیا $L = L^*$ هست یا خیر.

۹- فرض کنید L یک زبان منظم روی Σ و $\tilde{w}$ هر رشته ای در Σ^* باشد. الگوریتمی بیابید که تعیین کند آیا L شامل هر w هست بطوریکه $\tilde{w}$ زیر رشته‌ای از آن باشد، یعنی بطوریکه $w = u\tilde{w}v$ که $u, v \in \Sigma^*$ باشد.

۱۰- نشان دهید برای تعیین اینکه برای هر زبان L آیا $L = shuffle(L, L)$ هست، الگوریتمی وجود دارد.

۱۱- عملیات $tail(L)$ بصورت زیر تعریف شده است

$$tail(L) = \{v : uv \in L, u, v \in \Sigma^*\}.$$

نشان دهید که برای تعیین اینکه آیا $L = tail(L)$ برای هر زبان L هست یا خیر، الگوریتمی وجود دارد.

۱۲- فرض کنید L هر زبان منظمی روی $\Sigma = \{a, b\}$ باشد. نشان دهید برای تعیین اینکه آیا L شامل رشته‌هایی با طول زوج است، الگوریتمی وجود دارد.

۱۳- الگوریتمی برای تعیین اینکه آیا یک زبان منظم L شامل تعداد نامتناهی از رشته‌هایی با طول زوج است بیابید.

۱۴- الگوریتمی توصیف کنید که با گرفتن یک گرامر منظم G، بتواند به ما بگوید آیا $L(G) = \Sigma^*$ هست یا خیر.

۴-۳ تشخیص زبان‌های غیرمنظم

زبان‌های منظم می توانند مانند اکثر مثال‌های مشروح ما نامتناهی باشند. حقیقت این است که زبان‌های منظم با ماشین‌هایی ارتباط دارند که دارای حافظه متناهی هستند، به هر حال، این باعث محدودیت‌هایی روی ساختار یک زبان منظم می گردد. برخی محدودیت‌های شدید باید رعایت شود اگر حفظ منظم بودن لازم باشد. حدس می‌زنیم که یک زبان منظم است فقط اگر در پردازش هر رشته، اطلاعاتی که باید در هر مرحله به خاطر آورده شود، کاملاً محدود باشد. این مطلب درست است، ولی باید دقیقاً نشان داده شود تا به روشی معنادار قابل استفاده باشد. چندین راه برای تحقق این دقت وجود دارد.

استفاده از اصل لانه کبوتر

اصطلاح "اصل لانه کبوتر" بوسیله ریاضی‌دانان برای مراجعه به مشاهده ساده زیر استفاده می‌شود. اگر ما n شی را در m جعبه (لانه کبوتر) قرار دهیم، و اگر $n > m$ باشد، آنگاه حداقل یک جعبه باید بیش از یک قلم را در خود داشته باشد. این حقیقت به اندازه‌ای واضح است که نتایج عمیقی که می‌تواند از آن بدست آید باعث تعجب می‌شود.

مثال ۴-۶ : آیا زبان $L = \{a^n b^n : n \geq 0\}$ منظم است؟ جواب، چنانچه ما با استفاده از اثبات از طریق تناقض نشان می‌دهیم، خیر است.

فرض کنید L منظم باشد. در اینصورت یک پذیرنده متناهی قطعی $M=(Q,\{a,b\},\delta,q_0,F)$ برای آن وجود دارد. اکنون به $\delta^*(q_0,a^i)$ به ازای $i=1,2,3,\ldots$ بنگرید. از آنجایی که تعداد نامحدودی از i ها وجود دارند، ولی تعداد محدودی از حالات در M هستند، اصل لانه کبوتر به ما می‌گوید که باید یک حالت مانند q وجود داشته باشد، بطوریکه

$$\delta^*(q_0,a^n)=q$$

و

$$\delta^*(q_0,a^m)=q$$

که $n\neq m$ می‌باشد. ولی از آنجایی که M، a^nb^n را می‌پذیرد، باید داشته باشیم

$$\delta^*(q,b^n)=q_f\in F.$$

از این می‌توانیم نتیجه بگیریم که

$$\begin{aligned}\delta^*(q_0,a^mb^n)&=\delta^*(\delta^*(q_0,a^m),b^n)\\&=\delta^*(q,b^n\\&=q_f.\end{aligned}$$

این با فرض اولیه در تناقض است که M، a^mb^n را می‌پذیرد اگر $n=m$ باشد، و ما را به این نتیجه رهنمون می‌کند که L نمی‌تواند منظم باشد.

▲

در این استدلال، اصل لانه کبوتر روشی برای بیان دقیق این جمله است که یک ماشین متناهی دارای حافظه محدود است. برای پذیرش هر a^nb^n، یک ماشین باید بین همه پیشوندهای a^n و a^m تمایز قائل شود. ولی از آنجا که تعداد حالات داخلی، متناهی است، n و m وجود دارند که نتوان بین آنها تمایزی قائل شد.

به منظور استفاده از این نوع استدلال در موقعیت‌های مختلف، مناسب است آن را به صورت یک قضیه عمومی تدوین کنیم. چندین راه برای انجام این کار وجود دارد. راهی که ما در اینجا ارائه می‌کنیم شاید مشهورترین آنها باشد.

یک لم تزریق

نتیجه زیر، که به **لم تزریق** برای زبان‌های منظم معروف است، از اصل لانه کبوتر به شکل دیگری استفاده می‌کند. اثبات بر این پایه استوار است که در یک گراف انتقال با n راس، هر راهی با طول n یا بیشتر باید یک راس را تکرار کند، یعنی دارای یک چرخه باشد.

قضیه ٤-٨ : فرض کنید L یک زبان منظم نامتناهی باشد. در اینصورت یک عدد صحیح مثبت m وجود دارد بطوریکه هر $w\in L$ با $|w|\geq m$ می‌تواند بصورت زیر تجزیه شود

$$w = xyz,$$

با

$$|xy| \le m,$$

و

$$|y| \ge 1,$$

بطوریکه

$$w_i = xy^i z, \tag{۲-۴}$$

به ازای همه $i = 0,1,2,\ldots$ نیز در L باشد.

در تشریح این، هر رشته به طول کافی در L را می‌توان به سه قسمت تجزیه کرد بطوریکه تکرار به تعداد دلخواه در قسمت وسط منجر به رشته دیگری در L بشود. گوییم که رشته وسط "تزریق شده است"، از این رو اصطلاح لم تزریق برای این نتیجه انتخاب شده است.

اثبات : اگر L منظم باشد، یک پذیرنده متناهی قطعی وجود دارد که آن را تشخیص دهد. فرض کنید چنین پذیرنده متناهی قطعی دارای حالاتی با برچسب‌های $q_0, q_1, q_2, \ldots, q_n$ باشند. اکنون رشته w در L را انتخاب کنید بطوریکه $|w| \ge m = n+1$ باشد. از آنجایی که L نامتناهی فرض شده است، همواره می توان این کار را انجام داد. مجموعه حالاتی که ماشین در حین پردازش w طی می‌کند، بصورت زیر در نظر بگیرید

$$q_0, q_i, q_j, \ldots, q_f.$$

از آنجایی که این دنباله دقیقاً دارای $|w|+1$ قلم است، حداقل یک حالت باید تکرار شده باشد، و چنین تکراری نباید پس از nامین حرکت صورت گیرد. بنابراین دنباله باید مشابه زیر باشد

$$q_0, q_i, q_j, \ldots, q_r, \ldots, q_r, \ldots, q_f,$$

که نشان می‌دهد باید زیر رشته‌های x, y, z از w باشند بطوریکه

$$\delta^*(q_0, x) = q_r,$$
$$\delta^*(q_r, y) = q_r,$$
$$\delta^*(q_r, z) = q_f,$$

با $|xy| \le n+1 = m$ و $|y| \ge 1$. از این فوراً داریم که

$$\delta^*(q_0, xz) = q_f,$$

و نیز

$$\delta^*(q_0, xy^2z) = q_f,$$
$$\delta^*(q_0, xy^3z) = q_f,$$

و مانند آن، و اثبات قضیه تکمیل می‌شود. ■

ما لم تزریق را فقط برای زبان‌های نامتناهی ارائه دادیم. اگر چه زبان‌های متناهی همواره منظم هستند، نمی‌توانند تزریق شوند زیرا تزریق بصورت خودکار مجموعه‌ای نامتناهی ایجاد می‌کند. این قضیه برای زبان‌های متناهی برقرار است، ولی بی معنا است. فرض می‌شود که m در لم تزریق بزرگتر از طولانی‌ترین رشته باشد، بطوریکه هیچ رشته‌ای نتواند تزریق شود.

لم تزریق، شبیه استدلال لانه کبوتر در مثال ۴-۶، برای نشان دادن اینکه زبان‌های خاصی منظم نیستند استفاده می‌شود. اثبات همواره بوسیله تناقض است. همچنان که در اینجا بیان کردیم، هیچ چیزی در لم تزریق وجود ندارد، و نمی‌توان از آن برای اثبات منظم بودن یک زبان استفاده کرد. اگر چه می‌توانیم نشان دهیم (و این معمولاً کاملاً مشکل است) که هر رشته تزریق شده باید در زبان اولیه باشد، هیچ چیزی در قضیه ۴-۸ بیان نشده است که بتوانیم از آن نتیجه بگیریم که زبان مورد بحث منظم است.

مثال ۴-۷: با استفاده از لم تزریق نشان دهید که $L = \{a^n b^n : n \geq 0\}$ منظم نیست.

فرض کنید که L منظم باشد، بطوریکه لم تزریق برقرار باشد. ما مقدار m را نمی‌دانیم، ولی هر چه باشد، همواره می‌توانیم $n = m$ را انتخاب کنیم. بنابراین، زیر رشته y باید فقط از aها تشکیل شده باشد. فرض کنید $|y| = k$ باشد. در اینصورت رشته بدست آمده با استفاده از $i = 0$ در معادله (۴-۲) عبارت است از

$$w_0 = a^{m-k} b^m$$

و مسلماً در L نیست. این با لم تزریق تناقض دارد و بدین وسیله نشان می‌دهد که فرض منظم بودن L نادرست است. ▲

در کاربرد لم تزریق، باید به خاطر داشته باشیم که قضیه چه می‌گوید. ما وجود m و تجزیه xyz را تضمین می‌کنیم، ولی مقادیر آنها را نمی‌دانیم. ما به دلیل اینکه برای مقادیر خاصی از m یا xyz لم تزریق نقض می شود، نمی‌توانیم ادعا کنیم که به تناقض رسیدیم. از طرف دیگر، لم تزریق برای هر $w \in L$ و هر i برقرار است. بنابراین، اگر لم تزریق حتی برای یک w یا i نقض شود، آنگاه زبان نمی‌تواند منظم باشد.

استدلال صحیح می‌تواند به عنوان یک بازی نگریسته شود که ما در مقابل یک رقیب بازی می‌کنیم. هدف ما برد بازی بوسیله برقراری یک تناقض در لم تزریق است، در حالیکه رقیب سعی در شکست ما دارد. چهار حرکت در بازی وجود دارد.

۱-رقیب m را انتخاب می‌کند.

۲-با داشتن m، رشته w در L با طول بزرگتر یا مساوی m را انتخاب می‌کنیم. ما با توجه به $w \in L$ و $|w| \geq m$ آزاد هستیم که هر w انتخاب کنیم.

۳-رقیب تجزیه xyz را با توجه به $|xy| \leq m, |y| \geq 1$ انتخاب می‌کند. ما فرض می‌کنیم که رقیب انتخاب را به گونه‌ای انجام دهد که سخت ترین حالت برای ما جهت بردن بازی است.

۴-ما سعی می‌کنیم i را به گونه‌ای انتخاب کنیم که رشته تزریق شده w_i، که در معادله (۴-۲) تعریف شد، در L نباشد. اگر بتوانیم این کار را انجام دهیم، بازی را برده‌ایم.

یک استراتژی که به ما با هر گونه انتخابی که حریف انجام دهد اجازه برد می‌دهد، اثبات این است که زبان منظم نیست. در اینجا، مرحله ۲ بسیار مهم است. در عین حال که نمی‌توانیم رقیب را به انتخاب تجزیه خاصی از w وادار نماییم، ممکن است بتوانیم w را به گونه‌ای انتخاب کنیم که رقیب در مرحله ۳ بسیار محدود باشد، بطوریکه مجبور به انتخاب x,y و z به گونه‌ای باشد که به ما اجازه تولید نقضی از لم تزریق روی حرکت بعدیمان را بدهد.

مثال ۴-۸: فرض کنید $\Sigma = \{a,b\}$ باشد. نشان دهید که

$$L = \{ww^R : w \in \Sigma^*\}$$

منظم نیست.

هر مقداری را که رقیب برای m در مرحله ۱ انتخاب می‌کند، همواره می‌توانیم یک w را مطابق شکل ۴-۵ انتخاب کنیم. بدلیل این انتخاب، و لزوم اینکه $|xy| \leq m$ باشد، رقیب در مرحله ۳ به انتخاب یک y محدود می‌شود که شامل تماماً aها باشد. در مرحله ۴، از $i = 0$ استفاده می‌کنیم. رشته بدست آمده از این طریق دارای aهای کمتری در سمت چپ نسبت به سمت راست می‌باشد و بنابراین نمی‌تواند به شکل ww^R باشد. بنابراین L منظم نیست.

توجه کنید اگر w را خیلی کوتاه انتخاب کرده بودیم، آنگاه رقیب می‌توانست یک y با تعداد زوجی از bها انتخاب کند. در آن صورت، نمی‌توانستیم در مرحله آخر به تناقضی در لم تزریق برسیم. همچنین اگر رشته‌ای شامل تماماً aها مانند زیر انتخاب می‌کردیم

$$w = a^{2m},$$

که در L هست، شکست می‌خوردیم. رقیب برای شکست دادن ما نیاز به انتخاب زیر داشت

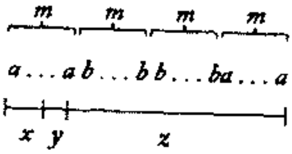
$$y = aa.$$

اکنون w_i برای هر i در L هست، و ما می‌باختیم.

در اعمال لم تزریق نمی‌توانیم فرض کنیم که رقیب حرکتی اشتباه اشتباهی انجام دهد. اگر در حالیکه ما $w = a^{2m}$ را انتخاب کردیم، رقیب

$$y = a,$$

را انتخاب نماید، آنگاه w_0 رشته‌ای با طول فرد خواهد بود و بنابراین در L نخواهد بود. ولی هر استدلالی که فرض کند رقیب چنین کاری انجام می‌دهد، به طور خودکار نادرست است.



شکل۴-۵

مثال ۴-۹: فرض کنید $\Sigma = \{a, b\}$ باشد. زبان

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

منظم نیست.

فرض کنید m به ما داده شده باشد. از آنجایی که ما در انتخاب w آزادی کامل داریم، $w = a^m b^{m+1}$ را انتخاب می‌کنیم. اکنون، بدلیل اینکه $|xy|$ نمی‌تواند بزرگتر از m باشد، رقیب به جز انتخاب یک y که شامل تماماً aها باشد، نمی‌تواند کاری انجام دهد، یعنی

$$y = a^k, \qquad 1 \le k \le m.$$

اکنون، با استفاده از $i = 2$ عمل تزریق را انجام می‌دهیم. رشته حاصل شده

$$w_2 = a^{m+k} b^{m+1}$$

در L نیست. بنابراین، لم تزریق نقض می‌شود، و L منظم نیست.

مثال۴-۱۰: زبان

$$L = \{(ab)^n a^k : n > k, k \ge 0\}$$

منظم نیست.

با داشتن m، رشته مان را بصورت

$$w = (ab)^{m+1} a^m$$

انتخاب می‌کنیم که در L هست. بدلیل محدودیت $|xy| \le m$، هم x و هم y باید بخشی از رشته ساخته شده از abها باشند. انتخاب x روی استدلال تاثیری ندارد، بنابراین اجازه دهید ببینیم با y چه کاری می‌توانیم انجام دهیم. اگر رقیب ما $y = a$ را انتخاب نماید، ما $i = 0$ را انتخاب می‌کنیم و رشته‌ای را بدست می‌آوریم که در $L((ab)^* a^*)$ نیست. اگر رقیب $y = ab$ را انتخاب نماید، ما می‌توانیم مجدداً $i = 0$ را انتخاب کنیم. اکنون رشته $(ab)^m a^m$، را بدست می‌آوریم که در L نیست. به همین روش، می‌توانیم به ازای هر انتخاب ممکنی بوسیله رقیب برخورد کنیم، و بدین وسیله ادعای ما اثبات می‌شود.

مثال۴-۱۱: نشان دهید که

$$L = \{a^{n!} : n \ge 0\}$$

منظم نیست.

با انتخاب m توسط رقیب، رشته $a^{m!}$ را به عنوان w انتخاب می‌کنیم (مگر اینکه رقیب $m < 3$ را انتخاب کرده باشد، که در این مورد می‌توانیم از $a^{3!}$ به عنوان w استفاده کنیم). واضح است که تجزیه‌های مختلف w فقط در طول زیر رشته‌ها با یکدیگر تفاوت دارند. رقیب y را به گونه‌ای انتخاب می‌کند که

$$|y| = k \le m.$$

سپس به xz می‌نگریم که دارای طول $m!-k$ می‌باشد. این رشته در L هست اگر و فقط اگر یک j وجود داشته باشد بطوریکه

$$m!-k = j!$$

ولی این غیر ممکن است، زیرا برای $m > 2$ و $k \le m$ داریم

$$m!-k > (m-1)!$$

بنابراین زبان مذکور منظم نیست.

در برخی موارد، خواص بستاری می‌تواند برای ارتباط یک مسئله داده شده به مسئله‌ای که قبلا حل شده مورد استفاده قرار گیرد. این کار ممکن است بسیار ساده‌تر از کاربرد مستقیم لم تزریق باشد.

مثال ۴-۱۲ : نشان دهید که زبان

$$L = \{a^n b^k c^{n+k} : n \ge 0, k \ge 0\}$$

منظم نیست.

اعمال مستقیم لم تزریق مشکل نیست، ولی حتی آسان‌تر از آن استفاده از بستار تحت همریختی است. فرض کنید

$$h(a) = a, h(b) = a, h(c) = c$$

در اینصورت

$$h(L) = \{a^{n+k} c^{n+k} : n + k \ge 0\}$$
$$= \{a^i c^i : i \ge 0\},$$

ولی می‌دانیم که این زبان منظم نیست، بنابراین L نیز نمی‌تواند منظم باشد.

مثال ۴-۱۳ : نشان دهید که زبان

$$L = \{a^n b^l : n \ne l\}$$

منظم نیست.

در اینجا برای اعمال مستقیم لم تزریق نیاز به کمی خلاقیت داریم. یک رشته با $n = l + 1$ یا $n = l + 2$ انتخاب نخواهد شد، زیرا رقیب ما می‌تواند همواره تجزیه‌ای را انتخاب کند که تزریق رشته خارج از زبان غیر ممکن باشد (یعنی، تزریق آن نیز دارای تعداد مساوی از aها و bها باشد). ما باید خلاق‌تر باشیم. فرض کنید $n = m!$ و $l = (m+1)!$ را انتخاب کنیم. اکنون اگر رقیب یک y (که